

*Project:* DRA FRONT END FILTER PROJECT

*Title:* ProofPower Theory Listings

*Ref:* DS/FMU/FEF/017

*Issue: Revision : 2.2*

*Date:* 5 June 2016

*Status:* Draft

*Type:* Specification

*Keywords:*

*Author:*

<i>Name</i>	<i>Location</i>	<i>Signature</i>	<i>Date</i>
G. M. Prout	WIN01		

*Authorisation for Issue:*

<i>Name</i>	<i>Function</i>	<i>Signature</i>	<i>Date</i>
R.B. Jones	HAT Manager		

*Abstract:* This document contains listings of all the ProofPower theories used in the DRA front end filter project RSRE 1C/6130.

*Distribution:* HAT FEF File  
Simon Wiseman

---

## 0 DOCUMENT CONTROL

### 0.1 Contents List

0	DOCUMENT CONTROL	2
1	GENERAL	4
2	THE THEORY basic_hol	5
3	THE THEORY bin_rel	5
4	THE THEORY cache'fef	9
5	THE THEORY char	10
6	THE THEORY combin	10
7	THE THEORY dyadic	11
8	THE THEORY fef003	14
9	THE THEORY fef004	15
10	THE THEORY fef005	28
11	THE THEORY fef006	35
12	THE THEORY fef007	36
13	THE THEORY fef009	38
14	THE THEORY fef010	39
15	THE THEORY fef011	42
16	THE THEORY fef012	49
17	THE THEORY fef013	56
18	THE THEORY fef014	62
19	THE THEORY fef015	111
20	THE THEORY fef021	113
21	THE THEORY fef022	117
22	THE THEORY fef024	121
23	THE THEORY fef025	124
24	THE THEORY fef026	130

---

25 THE THEORY fef028	136
26 THE THEORY fef029	153
27 THE THEORY fef031	189
28 THE THEORY fef032	191
29 THE THEORY fef033	207
30 THE THEORY fef034	215
31 THE THEORY fef035	218
32 THE THEORY fef036	228
33 THE THEORY fef040	231
34 THE THEORY fef042	235
35 THE THEORY fef043	241
36 THE THEORY fin_set	248
37 THE THEORY fin_thms	250
38 THE THEORY fun_rel	252
39 THE THEORY fun_rel_thms	254
40 THE THEORY hol	255
41 THE THEORY init	256
42 THE THEORY lib_thms	256
43 THE THEORY list	258
44 THE THEORY log	261
45 THE THEORY min	263
46 THE THEORY misc	263
47 THE THEORY one	266
48 THE THEORY orders	267
49 THE THEORY pair	272
50 THE THEORY seq	274
51 THE THEORY sets	276

---

---

<b>52 THE THEORY</b> set_thms	<b>279</b>
<b>53 THE THEORY</b> sum	<b>280</b>
<b>54 THE THEORY</b> wrk049	<b>282</b>
<b>55 THE THEORY</b> wrk057	<b>289</b>
<b>56 THE THEORY</b> $\mathbb{R}$	<b>294</b>
<b>57 THE THEORY</b> $\mathbb{N}$	<b>312</b>
<b>58 THE THEORY</b> $\mathbb{Z}$	<b>317</b>
<b>59 INDEX</b>	<b>325</b>

## 0.2 Document Cross References

## 0.3 Changes History

**Issue Revision : 2.2 (5 June 2016)** Version which lists all fe theories as well as ProofPower theories.

**Issue 2.3** Removed dependency on ICL logo font

## 0.4 Changes Forecast

None.

# 1 GENERAL

## 1.1 Scope

This document contains listings of all the ProofPower release 0.4 theories which are used in the DRA front end filter project RSRE 1C/6130.

## 1.2 Introduction

This document provides listings of all the theories in ProofPower release 0.4 and all the “fef” theories. Some general library results are provided in work files *wrk044*, *wrk046* and *wrk049*, which produce theories *fin\_thms*, *lib\_thms* and *wrk049* respectively. Theories *fin\_thms* and *lib\_thms* are supplied with the ProofPower release 0.4. In addition, the theory *cache\_thms* is listed. System functions use this theory to cache various definitions and theorems eg. labelled product definitions.

## 2 THE THEORY basic\_hol

### 2.1 Parents

*char*

### 2.2 Children

*cache'fef sets combin one*

### 2.3 Notes

This theory is a cache theory; its contents have not been listed.

## 3 THE THEORY bin\_rel

### 3.1 Parents

*hol*

### 3.2 Children

*fun\_rel*

### 3.3 Constants

$\$ \mapsto$	$'a \rightarrow 'b \rightarrow 'a \times 'b$
$\$ \times$	$'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow 'a \leftrightarrow 'b$
$\$ \leftrightarrow$	$'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$
<b>Dom</b>	$'a \leftrightarrow 'b \rightarrow 'a \mathbb{P}$
<b>Ran</b>	$'a \leftrightarrow 'b \rightarrow 'b \mathbb{P}$
<b>Id</b>	$'a \mathbb{P} \rightarrow 'a \leftrightarrow 'a$
<b>Graph</b>	$('a \rightarrow 'b) \rightarrow 'a \leftrightarrow 'b$
$\$ \circ$	$('a \rightarrow 'b) \rightarrow ('b \rightarrow 'c) \rightarrow 'a \rightarrow 'c$
$\$ \mathbf{R}_{\circ} \mathbf{-R}$	$'a \leftrightarrow 'b \rightarrow 'b \leftrightarrow 'c \rightarrow 'a \leftrightarrow 'c$
$\$ \mathbf{R}_{\circ} \mathbf{o} \mathbf{-R}$	$'b \leftrightarrow 'c \rightarrow 'a \leftrightarrow 'b \rightarrow 'a \leftrightarrow 'c$
$\$ \triangleleft$	$'a \mathbb{P} \rightarrow 'a \leftrightarrow 'b \rightarrow 'a \leftrightarrow 'b$
$\$ \triangleright$	$'a \leftrightarrow 'b \rightarrow 'b \mathbb{P} \rightarrow 'a \leftrightarrow 'b$
$\$ \triangleleft$	$'a \mathbb{P} \rightarrow 'a \leftrightarrow 'b \rightarrow 'a \leftrightarrow 'b$
$\$ \triangleright$	$'a \leftrightarrow 'b \rightarrow 'b \mathbb{P} \rightarrow 'a \leftrightarrow 'b$
<b>InvRel</b>	$'a \leftrightarrow 'b \rightarrow 'b \leftrightarrow 'a$
$\$ \mathbf{Image}$	$'a \leftrightarrow 'b \rightarrow 'a \mathbb{P} \rightarrow 'b \mathbb{P}$
<b>Reflexive</b>	$('a \leftrightarrow 'a) \mathbb{P}$
<b>Symmetric</b>	$('a \leftrightarrow 'a) \mathbb{P}$
<b>Transitive</b>	$('a \leftrightarrow 'a) \mathbb{P}$

<b>Injective</b>	$('a \leftrightarrow 'b) \mathbb{P}$
<b>Surjective</b>	$'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$
<b>Total</b>	$'a \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$
<b>Functional</b>	$('a \leftrightarrow 'b) \mathbb{P}$
$\$ \oplus$	$'a \leftrightarrow 'b \rightarrow 'a \leftrightarrow 'b \rightarrow 'a \leftrightarrow 'b$
$\$ +$	$'a \leftrightarrow 'a \rightarrow 'a \leftrightarrow 'a$
$\$ *$	$'a \leftrightarrow 'a \rightarrow 'a \leftrightarrow 'a$
<b>RelCombine</b>	$'a \leftrightarrow 'b \rightarrow 'a \leftrightarrow 'c \rightarrow 'a \leftrightarrow ('b \times 'c)$

### 3.4 Aliases

$\%$	$\$R_{-\%}R : 'b \leftrightarrow 'a \rightarrow 'a \leftrightarrow 'c \rightarrow 'b \leftrightarrow 'c$
$\circ$	$\$R_{-\circ}R : 'a \leftrightarrow 'c \rightarrow 'b \leftrightarrow 'a \rightarrow 'b \leftrightarrow 'c$
$\sim$	$InvRel : 'b \leftrightarrow 'a \rightarrow 'a \leftrightarrow 'b$

### 3.5 Type Abbreviations

$'a \mathbb{P}$	$'a \mathbb{P}$
$'a \leftrightarrow 'b$	$'a \leftrightarrow 'b$

### 3.6 Fixity

Right Infix 240:

**R\_o\_R**      **R\_%-R**  $\triangleright$        $\triangleright$        $\leftrightarrow$        $\oplus$        $\%$        $\triangleleft$        $\triangleleft$

Right Infix 280:

**Image**

Right Infix 300:

$\mapsto$

Postfix 300:

$*$        $+$        $\sim$

### 3.7 Definitions

$\mapsto$	$\vdash \$\mapsto = \$,$
$\times$	$\vdash \forall x y \bullet (x \times y) = \{(v, w)   v \in x \wedge w \in y\}$
$\leftrightarrow$	$\vdash \forall x y \bullet x \leftrightarrow y = \mathbb{P} (x \times y)$
<b>Dom</b>	$\vdash \forall r \bullet Dom r = \{x   \exists y \bullet (x, y) \in r\}$
<b>Ran</b>	$\vdash \forall r \bullet Ran r = \{y   \exists x \bullet (x, y) \in r\}$
<b>Id</b>	$\vdash \forall s \bullet Id s = \{(x, y)   x = y \wedge x \in s\}$
<b>Graph</b>	$\vdash \forall f \bullet Graph f = \{(x, y)   y = f x\}$
$\%$	$\vdash \forall f g \bullet f \% g = g \circ f$
<b>R_%-R</b>	$\vdash \forall r s \bullet r \% s = \{(x, z)   \exists y \bullet (x, y) \in r \wedge (y, z) \in s\}$
<b>R_o_R</b>	$\vdash \forall r s \bullet r \circ s = s \% r$
$\triangleleft$	$\vdash \forall a r \bullet a \triangleleft r = \{(x, y)   x \in a \wedge (x, y) \in r\}$
$\triangleright$	$\vdash \forall a r \bullet r \triangleright a = \{(x, y)   y \in a \wedge (x, y) \in r\}$
$\triangleleft$	$\vdash \forall a r \bullet a \triangleleft r = \{(x, y)   \neg x \in a \wedge (x, y) \in r\}$
$\triangleright$	$\vdash \forall a r \bullet r \triangleright a = \{(x, y)   \neg y \in a \wedge (x, y) \in r\}$
<b>InvRel</b>	$\vdash \forall r \bullet r \sim = \{(x, y)   (y, x) \in r\}$

---

<b>Image</b>	$\vdash \forall r \bullet r \text{ Image } s = \{y \mid \exists x \bullet x \in s \wedge (x, y) \in r\}$
<b>Reflexive</b>	$\vdash \text{Reflexive} = \{r \mid \forall x \bullet (x, x) \in r\}$
<b>Symmetric</b>	$\vdash \text{Symmetric} = \{r \mid \forall x y \bullet (x, y) \in r \Rightarrow (y, x) \in r\}$
<b>Transitive</b>	$\vdash \text{Transitive}$ $= \{r \mid \forall x y z \bullet (x, y) \in r \wedge (y, z) \in r \Rightarrow (x, z) \in r\}$
<b>Injective</b>	$\vdash \text{Injective}$ $= \{r \mid \forall x y z \bullet (x, z) \in r \wedge (y, z) \in r \Rightarrow x = y\}$
<b>Surjective</b>	$\vdash \forall s \bullet \text{Surjective } s = \{r \mid s = \text{Ran } r\}$
<b>Total</b>	$\vdash \forall s \bullet \text{Total } s = \{r \mid s = \text{Dom } r\}$
<b>Functional</b>	$\vdash \text{Functional}$ $= \{r \mid \forall x w z \bullet (x, w) \in r \wedge (x, z) \in r \Rightarrow w = z\}$
$\oplus$	$\vdash \forall r \bullet r \oplus s = (\text{Dom } s \triangleleft r) \cup s$
$+$	$\vdash \forall r \bullet r^+ = \bigcap \{q \mid r \subseteq q \wedge q \in \text{Transitive}\}$
$*$	$\vdash \forall r$ $\bullet r^*$ $= \bigcap \{q \mid r \subseteq q \wedge q \in \text{Reflexive} \wedge q \in \text{Transitive}\}$
<b>RelCombine</b>	$\vdash \forall f g$ $\bullet \text{RelCombine } f g$ $= \{(x, y, z) \mid (x, y) \in f \wedge (x, z) \in g\}$

### 3.8 Theorems

#### rel\_in\_clauses

$\vdash \forall a b x x1 y z r r1 r2 s t q f$
$\bullet (x \mapsto y \in r \Leftrightarrow (x, y) \in r)$
$\wedge ((x, y) \in (a \times b) \Leftrightarrow x \in a \wedge y \in b)$
$\wedge (r \in a \Leftrightarrow b \Leftrightarrow r \subseteq (a \times b))$
$\wedge (x \in \text{Dom } r \Leftrightarrow (\exists y \bullet (x, y) \in r))$
$\wedge (y \in \text{Ran } r \Leftrightarrow (\exists x \bullet (x, y) \in r))$
$\wedge ((x, x1) \in \text{Id } a \Leftrightarrow x = x1 \wedge x \in a)$
$\wedge ((x, y) \in \text{Graph } f \Leftrightarrow y = f x)$
$\wedge ((x, z) \in r \circ s$
$\Leftrightarrow (\exists y \bullet (x, y) \in r \wedge (y, z) \in s))$
$\wedge ((x, z) \in s \circ r \Leftrightarrow (x, z) \in r \circ s)$
$\wedge ((x, y) \in a \triangleleft r \Leftrightarrow x \in a \wedge (x, y) \in r)$
$\wedge ((x, y) \in r \triangleright b \Leftrightarrow y \in b \wedge (x, y) \in r)$
$\wedge ((x, y) \in a \triangleleft r \Leftrightarrow \neg x \in a \wedge (x, y) \in r)$
$\wedge ((x, y) \in r \triangleright b \Leftrightarrow \neg y \in b \wedge (x, y) \in r)$
$\wedge ((y, x) \in r \sim \Leftrightarrow (x, y) \in r)$
$\wedge (y \in r \text{ Image } a \Leftrightarrow (\exists x \bullet x \in a \wedge (x, y) \in r))$
$\wedge (q \in \text{Reflexive} \Leftrightarrow (\forall x \bullet (x, x) \in q))$
$\wedge (q \in \text{Symmetric}$
$\Leftrightarrow (\forall x1 x2 \bullet (x1, x2) \in q \Rightarrow (x2, x1) \in q))$
$\wedge (q \in \text{Transitive}$
$\Leftrightarrow (\forall x1 x2 x3$
$\bullet (x1, x2) \in q \wedge (x2, x3) \in q \Rightarrow (x1, x3) \in q))$
$\wedge (r \in \text{Injective}$
$\Leftrightarrow (\forall x1 x2 y$

$$\begin{aligned}
& \bullet (x1, y) \in r \wedge (x2, y) \in r \Rightarrow x1 = x2)) \\
& \wedge (r \in \text{Surjective } b \\
& \Leftrightarrow (\forall y \bullet y \in b \Leftrightarrow (\exists x \bullet (x, y) \in r))) \\
& \wedge (r \in \text{Total } a \\
& \Leftrightarrow (\forall x \bullet x \in a \Leftrightarrow (\exists y \bullet (x, y) \in r))) \\
& \wedge (r \in \text{Functional} \\
& \Leftrightarrow (\forall x \ y1 \ y2 \\
& \bullet (x, y1) \in r \wedge (x, y2) \in r \Rightarrow y1 = y2)) \\
& \wedge ((x, y) \in r1 \oplus r2 \\
& \Leftrightarrow (x, y) \in (\text{Dom } r2 \triangleleft r1) \cup r2) \\
& \wedge ((x, y, z) \in \text{RelCombine } r \ t \\
& \Leftrightarrow (x, y) \in r \wedge (x, z) \in t)
\end{aligned}$$

**bin\_rel\_ext\_clauses**

$$\begin{aligned}
& \vdash \forall r1 \ r2 \\
& \bullet (r1 \subset r2 \\
& \Leftrightarrow (\forall x \ y \bullet (x, y) \in r1 \Rightarrow (x, y) \in r2) \\
& \quad \wedge (\exists x \ y \bullet \neg (x, y) \in r1 \wedge (x, y) \in r2)) \\
& \wedge (r1 \subseteq r2 \Leftrightarrow (\forall x \ y \bullet (x, y) \in r1 \Rightarrow (x, y) \in r2)) \\
& \wedge (r1 = r2 \Leftrightarrow (\forall x \ y \bullet (x, y) \in r1 \Leftrightarrow (x, y) \in r2))
\end{aligned}$$

**inv\_rel\_thm**

$$\begin{aligned}
& \vdash \forall f \ a \ b \\
& \bullet (f \sim \in \text{Functional} \Leftrightarrow f \in \text{Injective}) \\
& \wedge (f \sim \in \text{Injective} \Leftrightarrow f \in \text{Functional}) \\
& \wedge (f \sim \in \text{Surjective } a \Leftrightarrow f \in \text{Total } a) \\
& \wedge (f \sim \in \text{Total } b \Leftrightarrow f \in \text{Surjective } b)
\end{aligned}$$

**bin\_rel\_∅\_universe\_thm**

$$\begin{aligned}
& \vdash \forall f \ g \ r0 \ r1 \ a \ b \\
& \bullet (\text{Dom } r0 = \{\} \Leftrightarrow r0 = \{\}) \\
& \wedge (\text{Ran } r0 = \{\} \Leftrightarrow r0 = \{\}) \\
& \wedge \text{Dom } \{\} = \{\} \\
& \wedge \text{Ran } \{\} = \{\} \\
& \wedge \text{Dom } \text{Universe} = \text{Universe} \\
& \wedge \text{Ran } \text{Universe} = \text{Universe} \\
& \wedge (\text{Id } r0 = \{\} \Leftrightarrow r0 = \{\}) \\
& \wedge \text{Id } \{\} = \{\} \\
& \wedge (r0 \sim = \{\} \Leftrightarrow r0 = \{\}) \\
& \wedge \{\} \sim = \{\} \\
& \wedge \text{Universe} \sim = \text{Universe} \\
& \wedge r0 \ ; \ \{\} = \{\} \\
& \wedge \{\} \ ; \ r0 = \{\} \\
& \wedge (\{\} = r0 \ ; \ r1 \Leftrightarrow \text{Ran } r0 \cap \text{Dom } r1 = \{\}) \\
& \wedge (r0 \ ; \ r1 = \{\} \Leftrightarrow \text{Ran } r0 \cap \text{Dom } r1 = \{\}) \\
& \wedge \text{RelCombine } r0 \ \{\} = \{\} \\
& \wedge \text{RelCombine } \{\} \ r0 = \{\} \\
& \wedge r0 \ \text{Image } \{\} = \{\} \\
& \wedge \{\} \ \text{Image } a = \{\} \\
& \wedge f \oplus \{\} = f \\
& \wedge \{\} \oplus f = f \\
& \wedge (f \oplus g = \{\} \Leftrightarrow f = \{\} \wedge g = \{\})
\end{aligned}$$

$$\begin{aligned}
& \wedge (f = \{\} \wedge g = \{\} \Rightarrow f \oplus g = \{\}) \\
& \wedge (f \oplus g = \{\} \Leftrightarrow f = \{\} \wedge g = \{\}) \\
& \wedge f \oplus \text{Universe} = \text{Universe} \\
& \wedge \text{Universe} \triangleleft r0 = \{\} \\
& \wedge a \triangleleft \{\} = \{\} \\
& \wedge \{\} \triangleleft r0 = r0 \\
& \wedge \text{Universe} \triangleleft r0 = r0 \\
& \wedge a \triangleleft \{\} = \{\} \\
& \wedge \{\} \triangleleft r0 = \{\} \\
& \wedge r0 \triangleright \text{Universe} = \{\} \\
& \wedge \{\} \triangleright b = \{\} \\
& \wedge r0 \triangleright \{\} = r0 \\
& \wedge r0 \triangleright \text{Universe} = r0 \\
& \wedge \{\} \triangleright b = \{\} \\
& \wedge r0 \triangleright \{\} = \{\}
\end{aligned}$$

**bin\_rel\_insert\_thm**

$$\begin{aligned}
& \vdash \forall a b r x xy y \\
& \bullet \text{Dom (Insert (x, y) r) = Insert x (Dom r)} \\
& \quad \wedge \text{Dom (Insert xy r) = Insert (Fst xy) (Dom r)} \\
& \quad \wedge \text{Ran (Insert (x, y) r) = Insert y (Ran r)} \\
& \quad \wedge \text{Ran (Insert xy r) = Insert (Snd xy) (Ran r)} \\
& \quad \wedge \text{Id (Insert x a) = Insert (x, x) (Id a)} \\
& \quad \wedge \text{Insert (x, y) r Image a} \\
& \quad \quad = r \text{ Image a} \cup (\text{if } x \in a \text{ then } \{y\} \text{ else } \{\}) \\
& \quad \wedge \text{Insert (x, y) r } \triangleright b \\
& \quad \quad = (r \triangleright b) \cup (\text{if } \neg y \in b \text{ then } \{(x, y)\} \text{ else } \{\}) \\
& \quad \wedge \text{Insert (x, y) r } \triangleright b \\
& \quad \quad = (r \triangleright b) \cup (\text{if } y \in b \text{ then } \{(x, y)\} \text{ else } \{\}) \\
& \quad \wedge a \triangleleft \text{Insert (x, y) r} \\
& \quad \quad = (a \triangleleft r) \cup (\text{if } \neg x \in a \text{ then } \{(x, y)\} \text{ else } \{\}) \\
& \quad \wedge a \triangleleft \text{Insert (x, y) r} \\
& \quad \quad = (a \triangleleft r) \cup (\text{if } x \in a \text{ then } \{(x, y)\} \text{ else } \{\}) \\
& \quad \wedge \text{Insert (x, y) r } \sim = \text{Insert (y, x) (r } \sim)
\end{aligned}$$

**4 THE THEORY cache'fef****4.1 Parents***basic\_hol***4.2 Children***fef004***4.3 Notes**

This theory is a cache theory; its contents have not been listed.

## 5 THE THEORY char

### 5.1 Parents

*list*

### 5.2 Children

*basic\_hol*

### 5.3 Constants

**IsCharRep**     *Worth*  $\rightarrow$  *Bool*  
**AbsChar**        *Worth*  $\rightarrow$  *Char*  
**RepChar**        *Char*  $\rightarrow$  *Worth*

### 5.4 Types

**Char**

### 5.5 Type Abbreviations

**STRING**        *Text*

### 5.6 Definitions

**IsCharRep**

**is\_char\_rep\_def**

$\vdash \forall x \bullet \text{IsCharRep } x \Leftrightarrow x < 256$

**CHAR**

**char\_def**         $\vdash \exists f \bullet \text{TypeDefn IsCharRep } f$

**AbsChar**

**RepChar**

**abs\_char\_rep\_char\_def**

$\vdash (\forall a \bullet \text{AbsChar } (\text{RepChar } a) = a)$   
 $\wedge (\forall r \bullet \text{IsCharRep } r \Leftrightarrow \text{RepChar } (\text{AbsChar } r) = r)$

## 6 THE THEORY combin

### 6.1 Parents

*basic\_hol*

### 6.2 Children

*sum*

### 6.3 Constants

$\$o$	$( 'b \rightarrow 'c ) \rightarrow ( 'a \rightarrow 'b ) \rightarrow 'a \rightarrow 'c$
<b>CombS</b>	$( 'a \rightarrow 'b \rightarrow 'c ) \rightarrow ( 'a \rightarrow 'b ) \rightarrow 'a \rightarrow 'c$
<b>CombK</b>	$'a \rightarrow 'b \rightarrow 'a$
<b>CombI</b>	$'a \rightarrow 'a$

### 6.4 Fixity

*Right Infix 300:*

o

### 6.5 Definitions

o

<b>o_def</b>	$\vdash \forall f g x \bullet (f \circ g) x = f (g x)$
<b>CombS</b>	
<b>comb_s_def</b>	$\vdash \forall f g x \bullet CombS f g x = f x (g x)$
<b>CombK</b>	
<b>comb_k_def</b>	$\vdash \forall x y \bullet CombK x y = x$
<b>CombI</b>	
<b>comb_i_def</b>	$\vdash \forall x \bullet CombI x = x$

### 6.6 Theorems

<b>s_k_thm</b>	$\vdash \forall x \bullet CombS CombK x = CombI$
<b>o_assoc_thm</b>	$\vdash \forall f g h \bullet f \circ g \circ h = (f \circ g) \circ h$
<b>o_i_thm</b>	$\vdash \forall f \bullet CombI \circ f = f \wedge f \circ CombI = f$

## 7 THE THEORY dyadic

### 7.1 Parents

$\mathbb{Z}$

### 7.2 Children

$\mathbb{R}$

### 7.3 Constants

$\$^{\wedge}$	$\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$
<b><math>\\$dy\_times</math></b>	$DYADIC \rightarrow DYADIC \rightarrow DYADIC$
<b><math>dy\_one</math></b>	$DYADIC$
<b><math>\\$dy\_exp</math></b>	$DYADIC \rightarrow \mathbb{N} \rightarrow DYADIC$
<b><math>\\$dy\_less</math></b>	$DYADIC \rightarrow DYADIC \rightarrow BOOL$

## 7.4 Type Abbreviations

**DYADIC**      *DYADIC*

## 7.5 Fixity

*Right Infix 210:*

**dy\_less**

*Right Infix 310:*

**dy\_exp      dy\_times**

*Right Infix 320:*

**^**

## 7.6 Definitions

**^**       $\vdash (\forall i \bullet i \wedge 0 = 1) \wedge (\forall i \ m \bullet i \wedge (m + 1) = i * i \wedge m)$

**dy\_times**       $\vdash \forall (m, i) (n, j)$   
                   •  $(m, i) \text{ dy\_times } (n, j) = (2 * m * n + m + n, i + j)$

**dy\_one**       $\vdash \text{dy\_one} = (0, \text{NZ } 0)$

**dy\_exp**       $\vdash (\forall x \bullet x \text{ dy\_exp } 0 = \text{dy\_one})$   
                    $\wedge (\forall x \ m \bullet x \text{ dy\_exp } (m + 1) = x \text{ dy\_times } x \text{ dy\_exp } m)$

**dy\_less**       $\vdash \forall (m, i) (n, j)$   
                   •  $(m, i) \text{ dy\_less } (n, j)$   
                    $\Leftrightarrow (\exists a \ b$   
                   •  $\text{NZ } a + i = \text{NZ } b + j$   
                    $\wedge 2 \wedge a * (2 * m + 1) < 2 \wedge b * (2 * n + 1))$

## 7.7 Theorems

**dy\_less\_irrefl\_thm**

$\vdash \forall x \bullet \neg x \text{ dy\_less } x$

**dy\_less\_antisym\_thm**

$\vdash \forall x \ y \bullet \neg (x \text{ dy\_less } y \wedge y \text{ dy\_less } x)$

**dy\_less\_trans\_thm**

$\vdash \forall x \ y \ z \bullet x \text{ dy\_less } y \wedge y \text{ dy\_less } z \Rightarrow x \text{ dy\_less } z$

**dy\_less\_trich\_thm**

$\vdash \forall x \ y \bullet \neg x = y \Rightarrow x \text{ dy\_less } y \vee y \text{ dy\_less } x$

**dy\_times\_comm\_thm**

$\vdash \forall x \ y \bullet x \text{ dy\_times } y = y \text{ dy\_times } x$

**dy\_times\_assoc\_thm**

$\vdash \forall x \ y \ z$   
                   •  $(x \text{ dy\_times } y) \text{ dy\_times } z = x \text{ dy\_times } y \text{ dy\_times } z$

**dy\_times\_order\_thm**

$\vdash \forall u \ x \ y$   
                   •  $x \text{ dy\_times } u = u \text{ dy\_times } x$   
                    $\wedge (u \text{ dy\_times } x) \text{ dy\_times } y$   
                    $= u \text{ dy\_times } x \text{ dy\_times } y$

$$\begin{aligned} & \wedge x \text{ dy\_times } u \text{ dy\_times } y \\ & = u \text{ dy\_times } x \text{ dy\_times } y \end{aligned}$$

**dy\_times\_unit\_thm**

$$\vdash \forall x \bullet x \text{ dy\_times } \text{dy\_one} = x$$

**dy\_times\_unit\_clauses**

$$\vdash \forall x \bullet x \text{ dy\_times } \text{dy\_one} = x \wedge \text{dy\_one} \text{ dy\_times } x = x$$

**dy\_exp\_clauses**

$$\vdash \forall x \ m \ n$$

- $x \text{ dy\_exp } (m + n) = (x \text{ dy\_exp } m) \text{ dy\_times } x \text{ dy\_exp } n$
- $x \text{ dy\_exp } 0 = \text{dy\_one}$
- $x \text{ dy\_exp } 1 = x$
- $x \text{ dy\_exp } 2 = x \text{ dy\_times } x$
- $x \text{ dy\_exp } 3 = x \text{ dy\_times } x \text{ dy\_times } x$

**dy\_times\_mono\_thm**

$$\vdash \forall x \ y \ z$$

- $y \text{ dy\_less } z \Rightarrow x \text{ dy\_times } y \text{ dy\_less } x \text{ dy\_times } z$

**dy\_times\_mono\_thm1**

$$\vdash \forall x \ y \ z$$

- $y \text{ dy\_less } z \Rightarrow y \text{ dy\_times } x \text{ dy\_less } z \text{ dy\_times } x$

**dy\_times\_mono\_thm2**

$$\vdash \forall x \ y \ v \ w$$

- $x \text{ dy\_less } y \wedge v \text{ dy\_less } w$   
 $\Rightarrow x \text{ dy\_times } v \text{ dy\_less } y \text{ dy\_times } w$

**dy\_times\_mono\_⇔\_thm**

$$\vdash \forall x \ y \ z$$

- $x \text{ dy\_times } y \text{ dy\_less } x \text{ dy\_times } z \Leftrightarrow y \text{ dy\_less } z$

**dy\_arch\_thm**  $\vdash \forall x \ y \bullet \text{dy\_one} \text{ dy\_less } x \Rightarrow (\exists t \bullet y \text{ dy\_less } x \text{ dy\_exp } t)$ **dy\_balance\_thm1**

$$\vdash \forall x \bullet \exists y \bullet \text{dy\_one} \text{ dy\_less } x \text{ dy\_times } y$$

**dy\_balance\_thm2**

$$\vdash \forall x \bullet \exists y \bullet x \text{ dy\_times } y \text{ dy\_less } \text{dy\_one}$$

**dy\_right\_dense\_thm**

$$\vdash \forall x \ y$$

- $x \text{ dy\_less } y$   
 $\Rightarrow (\exists z$
- $x \text{ dy\_less } x \text{ dy\_times } z$   
 $\wedge x \text{ dy\_times } z \text{ dy\_less } y)$

**dy\_less\_dense\_thm**

$$\vdash \forall x \ y \bullet x \text{ dy\_less } y \Rightarrow (\exists z \bullet x \text{ dy\_less } z \wedge z \text{ dy\_less } y)$$

**dy\_left\_dense\_thm**

$$\vdash \forall x \ y$$

- $x \text{ dy\_less } y$   
 $\Rightarrow (\exists z$
- $x \text{ dy\_less } y \text{ dy\_times } z$   
 $\wedge y \text{ dy\_times } z \text{ dy\_less } y)$

## 8 THE THEORY fef003

### 8.1 Parents

*wrk049*

### 8.2 Children

*fef040 fef004*

### 8.3 Constants

<b>\$glb</b>	$Class \rightarrow Class \rightarrow Class$
<b>\$lub</b>	$Class \rightarrow Class \rightarrow Class$
<b>lattice_top</b>	$Class$
<b>lattice_bottom</b>	$Class$
<b>\$dominates</b>	$Class \rightarrow Class \rightarrow BOOL$
<b>same_ins</b>	$Class$ $\rightarrow ('QUERY \times Class) LIST \leftrightarrow ('QUERY \times Class) LIST$
<b>same_outs</b>	$Class \rightarrow (Class \times 'DATA) LIST \leftrightarrow (Class \times 'DATA) LIST$
<b>secure</b>	$('QUERY, 'DATA) BEHAVIOURS \mathbb{P}$

### 8.4 Types

**Class**

### 8.5 Type Abbreviations

$('QUERY, 'DATA) \mathbf{BEHAVIOURS}$   
 $('QUERY, 'DATA) BEHAVIOURS$

### 8.6 Fixity

*Right Infix 150:*

**dominates**

*Right Infix 250:*

**glb    lub**

## 8.7 Definitions

**dominates**

**lattice\_bottom**

**lattice\_top**

**lub**

**glb**

$\vdash \text{ConstSpec}$

( $\lambda$

$(\text{dominates}', \text{lattice\_bottom}', \text{lattice\_top}',$   
 $\text{lub}', \text{glb}')$

•  $\forall x y z$

•  $\text{dominates}' x x$

$\wedge (\text{dominates}' x y \wedge \text{dominates}' y x \Rightarrow x = y)$

$\wedge (\text{dominates}' x y \wedge \text{dominates}' y z$   
 $\Rightarrow \text{dominates}' x z)$

$\wedge (\forall x \bullet \text{dominates}' x \text{lattice\_bottom}')$

$\wedge (\forall x \bullet \text{dominates}' \text{lattice\_top}' x)$

$\wedge (\forall x y z$

•  $\text{dominates}' (\text{lub}' x y) x$

$\wedge \text{dominates}' (\text{lub}' x y) y$

$\wedge (\text{dominates}' z x \wedge \text{dominates}' z y$   
 $\Rightarrow \text{dominates}' z (\text{lub}' x y))$

$\wedge (\forall x y z$

•  $\text{dominates}' x (\text{glb}' x y)$

$\wedge \text{dominates}' y (\text{glb}' x y)$

$\wedge (\text{dominates}' x z \wedge \text{dominates}' y z$   
 $\Rightarrow \text{dominates}' (\text{glb}' x y) z))$

$(\$dominates, \text{lattice\_bottom}, \text{lattice\_top}, \$lub,$   
 $\$glb)$

**same\_ins**

$\vdash \forall \text{clear } si1 \ si2$

•  $(si1, si2) \in \text{same\_ins clear}$

$\Leftrightarrow (\text{let } v = \{(q, c) | \text{clear dominates } c\}$   
 $\text{in } si1 \upharpoonright v = si2 \upharpoonright v)$

**same\_outs**

$\vdash \forall \text{clear } so1 \ so2$

•  $(so1, so2) \in \text{same\_outs clear}$

$\Leftrightarrow (\text{let } v = \{(c, d) | \text{clear dominates } c\}$   
 $\text{in } so1 \upharpoonright v = so2 \upharpoonright v)$

**secure**

$\vdash \forall bm$

•  $bm \in \text{secure}$

$\Leftrightarrow (\forall \text{clear } si1 \ si2$

•  $(si1, si2) \in \text{same\_ins clear}$

$\Rightarrow (bm \ si1, bm \ si2) \in \text{same\_outs clear})$

## 9 THE THEORY fef004

### 9.1 Parents

*cache'fef*

*fef003*

## 9.2 Children

*fef005*

## 9.3 Constants

<b>Num</b>	<i>Worth</i> $\mathbb{P}$
<b>Ide</b>	<i>Ide</i> $\mathbb{P}$
<b>I_exponent</b>	<i>Int</i> $\rightarrow$ <i>Worth</i>
<b>I_mantissa</b>	<i>Int</i> $\rightarrow$ <i>Worth</i>
<b>MkInt</b>	<i>Worth</i> $\rightarrow$ <i>Worth</i> $\rightarrow$ <i>Int</i>
<b>F_exponent</b>	<i>Float</i> $\rightarrow$ <i>Worth</i>
<b>F_mantissa</b>	<i>Float</i> $\rightarrow$ <i>Worth</i>
<b>MkFloat</b>	<i>Worth</i> $\rightarrow$ <i>Worth</i> $\rightarrow$ <i>Float</i>
<b>accessDenied</b>	<i>Worth</i>
<b>rowClassTooLow</b>	<i>Worth</i>
<b>fieldClassOutOfRange</b>	<i>Worth</i>
<b>noNulls</b>	<i>Worth</i>
<b>nonUniqueValues</b>	<i>Worth</i>
<b>nonUniformValues</b>	<i>Worth</i>
<b>ambiguousHaving</b>	<i>Worth</i>
<b>ambiguousEvaluate</b>	<i>Worth</i>
<b>noSuchDirectory</b>	<i>Worth</i>
<b>noSuchTable</b>	<i>Worth</i>
<b>classChange</b>	<i>Worth</i>
<b>downGrade</b>	<i>Worth</i>
<b>underClassified</b>	<i>Worth</i>
<b>mayNotBeComplete</b>	<i>Worth</i>
<b>ambiguousUpdate</b>	<i>Worth</i>
<b>ambiguousColumn</b>	<i>Worth</i>
<b>noSuchColumn</b>	<i>Worth</i>
<b>nullValue</b>	<i>Worth</i>
<b>wrongType</b>	<i>Worth</i>
<b>tooTall</b>	<i>Worth</i>
<b>tooWide</b>	<i>Worth</i>
<b>notCleared</b>	<i>Worth</i>
<b>error</b>	<i>Worth</i>

---

<b>priceless</b>	<i>Worth</i>
<b>sterling</b>	<i>Worth</i>
<b>dinary</b>	<i>Worth</i>
<b>worthless</b>	<i>Worth</i>
<b>ExceptionVal</b>	<i>Val</i>
<b>ClassVal</b>	<i>Class</i> $\rightarrow$ <i>Val</i>
<b>CodeVal</b>	<i>Code</i> $\rightarrow$ <i>Val</i>
<b>IntervalVal</b>	<i>Interval</i> $\rightarrow$ <i>Val</i>
<b>TimeVal</b>	<i>Time</i> $\rightarrow$ <i>Val</i>
<b>FloatVal</b>	<i>Float</i> $\rightarrow$ <i>Val</i>
<b>IntVal</b>	<i>Int</i> $\rightarrow$ <i>Val</i>
<b>StringVal</b>	<i>Ide</i> $\rightarrow$ <i>Val</i>
<b>BoolVal</b>	<i>Bool</i> $\rightarrow$ <i>Val</i>
<b>VoidVal</b>	<i>Val</i>
<b>VI_val</b>	<i>ValuedItem</i> $\rightarrow$ <i>Val</i>
<b>VL_worth</b>	<i>ValuedItem</i> $\rightarrow$ <i>Worth</i>
<b>MkValuedItem</b>	<i>Worth</i> $\rightarrow$ <i>Val</i> $\rightarrow$ <i>ValuedItem</i>
<b>NullItemItem</b>	<i>NullItem</i> $\rightarrow$ <i>Item</i>
<b>ValuedItemItem</b>	<i>ValuedItem</i> $\rightarrow$ <i>Item</i>
<b>Dat_item</b>	<i>Data</i> $\rightarrow$ <i>Item</i>
<b>Dat_class</b>	<i>Data</i> $\rightarrow$ <i>Class</i>
<b>MkData</b>	<i>Class</i> $\rightarrow$ <i>Item</i> $\rightarrow$ <i>Data</i>
<b>DataS</b>	<i>Data</i> $\mathbb{P}$
<b>DataUpdate</b>	<i>Data</i> $\rightarrow$ <i>Update</i>
<b>ClassUpdate</b>	<i>Class</i> $\rightarrow$ <i>Update</i>
<b>ItemUpdate</b>	<i>Item</i> $\rightarrow$ <i>Update</i>
<b>isData</b>	<i>Update</i> $\rightarrow$ <i>Bool</i>
<b>isClass</b>	<i>Update</i> $\rightarrow$ <i>Bool</i>
<b>isItem</b>	<i>Update</i> $\rightarrow$ <i>Bool</i>
<b>destData</b>	<i>Update</i> $\rightarrow$ <i>Data</i>
<b>destClass</b>	<i>Update</i> $\rightarrow$ <i>Class</i>
<b>destItem</b>	<i>Update</i> $\rightarrow$ <i>Item</i>
<b>CS_max</b>	<i>ColSpec</i> $\rightarrow$ <i>Class</i>
<b>CS_min</b>	<i>ColSpec</i> $\rightarrow$ <i>Class</i>
<b>CS_consGroup</b>	<i>ColSpec</i> $\rightarrow$ <i>Worth</i>
<b>CS_default</b>	<i>ColSpec</i> $\rightarrow$ <i>Data</i>
<b>CS_nullType</b>	<i>ColSpec</i> $\rightarrow$ <i>Bool</i>
<b>CS_sterlingType</b>	<i>ColSpec</i> $\rightarrow$ <i>Type</i>
<b>CS_dinaryType</b>	<i>ColSpec</i> $\rightarrow$ <i>Type</i>
<b>CS_posn</b>	<i>ColSpec</i> $\rightarrow$ <i>Worth</i>
<b>CS_ide</b>	<i>ColSpec</i> $\rightarrow$ <i>Ide</i>
<b>MkColSpec</b>	<i>Ide</i> $\rightarrow$ <i>Worth</i> $\rightarrow$ <i>Type</i> $\rightarrow$ <i>Type</i>

---

---

	$\rightarrow Bool$
	$\rightarrow Data$
	$\rightarrow Worth$
	$\rightarrow Class$
	$\rightarrow Class$
	$\rightarrow ColSpec$
<b>R_group</b>	Reference $\rightarrow Worth$
<b>R_table</b>	Reference $\rightarrow Tab$
<b>MkReference</b>	Tab $\rightarrow Worth \rightarrow Reference$
<b>CC_referential</b>	
	ColCon $\rightarrow Reference LIST$
<b>CC_secondary</b>	ColCon $\rightarrow Bool$
<b>CC_primary</b>	ColCon $\rightarrow Bool$
<b>CC_classLimited</b>	
	ColCon $\rightarrow Bool$
<b>CC_unique</b>	ColCon $\rightarrow Bool$
<b>CC_uniform</b>	ColCon $\rightarrow Bool$
<b>CC_exist</b>	ColCon $\rightarrow Class$
<b>MkColCon</b>	Class
	$\rightarrow Bool$
	$\rightarrow Bool$
	$\rightarrow Bool$
	$\rightarrow Bool$
	$\rightarrow Bool$
	$\rightarrow Reference LIST$
	$\rightarrow ColCon$
<b>ColConS</b>	ColCon $\mathbb{P}$
<b>R_data</b>	Row $\rightarrow Worth \leftrightarrow Data$
<b>R_exist</b>	Row $\rightarrow Class$
<b>MkRow</b>	Class $\rightarrow Worth \leftrightarrow Data \rightarrow Row$
<b>RowS</b>	Row $\mathbb{P}$
<b>TS_rows</b>	TableSpec $\rightarrow Row LIST$
<b>TS_cons</b>	TableSpec $\rightarrow Worth \leftrightarrow ColCon$
<b>TS_colspecs</b>	TableSpec $\rightarrow ColSpec \mathbb{P}$
<b>TS_maxRow</b>	TableSpec $\rightarrow Class$
<b>TS_class</b>	TableSpec $\rightarrow Class$
<b>MkTableSpec</b>	Class
	$\rightarrow Class$
	$\rightarrow ColSpec \mathbb{P}$
	$\rightarrow Worth \leftrightarrow ColCon$
	$\rightarrow Row LIST$
	$\rightarrow TableSpec$
<b>TableSpecS</b>	TableSpec $\mathbb{P}$
<b>Dir_class</b>	Directory $\rightarrow Class$
<b>Dir_exist</b>	Directory $\rightarrow Class$
<b>Dir_tables</b>	Directory $\rightarrow Ide \leftrightarrow TableSpec$
<b>MkDirectory</b>	Ide $\leftrightarrow TableSpec \rightarrow Class \rightarrow Class \rightarrow Directory$
<b>DirectoryS</b>	Directory $\mathbb{P}$

---

---

<b>IdeL</b>	<i>Tab</i> $\mathbb{P}$
<b>StatesS</b>	<i>StateR</i> $\mathbb{P}$
<b>isState</b>	<i>StateR</i> $\rightarrow$ <i>Bool</i>
<b>isError</b>	<i>'a</i> + <i>'Error</i> $\rightarrow$ <i>Bool</i>
<b>isVal</b>	<i>'a</i> + <i>'Error</i> $\rightarrow$ <i>Bool</i>
<b>destError</b>	<i>'a</i> + <i>'Error</i> $\rightarrow$ <i>'Error</i>
<b>destVal</b>	<i>'a</i> + <i>'Error</i> $\rightarrow$ <i>'a</i>
<b>giveError</b>	<i>'Error</i> $\rightarrow$ <i>'a</i> + <i>'Error</i>
<b>giveVal</b>	<i>'a</i> $\rightarrow$ <i>'a</i> + <i>'Error</i>
<b>SelectEffect</b>	<i>Select</i> $\rightarrow$ <i>Effect</i>
<b>UpdateEffect</b>	<i>UpDate</i> $\rightarrow$ <i>Effect</i>
<b>DeleteEffect</b>	<i>Delete</i> $\rightarrow$ <i>Effect</i>
<b>InsertEffect</b>	<i>Insert</i> $\rightarrow$ <i>Effect</i>
<b>isSelect</b>	<i>Effect</i> $\rightarrow$ <i>Bool</i>
<b>isUpdate</b>	<i>Effect</i> $\rightarrow$ <i>Bool</i>
<b>isDelete</b>	<i>Effect</i> $\rightarrow$ <i>Bool</i>
<b>isInsert</b>	<i>Effect</i> $\rightarrow$ <i>Bool</i>
<b>destSelect</b>	<i>Effect</i> $\rightarrow$ <i>Select</i>
<b>destUpdate</b>	<i>Effect</i> $\rightarrow$ <i>UpDate</i>
<b>destDelete</b>	<i>Effect</i> $\rightarrow$ <i>Delete</i>
<b>destInsert</b>	<i>Effect</i> $\rightarrow$ <i>Insert</i>
<b>tabFromEffect</b>	<i>Effect</i> $\rightarrow$ <i>Tab</i>

#### 9.4 Aliases

<b>null</b>	<i>One</i> : <i>NullItem</i>
<b>monolean</b>	<i>One</i> : <i>NullItem</i>
<b>boolean</b>	<i>One</i> : <i>NullItem</i>
<b>chars</b>	<i>One</i> : <i>NullItem</i>
<b>integer</b>	<i>One</i> : <i>NullItem</i>
<b>floating</b>	<i>One</i> : <i>NullItem</i>
<b>time</b>	<i>One</i> : <i>NullItem</i>
<b>interval</b>	<i>One</i> : <i>NullItem</i>
<b>class</b>	<i>One</i> : <i>NullItem</i>

#### 9.5 Types

**Int**  
**Float**  
**Time**  
**Interval**  
**Code**  
**ValuedItem**  
**Data**  
**ColSpec**  
**Reference**  
**ColCon**

Row  
TableSpec  
Directory  
State

## 9.6 Type Abbreviations

<b>Char</b>	<i>Char</i>
<b>String</b>	<i>Ide</i>
<b>Num</b>	<i>Worth</i>
<b>Bool</b>	<i>Bool</i>
<b>Ide</b>	<i>Ide</i>
<b>Error</b>	<i>Worth</i>
<b>Worth</b>	<i>Worth</i>
<b>Val</b>	<i>Val</i>
<b>NullItem</b>	<i>NullItem</i>
<b>Item</b>	<i>Item</i>
<b>Update</b>	<i>Update</i>
<b>Type</b>	<i>Type</i>
<b>Errors</b>	<i>Errors</i>
<b>Tab</b>	<i>Tab</i>
<b>StateR</b>	<i>StateR</i>
<b>Insert</b>	<i>Insert</i>
<b>Delete</b>	<i>Delete</i>
<b>UpDate</b>	<i>UpDate</i>
<b>Select</b>	<i>Select</i>
<b>Effect</b>	<i>Effect</i>

## 9.7 Definitions

<b>Num</b>	$\vdash \text{Num} = \text{Universe}$
<b>Ide</b>	$\vdash \text{Ide} = \text{Universe}$
<b>Int</b>	$\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet T) f$
<b>MkInt</b>	
<b>I_mantissa</b>	
<b>I_exponent</b>	$\vdash \forall t \ x1 \ x2$ <ul style="list-style-type: none"> <li>• <math>I\_mantissa \ (\text{MkInt } x1 \ x2) = x1</math></li> <li><math>\wedge I\_exponent \ (\text{MkInt } x1 \ x2) = x2</math></li> <li><math>\wedge \text{MkInt } (I\_mantissa \ t) \ (I\_exponent \ t) = t</math></li> </ul>
<b>Float</b>	$\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet T) f$
<b>MkFloat</b>	
<b>F_mantissa</b>	
<b>F_exponent</b>	$\vdash \forall t \ x1 \ x2$ <ul style="list-style-type: none"> <li>• <math>F\_mantissa \ (\text{MkFloat } x1 \ x2) = x1</math></li> <li><math>\wedge F\_exponent \ (\text{MkFloat } x1 \ x2) = x2</math></li> <li><math>\wedge \text{MkFloat } (F\_mantissa \ t) \ (F\_exponent \ t) = t</math></li> </ul>
<b>error</b>	
<b>notCleared</b>	

**tooWide**  
**tooTall**  
**wrongType**  
**nullValue**  
**noSuchColumn**  
**ambiguousColumn**  
**ambiguousUpdate**  
**mayNotBeComplete**  
**underClassified**  
**downGrade**  
**classChange**  
**noSuchTable**  
**noSuchDirectory**  
**ambiguousEvaluate**  
**ambiguousHaving**  
**nonUniformValues**  
**nonUniqueValues**  
**noNulls**  
**fieldClassOutOfRange**  
**rowClassTooLow**  
**accessDenied**     $\vdash$  *error* = 1  
                           $\wedge$  *notCleared* = 2  
                           $\wedge$  *tooWide* = 3  
                           $\wedge$  *tooTall* = 4  
                           $\wedge$  *wrongType* = 5  
                           $\wedge$  *nullValue* = 6  
                           $\wedge$  *noSuchColumn* = 7  
                           $\wedge$  *ambiguousColumn* = 8  
                           $\wedge$  *ambiguousUpdate* = 9  
                           $\wedge$  *mayNotBeComplete* = 10  
                           $\wedge$  *underClassified* = 11  
                           $\wedge$  *downGrade* = 12  
                           $\wedge$  *classChange* = 13  
                           $\wedge$  *noSuchTable* = 14  
                           $\wedge$  *noSuchDirectory* = 15  
                           $\wedge$  *ambiguousEvaluate* = 16  
                           $\wedge$  *ambiguousHaving* = 17  
                           $\wedge$  *nonUniformValues* = 18  
                           $\wedge$  *nonUniqueValues* = 19  
                           $\wedge$  *noNulls* = 20  
                           $\wedge$  *fieldClassOutOfRange* = 21  
                           $\wedge$  *rowClassTooLow* = 22  
                           $\wedge$  *accessDenied* = 23

**worthless**  
**dinary**  
**sterling**  
**priceless**         $\vdash$  *worthless* = 1  
                           $\wedge$  *dinary* = 2

$\wedge \text{sterling} = 3$  $\wedge \text{priceless} = 4$ **VoidVal****BoolVal****StringVal****IntVal****FloatVal****TimeVal****IntervalVal****CodeVal****ClassVal****ExceptionVal**  $\vdash \forall b s i f t n c cl$ •  $\text{VoidVal} = \text{InL class}$  $\wedge \text{BoolVal } b = \text{InR (InL } b)$  $\wedge \text{StringVal } s = \text{InR (InR (InL } s))$  $\wedge \text{IntVal } i = \text{InR (InR (InR (InL } i))$  $\wedge \text{FloatVal } f = \text{InR (InR (InR (InR (InL } f))$  $\wedge \text{TimeVal } t = \text{InR (InR (InR (InR (InR (InL } t))$  $\wedge \text{IntervalVal } n$  $= \text{InR (InR (InR (InR (InR (InL } n))$  $\wedge \text{CodeVal } c$  $= \text{InR}$  $(\text{InR (InR (InR (InR (InR (InL } c))$  $\wedge \text{ClassVal } cl$  $= \text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR (InR (InL } cl))$  $\wedge \text{ExceptionVal}$  $= \text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $(\text{InR}$  $\text{class}))$ **ValuedItem**  $\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet T) f$ **MkValuedItem****VI\_worth****VI\_val** $\vdash \forall t x1 x2$ •  $\text{VI\_worth (MkValuedItem } x1 x2) = x1$  $\wedge \text{VI\_val (MkValuedItem } x1 x2) = x2$

---

	$\wedge MkValuedItem (VI\_worth\ t) (VI\_val\ t) = t$
<b>ValuedItemItem</b>	
<b>NullItemItem</b>	$\vdash \forall v\ n$
	<ul style="list-style-type: none"> <li>• <math>ValuedItemItem\ v = InL\ v \wedge NullItemItem\ n = InR\ n</math></li> </ul>
<b>Data</b>	$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet T)\ f$
<b>MkData</b>	
<b>Dat_class</b>	
<b>Dat_item</b>	$\vdash \forall t\ x1\ x2$ <ul style="list-style-type: none"> <li>• <math>Dat\_class\ (MkData\ x1\ x2) = x1</math></li> <li>  <math>\wedge Dat\_item\ (MkData\ x1\ x2) = x2</math></li> <li>  <math>\wedge MkData\ (Dat\_class\ t)\ (Dat\_item\ t) = t</math></li> </ul>
<b>DataS</b>	$\vdash DataS = Universe$
<b>ItemUpdate</b>	
<b>ClassUpdate</b>	
<b>DataUpdate</b>	$\vdash \forall i\ c\ d$ <ul style="list-style-type: none"> <li>• <math>ItemUpdate\ i = InL\ i</math></li> <li>  <math>\wedge ClassUpdate\ c = InR\ (InL\ c)</math></li> <li>  <math>\wedge DataUpdate\ d = InR\ (InR\ d)</math></li> </ul>
<b>isItem</b>	
<b>isClass</b>	
<b>isData</b>	$\vdash \forall u$ <ul style="list-style-type: none"> <li>• <math>(isItem\ u \Leftrightarrow (\exists i \bullet u = ItemUpdate\ i))</math></li> <li>  <math>\wedge (isClass\ u \Leftrightarrow (\exists c \bullet u = ClassUpdate\ c))</math></li> <li>  <math>\wedge (isData\ u \Leftrightarrow (\exists d \bullet u = DataUpdate\ d))</math></li> </ul>
<b>destItem</b>	
<b>destClass</b>	
<b>destData</b>	$\vdash ConstSpec$ $(\lambda (destItem',\ destClass',\ destData')$ <ul style="list-style-type: none"> <li>• <math>\forall i\ c\ d</math></li> <li>  • <math>destItem'\ (ItemUpdate\ i) = i</math></li> <li>  <math>\wedge destClass'\ (ClassUpdate\ c) = c</math></li> <li>  <math>\wedge destData'\ (DataUpdate\ d) = d</math></li> </ul> $(destItem,\ destClass,\ destData)$
<b>ColSpec</b>	$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet T)\ f$
<b>MkColSpec</b>	
<b>CS_ide</b>	
<b>CS_posn</b>	
<b>CS_dinaryType</b>	
<b>CS_sterlingType</b>	
<b>CS_nullType</b>	
<b>CS_default</b>	
<b>CS_consGroup</b>	
<b>CS_min</b>	
<b>CS_max</b>	$\vdash \forall t\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9$ <ul style="list-style-type: none"> <li>• <math>CS\_ide\ (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) = x1</math></li> <li>  <math>\wedge CS\_posn\ (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9)</math></li> <li>    <math>= x2</math></li> <li>  <math>\wedge CS\_dinaryType</math></li> </ul>

---

---


$$\begin{aligned}
& (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& = x3 \\
& \wedge CS\_sterlingType \\
& \quad (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& \quad = x4 \\
& \wedge (CS\_nullType \\
& \quad (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& \quad \Leftrightarrow x5) \\
& \wedge CS\_default \\
& \quad (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& \quad = x6 \\
& \wedge CS\_consGroup \\
& \quad (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& \quad = x7 \\
& \wedge CS\_min\ (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& \quad = x8 \\
& \wedge CS\_max\ (MkColSpec\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8\ x9) \\
& \quad = x9 \\
& \wedge MkColSpec \\
& \quad (CS\_ide\ t) \\
& \quad (CS\_posn\ t) \\
& \quad (CS\_dinaryType\ t) \\
& \quad (CS\_sterlingType\ t) \\
& \quad (CS\_nullType\ t) \\
& \quad (CS\_default\ t) \\
& \quad (CS\_consGroup\ t) \\
& \quad (CS\_min\ t) \\
& \quad (CS\_max\ t) \\
& \quad = t
\end{aligned}$$

**Reference**  $\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet T)\ f$   
**MkReference**  
**R\_table**  
**R\_group**  $\vdash \forall t\ x1\ x2$ 

- $R\_table\ (MkReference\ x1\ x2) = x1$
- $\wedge R\_group\ (MkReference\ x1\ x2) = x2$
- $\wedge MkReference\ (R\_table\ t)\ (R\_group\ t) = t$

**ColCon**  $\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet T)\ f$   
**MkColCon**  
**CC\_exist**  
**CC\_uniform**  
**CC\_unique**  
**CC\_classLimited**  
**CC\_primary**  
**CC\_secondary**  
**CC\_referential**  $\vdash \forall t\ x1\ x2\ x3\ x4\ x5\ x6\ x7$ 

- $CC\_exist\ (MkColCon\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x1$
- $\wedge (CC\_uniform\ (MkColCon\ x1\ x2\ x3\ x4\ x5\ x6\ x7))$

---

---

	$\Leftrightarrow x2)$ $\wedge (CC\_unique (MkColCon x1 x2 x3 x4 x5 x6 x7)$ $\Leftrightarrow x3)$ $\wedge (CC\_classLimited$ $(MkColCon x1 x2 x3 x4 x5 x6 x7)$ $\Leftrightarrow x4)$ $\wedge (CC\_primary (MkColCon x1 x2 x3 x4 x5 x6 x7)$ $\Leftrightarrow x5)$ $\wedge (CC\_secondary (MkColCon x1 x2 x3 x4 x5 x6 x7)$ $\Leftrightarrow x6)$ $\wedge CC\_referential (MkColCon x1 x2 x3 x4 x5 x6 x7)$ $= x7$ $\wedge MkColCon$ $(CC\_exist t)$ $(CC\_uniform t)$ $(CC\_unique t)$ $(CC\_classLimited t)$ $(CC\_primary t)$ $(CC\_secondary t)$ $(CC\_referential t)$ $= t$
<b>ColConS</b>	$\vdash ColConS = Universe$
<b>Row</b>	$\vdash \exists f \bullet TypeDefn (\lambda x \bullet T) f$
<b>MkRow</b>	
<b>R_exist</b>	
<b>R_data</b>	$\vdash \forall t x1 x2$ <ul style="list-style-type: none"> <li>• <math>R\_exist (MkRow x1 x2) = x1</math></li> <li><math>\wedge R\_data (MkRow x1 x2) = x2</math></li> <li><math>\wedge MkRow (R\_exist t) (R\_data t) = t</math></li> </ul>
<b>RowS</b>	$\vdash RowS = \{r   let data = R\_data r in data \in Num \rightarrow DataS\}$
<b>TableSpec</b>	$\vdash \exists f \bullet TypeDefn (\lambda x \bullet T) f$
<b>MkTableSpec</b>	
<b>TS_class</b>	
<b>TS_maxRow</b>	
<b>TS_colspecs</b>	
<b>TS_cons</b>	
<b>TS_rows</b>	$\vdash \forall t x1 x2 x3 x4 x5$ <ul style="list-style-type: none"> <li>• <math>TS\_class (MkTableSpec x1 x2 x3 x4 x5) = x1</math></li> <li><math>\wedge TS\_maxRow (MkTableSpec x1 x2 x3 x4 x5) = x2</math></li> <li><math>\wedge TS\_colspecs (MkTableSpec x1 x2 x3 x4 x5) = x3</math></li> <li><math>\wedge TS\_cons (MkTableSpec x1 x2 x3 x4 x5) = x4</math></li> <li><math>\wedge TS\_rows (MkTableSpec x1 x2 x3 x4 x5) = x5</math></li> <li><math>\wedge MkTableSpec</math> <ul style="list-style-type: none"> <li><math>(TS\_class t)</math></li> <li><math>(TS\_maxRow t)</math></li> <li><math>(TS\_colspecs t)</math></li> <li><math>(TS\_cons t)</math></li> <li><math>(TS\_rows t)</math></li> </ul> </li> </ul>

---

---

	$= t$
<b>TableSpecS</b>	$\vdash TableSpecS$ $= \{ts$ $  let\ cns = TS\_cons\ ts\ and\ rows = TS\_rows\ ts$ $in\ cns \in Num \rightarrow ColConS$ $\wedge (\forall r \bullet r \in_l\ rows \Rightarrow r \in RowS)\}$
<b>Directory</b>	$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet T)\ f$
<b>MkDirectory</b>	
<b>Dir_tables</b>	
<b>Dir_exist</b>	
<b>Dir_class</b>	$\vdash \forall t\ x1\ x2\ x3$ <ul style="list-style-type: none"> <li>• <math>Dir\_tables\ (MkDirectory\ x1\ x2\ x3) = x1</math></li> <li><math>\wedge Dir\_exist\ (MkDirectory\ x1\ x2\ x3) = x2</math></li> <li><math>\wedge Dir\_class\ (MkDirectory\ x1\ x2\ x3) = x3</math></li> <li><math>\wedge MkDirectory</math>  <math>(Dir\_tables\ t)</math>  <math>(Dir\_exist\ t)</math>  <math>(Dir\_class\ t)</math></li> </ul> $= t$
<b>DirectoryS</b>	$\vdash DirectoryS = \{d   Dir\_tables\ d \in Ide \rightarrow TableSpecS\}$
<b>IdeL</b>	$\vdash IdeL = Universe$
<b>StateS</b>	$\vdash StateS = IdeL \rightarrow DirectoryS$
<b>isState</b>	$\vdash \forall s \bullet isState\ s \Leftrightarrow s \in StateS$
<b>state_type_def</b>	$\vdash \exists f \bullet TypeDefn\ isState\ f$
<b>giveVal</b>	
<b>giveError</b>	
<b>destVal</b>	
<b>destError</b>	
<b>isVal</b>	
<b>isError</b>	$\vdash ConstSpec$ $(\lambda$ $(giveVal', giveError', destVal', destError',$ $isVal', isError'))$ <ul style="list-style-type: none"> <li>• <math>\forall v\ e\ ve</math></li> <li>• <math>giveVal'\ v = InL\ v</math>  <math>\wedge giveError'\ e = InR\ e</math>  <math>\wedge destVal'\ (giveVal'\ v) = v</math>  <math>\wedge destError'\ (giveError'\ e) = e</math>  <math>\wedge (isVal'\ ve \Leftrightarrow (\exists v_1 \bullet ve = giveVal'\ v_1))</math>  <math>\wedge (isError'\ ve</math>  <math>\Leftrightarrow (\exists e_1 \bullet ve = giveError'\ e_1))</math></li> </ul> $(giveVal, giveError, destVal, destError, isVal,$ $isError)$
<b>InsertEffect</b>	
<b>DeleteEffect</b>	
<b>UpdateEffect</b>	
<b>SelectEffect</b>	$\vdash \forall i\ d\ u\ s$

---

---

- $InsertEffect\ i = InL\ i$   
 $\wedge DeleteEffect\ d = InR\ (InL\ d)$   
 $\wedge UpdateEffect\ u = InR\ (InR\ (InL\ u))$   
 $\wedge SelectEffect\ s = InR\ (InR\ (InR\ s))$

**isInsert**  
**isDelete**  
**isUpdate**  
**isSelect**

$\vdash \forall e$

- $(isInsert\ e \Leftrightarrow (\exists i \bullet e = InsertEffect\ i))$   
 $\wedge (isDelete\ e \Leftrightarrow (\exists d \bullet e = DeleteEffect\ d))$   
 $\wedge (isUpdate\ e \Leftrightarrow (\exists u \bullet e = UpdateEffect\ u))$   
 $\wedge (isSelect\ e \Leftrightarrow (\exists s \bullet e = SelectEffect\ s))$

**destInsert**  
**destDelete**  
**destUpdate**  
**destSelect**

$\vdash ConstSpec$

$(\lambda$   
 $(destInsert', destDelete', destUpdate',$   
 $destSelect')$ 

- $\forall i\ d\ u\ s$ 
  - $destInsert'\ (InsertEffect\ i) = i$   
 $\wedge destDelete'\ (DeleteEffect\ d) = d$   
 $\wedge destUpdate'\ (UpdateEffect\ u) = u$   
 $\wedge destSelect'\ (SelectEffect\ s) = s$

 $(destInsert, destDelete, destUpdate, destSelect)$

**tabFromEffect**

$\vdash ConstSpec$

$(\lambda\ tabFromEffect'$   

- $\forall i\ d\ u$ 
  - $tabFromEffect'\ (InsertEffect\ i) = Fst\ i$   
 $\wedge tabFromEffect'\ (DeleteEffect\ d) = Fst\ d$   
 $\wedge tabFromEffect'\ (UpdateEffect\ u) = Fst\ u$

 $tabFromEffect$

## 9.8 Theorems

**dominates\_consistent**  
**lattice\_bottom\_consistent**  
**lattice\_top\_consistent**  
**lub\_consistent**  
**glb\_consistent**

$\vdash Consistent$

$(\lambda$   
 $(dominates', lattice_bottom', lattice_top',$   
 $lub', glb')$ 

- $\forall x\ y\ z$ 
  - $dominates'\ x\ x$   
 $\wedge (dominates'\ x\ y \wedge dominates'\ y\ x \Rightarrow x = y)$

$$\begin{aligned}
& \wedge (\text{dominates}' x y \wedge \text{dominates}' y z \\
& \quad \Rightarrow \text{dominates}' x z) \\
& \wedge (\forall x \bullet \text{dominates}' x \text{ lattice\_bottom}') \\
& \wedge (\forall x \bullet \text{dominates}' \text{ lattice\_top}' x) \\
& \wedge (\forall x y z \\
& \quad \bullet \text{dominates}' (\text{lub}' x y) x \\
& \quad \wedge \text{dominates}' (\text{lub}' x y) y \\
& \quad \wedge (\text{dominates}' z x \wedge \text{dominates}' z y \\
& \quad \quad \Rightarrow \text{dominates}' z (\text{lub}' x y))) \\
& \wedge (\forall x y z \\
& \quad \bullet \text{dominates}' x (\text{glb}' x y) \\
& \quad \wedge \text{dominates}' y (\text{glb}' x y) \\
& \quad \wedge (\text{dominates}' x z \wedge \text{dominates}' y z \\
& \quad \quad \Rightarrow \text{dominates}' (\text{glb}' x y) z))
\end{aligned}$$

## 10 THE THEORY fef005

### 10.1 Parents

*fef004*

### 10.2 Children

*fef014*

### 10.3 Constants

<b>absState</b>	<i>StateR</i> → <i>State</i>
<b>repState</b>	<i>State</i> → <i>StateR</i>
<b>cleanColCons</b>	<i>Class</i> → <i>TableSpec</i> → <i>Worth</i> ↔ <i>ColCon</i> × <i>ColSpec</i> $\mathbb{P}$
<b>filterRow</b>	<i>ColSpec</i> $\mathbb{P}$ → <i>Worth</i> ↔ <i>Data</i> → <i>Worth</i> ↔ <i>Data</i>
<b>Hidden</b>	<i>Ide</i>
<b>dummyVal</b>	<i>Val</i>
<b>replaceData</b>	<i>Class</i> → <i>Data</i> → <i>Data</i>
<b>cleanRow</b>	<i>Class</i> → <i>ColSpec</i> $\mathbb{P}$ → <i>Row</i> → <i>Row</i>
<b>cleanRows</b>	<i>Class</i> → <i>ColSpec</i> $\mathbb{P}$ → <i>Row LIST</i> → <i>Row LIST</i>
<b>cleanTable</b>	<i>Class</i> → <i>TableSpec</i> → <i>TableSpec</i>
<b>cleanDirectory</b>	<i>Class</i> → <i>Directory</i> → <i>Directory</i>
<b>hideR</b>	<i>Class</i> × <i>StateR</i> → <i>StateR</i>
<b>hide</b>	<i>Class</i> × <i>State</i> → <i>State</i>
<b>revealRow</b>	<i>Class</i> → <i>TableSpec</i> → <i>Worth</i> ↔ <i>Worth</i>
<b>replaceRows</b>	<i>TableSpec</i> → <i>Row LIST</i> → <i>TableSpec</i>
<b>changeSpec</b>	<i>Tab</i> → <i>TableSpec</i> → <i>StateR</i> → <i>StateR</i>
<b>visibleCols</b>	<i>Class</i> → <i>TableSpec</i> → <i>ColSpec</i> $\mathbb{P}$
<b>tabExists</b>	<i>Class</i> → <i>Tab</i> → <i>StateR</i> → <i>Bool</i>
<b>getTable</b>	<i>Tab</i> → <i>StateR</i> → <i>TableSpec</i>

---

<b>colDefaults</b>	$Class \rightarrow TableSpec \rightarrow Worth \leftrightarrow Data \rightarrow Worth \leftrightarrow Data$
<b>insertQuery</b>	$Class \times Insert \times StateR \times TableSpec \rightarrow StateR \times Errors$
<b>deleteQuery</b>	$Class \times Delete \times StateR \times TableSpec \rightarrow StateR$
<b>updateField</b>	$Class \rightarrow Class \rightarrow Update \times Data \rightarrow Data + Worth$
<b>updateRow</b>	$Class \rightarrow Class \rightarrow Worth \leftrightarrow Update \times Row \rightarrow Row + Errors$
<b>updateQuery</b>	$Class \times UpDate \times StateR \times TableSpec \rightarrow StateR \times Errors$
<b>updateStateR</b>	$Class \times (Effect \times Errors) \times StateR$ $\rightarrow StateR \times Class \times Select \times Errors$
<b>updateState</b>	$Class \times (Effect \times Errors) \times State$ $\rightarrow State \times Class \times Select \times Errors$

## 10.4 Definitions

<b>repState</b>	
<b>absState</b>	$\vdash ConstSpec$ $(\lambda (repState', absState')$ <ul style="list-style-type: none"> <li>• <math>(\forall a \bullet absState' (repState' a) = a)</math></li> <li><math>\wedge (\forall r</math></li> <li>• <math>isState r \Rightarrow repState' (absState' r) = r)</math></li> <li><math>\wedge (\forall a_1 a_2</math></li> <li>• <math>repState' a_1 = repState' a_2 \Leftrightarrow a_1 = a_2)</math></li> <li><math>\wedge (\forall r_1 r_2</math></li> <li>• <math>isState r_1 \wedge isState r_2</math>  <math>\Rightarrow (absState' r_1 = absState' r_2</math>  <math>\Leftrightarrow r_1 = r_2))</math></li> <li><math>\wedge (\forall s \bullet isState (repState' s)))</math></li> </ul> $(repState, absState)$
<b>cleanColCons</b>	$\vdash \forall clear ts$ <ul style="list-style-type: none"> <li>• <math>cleanColCons clear ts</math>  <math>= (let ccs</math>  <math>= TS\_cons ts</math>  <math>\triangleright \{cc clear\ dominates\ CC\_exist\ cc\}</math>  <math>in (ccs, \{col CS\_consGroup\ col \in Dom\ ccs\}))</math></li> </ul>
<b>filterRow</b>	$\vdash \forall cols ds$ <ul style="list-style-type: none"> <li>• <math>filterRow cols ds</math>  <math>= \{n \exists c \bullet c \in cols \wedge CS\_posn\ c = n\} \triangleleft ds</math></li> </ul>
<b>Hidden</b>	$\vdash T$
<b>dummyVal</b>	$\vdash dummyVal = StringVal Hidden$
<b>replaceData</b>	$\vdash \forall clear d$ <ul style="list-style-type: none"> <li>• <math>replaceData clear d</math>  <math>= (if \neg clear\ dominates\ Dat\_class\ d</math>  <math>then</math>  <math>MkData</math>  <math>(Dat\_class\ d)</math>  <math>(ValuedItemItem</math></li> </ul>

---

(*MkValuedItem worthless dummyVal*)

*else d*)

**cleanRow**     $\vdash \forall \text{ clear cols } r$

- *cleanRow clear cols r*
- = *MkRow*
- (*R\_exist r*)
- (*filterRow cols (R\_data r)*
- ; *Graph (replaceData clear)*)

**cleanRows**     $\vdash \forall \text{ clear cols } rs$

- *cleanRows clear cols rs*
- = *Map*
- (*cleanRow clear cols*)
- (*rs*  $\uparrow$   $\{r \mid \text{clear dominates } R\_exist\ r\}$ )

**cleanTable**     $\vdash \forall \text{ clear } ts$

- *cleanTable clear ts*
- = (*if*  $\neg$  *clear dominates TS\_class ts*
- then*
- MkTableSpec*
- (*TS\_class ts*)
- (*TS\_class ts*)
- { }
- { }
- []
- else*
- (*let (ccs, cols) = cleanColCons clear ts*
- in MkTableSpec*
- (*TS\_class ts*)
- (*TS\_maxRow ts*)
- cols*
- ccs*
- (*cleanRows clear cols (TS\_rows ts)*))

**cleanDirectory**     $\vdash \forall \text{ clear } dir$

- *cleanDirectory clear dir*
- = *MkDirectory*
- (*if clear dominates Dir\_class dir*
- then*
- Dir\_tables dir* ; *Graph (cleanTable clear)*
- else { }*)
- (*Dir\_exist dir*)
- (*Dir\_class dir*)

**hideR**     $\vdash \forall \text{ clear } s$

- *hideR (clear, s)*
- = (*s*  $\triangleright$   $\{dir \mid \text{clear dominates } Dir\_exist\ dir\}$ )
- ; *Graph (cleanDirectory clear)*

**hide**     $\vdash \forall \text{ clear } s$

- *hide (clear, s)*
- = *absState (hideR (clear, repState s))*

---

---

**revealRow**  $\vdash \forall \text{ clear } ts$

- $\text{revealRow clear } ts$

$$= (\text{let visiblerows} \\ = \{r \mid \text{clear dominates } R\_exist\ r\} \\ \text{in Squash} \\ (\text{Id} \\ (\text{Dom} \\ (\text{ListRel } (TS\_rows\ ts) \\ \triangleright \text{visiblerows}))))$$

**replaceRows**  $\vdash \forall ts\ rs$

- $\text{replaceRows } ts\ rs$

$$= \text{MkTableSpec} \\ (TS\_class\ ts) \\ (TS\_maxRow\ ts) \\ (TS\_colspecs\ ts) \\ (TS\_cons\ ts) \\ rs$$

**changeSpec**  $\vdash \forall i\ ts\ s$

- $\text{changeSpec } i\ ts\ s$

$$= (\text{let } dir = s @ \text{Front } i \\ \text{in let } dir' \\ = \text{MkDirectory} \\ (\text{Dir\_tables } dir \oplus \{(Last\ i, ts)\}) \\ (\text{Dir\_exist } dir) \\ (\text{Dir\_class } dir) \\ \text{in } s \oplus \{(Front\ i, dir')\})$$

**visibleCols**  $\vdash \forall \text{ clear } ts$

- $\text{visibleCols clear } ts = \text{Snd } (\text{cleanColCons clear } ts)$

**tabExists**  $\vdash \forall c\ i\ s$

- $\text{tabExists } c\ i\ s$

$$\Leftrightarrow \text{Front } i \in \text{Dom } s \\ \wedge c \text{ dominates } \text{Dir\_exist } (s @ \text{Front } i) \\ \wedge c \text{ dominates } \text{Dir\_class } (s @ \text{Front } i) \\ \wedge \text{Last } i \in \text{Dom } (\text{Dir\_tables } (s @ \text{Front } i))$$

**getTable**  $\vdash \forall i\ s$

- $\text{getTable } i\ s = \text{Dir\_tables } (s @ \text{Front } i) @ \text{Last } i$

**colDefaults**  $\vdash \forall \text{ clear } ts\ ds$

- $\text{colDefaults clear } ts\ ds$

$$= ds \\ \oplus \{(n, d) \\ \mid \exists c \\ \bullet \neg c \in \text{visibleCols clear } ts \\ \wedge n = \text{CS\_posn } c \\ \wedge d = \text{CS\_default } c\}$$

**insertQuery**  $\vdash \forall \text{ clear } i\ ds\ s\ ts$

- $\text{insertQuery } (\text{clear}, (i, ds), s, ts)$

$$= (\text{let } rl$$


---

```

= Map
  (MkRow clear o colDefaults clear ts)
  ds
in if ¬ Elems rl ⊆ RowS
then (s, [ambiguousColumn])
else
  (changeSpec
   i
   (replaceRows ts (TS_rows ts @ rl))
   s, []))

```

**deleteQuery**

```

⊢ ∀ clear i ns e s ts
• deleteQuery (clear, (i, ns), s, ts)
  = (let ns' = revealRow clear ts Image ns
     in let ns''
        = ns'
          ∩ {i
             | R_exist (Nth (TS_rows ts) i) = clear}
        in let rs
           = Extract
             (1 .. # (TS_rows ts) \ ns'')
             (TS_rows ts)
           in changeSpec i (replaceRows ts rs) s)

```

**updateField**

```

⊢ ∀ clear table_class table_d u
• updateField clear table_class (u, table_d)
  = (if clear = table_class
     then
       if isItem u
       then
         giveVal
           (MkData (Dat_class table_d) (destItem u))
       else if isClass u
       then
         if destClass u dominates Dat_class table_d
         then
           giveVal
             (MkData
              (destClass u)
              (Dat_item table_d))
         else giveError downGrade
       else giveVal (destData u)
     else if isItem u
     then
       if Dat_class table_d dominates clear
       then
         giveVal
           (MkData (Dat_class table_d) (destItem u))
       else giveError underClassified

```

---

```

      else giveError classChange)
updateRow  ⊢ ∀ clear table_class us r
  • updateRow clear table_class (us, r)
    = (if ¬ us ∈ Functional
      then giveError [ambiguousUpdate]
      else
        (let us'
          = RelCombine us (R_data r)
          % Graph
          (updateField clear table_class)
        in let es
          = (us' ▷ {x|isError x})
          % Graph destError
        in if es = {}
          then
            giveVal
              (MkRow
                (R_exist r)
                (R_data r ⊕ us' % Graph destVal))
            else giveError (RelList (Squash es))))

updateQuery
  ⊢ ∀ clear i us s ts
  • updateQuery (clear, (i, us), s, ts)
    = (let colnums
      = {n
        | ∃ c
          • c ∈ visibleCols clear ts
            ∧ CS_posn c = n}
      in if ¬ us ∈ Functional
        then (s, [ambiguousUpdate])
        else if ¬ Dom (⋃ (Ran us)) ⊆ colnums
          then (s, [noSuchColumn])
          else
            (let us' = revealRow clear ts ~ % us
              in let pr
                = RelCombine
                  us'
                  (ListRel (TS_rows ts))
                  % Graph
                  (updateRow clear (TS_class ts))
              in let es
                = (pr ▷ {x|isError x})
                % Graph destError
              in if es = {}
                then
                  let rs
                    = RelList
                      (ListRel (TS_rows ts))

```

---

---

```

                                 $\oplus pr$ 
                                 $\% Graph\ destVal)$ 
                                in (changeSpec
                                    i
                                    (replaceRows ts rs)
                                    s, [])
                                else (s, Flat (RelList (Squash es))))))
updateStateR  $\vdash \forall clear\ ef\ es\ s$ 
  • updateStateR (clear, (ef, es), s)
    = (if isSelect ef
        then (s, clear, destSelect ef, es)
        else if  $\neg es = []$ 
            then (s, clear, [], es)
            else
                (let i = tabFromEffect ef
                  in if tabExists clear i s
                      then
                          let ts = getTable i s
                            in if  $\neg clear\ dominates\ TS\_class\ ts$ 
                                then (s, clear, [], [accessDenied])
                                else
                                    (let (s', es')
                                        = (if isInsert ef
                                            then
                                                insertQuery
                                                  (clear, destInsert ef, s,
                                                    ts)
                                            else if isDelete ef
                                                then
                                                    deleteQuery
                                                      (clear,
                                                        destDelete ef, s,
                                                        ts), [])
                                        else
                                            updateQuery
                                              (clear, destUpdate ef, s,
                                                ts))
                                        in (s', clear, [], es'))
                                    else (s, clear, [], [noSuchTable])))
updateState  $\vdash \forall clear\ ef\ es\ s$ 
  • updateState (clear, (ef, es), s)
    = (let (sr, out)
        = updateStateR
          (clear, (ef, es), repState s)
        in (absState sr, out))

```

---

## 11 THE THEORY fef006

### 11.1 Parents

*fef014*

### 11.2 Children

*fef022 fef007*

### 11.3 Constants

<b>mkTf</b>	<i>Hide</i> $\rightarrow$ <i>Process</i> $\rightarrow$ <i>Ustate</i> $\rightarrow$ <i>Stf</i>
<b>isstate</b>	<i>State</i>
<b>SSQLam</b>	<i>Am</i>
<b>iterate</b>	$((('QUERY \times Class) \times 'STATE \rightarrow 'STATE \times Class \times 'DATA)$ $\rightarrow ('QUERY \times Class) LIST \times 'STATE$ $\rightarrow 'STATE \times (Class \times 'DATA) LIST$
<b>behaviours</b>	$((('QUERY \times Class) \times 'STATE \rightarrow 'STATE \times Class \times 'DATA)$ $\times 'STATE$ $\rightarrow ('QUERY, 'DATA) BEHAVIOURS$

### 11.4 Type Abbreviations

<b>Hide</b>	<i>Hide</i>
<b>Process</b>	<i>Process</i>
<b>Ustate</b>	<i>Ustate</i>
<b>Stf</b>	<i>Stf</i>
<b>Am</b>	<i>Am</i>

### 11.5 Definitions

<b>mkTf</b>	$\vdash \forall h p u q c s$ <ul style="list-style-type: none"> <li>• <math>mkTf\ h\ p\ u\ ((q, c), s)</math>  <math>= u\ (c, p\ (q, c, h\ (c, s)), s)</math></li> </ul>
<b>isstate</b>	$\vdash true$
<b>SSQLam</b>	$\vdash SSQLam$ $= (mkTf\ hide\ processQuery\ updateState, isstate)$
<b>iterate</b>	$\vdash ConstSpec$ $(\lambda\ iterate'$ <ul style="list-style-type: none"> <li>• <math>\forall s\ i\ si\ f</math></li> <li>• <math>iterate'\ f\ ([], s) = (s, [])</math>  <math>\wedge (let\ sso = iterate'\ f\ (si, s)</math>  <math>in\ iterate'\ f\ (si\ @\ [i], s)</math>  <math>= (Fst\ (f\ (i, Fst\ sso)),</math>  <math>Snd\ sso\ @\ [Snd\ (f\ (i, Fst\ sso))]))))</math>  <math>iterate</math></li> </ul>

**behaviours**  $\vdash \forall tf \text{ ivate } si$   
 $\bullet \text{ behaviours } (tf, \text{ ivate}) si$   
 $= \text{ Snd } (\text{ iterate } tf (si, \text{ ivate}))$

## 12 THE THEORY fef007

### 12.1 Parents

*fef006*

### 12.2 Children

*fef009*

### 12.3 Constants

**secureHide** *Hide*  $\mathbb{P}$   
**secureUpdate** *Hide*  $\leftrightarrow$  *Ustate*  
**Lemma1** *Bool*  
**Lemma2** *Bool*  
**secureStf** *Stf*  $\mathbb{P}$   
**SSQLtf** *Stf*  
**Lemma3** *Bool*  
**secureItf** *Itf*  $\mathbb{P}$   
**Lemma4** *Bool*  
**Lemma5** *Bool*

### 12.4 Type Abbreviations

**Itf** *Itf*

### 12.5 Definitions

**secureHide**  $\vdash \forall h$   
 $\bullet h \in \text{ secureHide}$   
 $\Leftrightarrow (\forall c_1 c_2 s_1 s_2$   
 $\bullet h (c_1, s_1) = h (c_1, s_2) \wedge c_1 \text{ dominates } c_2$   
 $\Rightarrow h (c_2, s_1) = h (c_2, s_2))$

**secureUpdate**  $\vdash \forall h u$   
 $\bullet (h, u) \in \text{ secureUpdate}$   
 $\Leftrightarrow (\forall c_1 c_2 s e$   
 $\bullet (\text{ let } s' = \text{ Fst } (u (c_1, e, s))$   
 $\text{ in } \neg h (c_2, s) = h (c_2, s')$   
 $\Rightarrow c_2 \text{ dominates } c_1))$   
 $\wedge (\forall c_1 c_2 s_1 s_2 e$   
 $\bullet (\text{ let } s'_1 = \text{ Fst } (u (c_2, e, s_1))$

---

	$\begin{aligned} & \text{and } s'_2 = \text{Fst } (u (c_2, e, s_2)) \\ & \text{in } h (c_1, s_1) = h (c_1, s_2) \\ & \quad \wedge c_1 \text{ dominates } c_2 \\ & \quad \Rightarrow h (c_1, s'_1) = h (c_1, s'_2))) \\ & \wedge (\forall c s_1 s_2 e \\ & \quad \bullet (\text{let } o_1 = \text{Snd } (u (c, e, s_1)) \\ & \quad \quad \text{and } o_2 = \text{Snd } (u (c, e, s_2)) \\ & \quad \quad \text{in } h (c, s_1) = h (c, s_2) \Rightarrow o_1 = o_2)) \\ & \quad \wedge (\forall c s e \bullet \text{Fst } (\text{Snd } (u (c, e, s))) = c) \end{aligned}$
<b>Lemma1</b>	$\begin{aligned} & \vdash \text{Lemma1} \\ & \Leftrightarrow \text{hide} \in \text{secureHide} \\ & \quad \wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate} \end{aligned}$
<b>Lemma2</b>	$\begin{aligned} & \vdash \text{Lemma2} \\ & \Leftrightarrow \text{hide} \in \text{secureHide} \\ & \quad \wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate} \\ & \quad \Rightarrow \text{behaviours SSQLam} \in \text{secure} \end{aligned}$
<b>secureStf</b>	$\begin{aligned} & \vdash \forall \text{stf} \\ & \bullet \text{stf} \in \text{secureStf} \\ & \Leftrightarrow (\forall s_1 s_2 i_1 i_2 c \\ & \quad \bullet \text{hide } (c, s_1) = \text{hide } (c, s_2) \\ & \quad \quad \wedge ([i_1], [i_2]) \in \text{same\_ins } c \\ & \quad \quad \Rightarrow (\text{let } (s'_1, o_1) = \text{stf } (i_1, s_1) \\ & \quad \quad \quad \text{and } (s'_2, o_2) = \text{stf } (i_2, s_2) \\ & \quad \quad \quad \text{in } \text{hide } (c, s'_1) = \text{hide } (c, s'_2) \\ & \quad \quad \quad \wedge ([o_1], [o_2]) \in \text{same\_outs } c)) \\ & \quad \wedge (\forall s i c \\ & \quad \bullet \neg c \text{ dominates } \text{Snd } i \\ & \quad \quad \Rightarrow \text{hide } (c, s) = \text{hide } (c, \text{Fst } (\text{stf } (i, s))) \\ & \quad \quad \quad \wedge \neg c \text{ dominates } \text{Fst } (\text{Snd } (\text{stf } (i, s)))) \end{aligned}$
<b>SSQLtf</b>	$\vdash \text{SSQLtf} = \text{mkTf } \text{hide } \text{processQuery } \text{updateState}$
<b>Lemma3</b>	$\begin{aligned} & \vdash \text{Lemma3} \\ & \Leftrightarrow \text{hide} \in \text{secureHide} \\ & \quad \wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate} \\ & \quad \Rightarrow \text{SSQLtf} \in \text{secureStf} \end{aligned}$
<b>secureItf</b>	$\begin{aligned} & \vdash \forall \text{itf} \\ & \bullet \text{itf} \in \text{secureItf} \\ & \Leftrightarrow (\forall s_1 s_2 si_1 si_2 c \\ & \quad \bullet \text{hide } (c, s_1) = \text{hide } (c, s_2) \\ & \quad \quad \wedge (si_1, si_2) \in \text{same\_ins } c \\ & \quad \quad \Rightarrow (\text{let } (s'_1, so_1) = \text{itf } (si_1, s_1) \\ & \quad \quad \quad \text{and } (s'_2, so_2) = \text{itf } (si_2, s_2) \\ & \quad \quad \quad \text{in } \text{hide } (c, s'_1) = \text{hide } (c, s'_2) \\ & \quad \quad \quad \wedge (so_1, so_2) \in \text{same\_outs } c)) \end{aligned}$
<b>Lemma4</b>	$\begin{aligned} & \vdash \text{Lemma4} \\ & \Leftrightarrow \text{SSQLtf} \in \text{secureStf} \\ & \quad \Rightarrow \text{iterate SSQLtf} \in \text{secureItf} \end{aligned}$
<b>Lemma5</b>	$\begin{aligned} & \vdash \text{Lemma5} \\ & \Leftrightarrow \text{iterate SSQLtf} \in \text{secureItf} \end{aligned}$

---

$\Rightarrow$  behaviours  $SSQLam \in secure$

## 12.6 Theorems

**lemma2\_thm**  $\vdash Lemma3 \wedge Lemma4 \wedge Lemma5 \Rightarrow Lemma2$

**lemma1\_2\_thm**  $\vdash Lemma1 \wedge Lemma2 \Rightarrow behaviours\ SSQLam \in secure$

**main\_thm**  $\vdash Lemma1 \wedge Lemma3 \wedge Lemma4 \wedge Lemma5$   
 $\Rightarrow behaviours\ SSQLam \in secure$

## 13 THE THEORY fef009

### 13.1 Parents

*fef007*

### 13.2 Children

*fef010*

### 13.3 Constants

**iterate\_witness**

$(( 'QUERY \times Class ) \times 'STATE \rightarrow 'STATE \times Class \times 'DATA)$   
 $\rightarrow 'STATE$   
 $\rightarrow ( 'QUERY \times Class ) LIST$   
 $\rightarrow 'STATE \times ( Class \times 'DATA ) LIST$

### 13.4 Definitions

**iterate\_witness**

$\vdash \forall s\ i\ si\ f$

- $iterate\_witness\ f\ s\ [] = (s, [])$   
 $\wedge\ iterate\_witness\ f\ s\ (Cons\ i\ si)$   
 $= (Fst\ (f\ (i,\ Fst\ (iterate\_witness\ f\ s\ si))),$   
 $Snd\ (iterate\_witness\ f\ s\ si))$   
 $\quad @\ [Snd$   
 $\quad\ (f$   
 $\quad\ (i,$   
 $\quad\ Fst$   
 $\quad\ (iterate\_witness$   
 $\quad\ f$   
 $\quad\ s$   
 $\quad\ si)))]])$

## 13.5 Theorems

**Lemma3**  $\vdash$  *Lemma3*

**iterate\_consistent**

$\vdash$  *Consistent*

$(\lambda$  *iterate'*

•  $\forall s i si f$

• *iterate' f* ( $\square$ ,  $s$ ) = ( $s$ ,  $\square$ )

$\wedge$  (*let*  $sso =$  *iterate' f* ( $si$ ,  $s$ ))

*in* *iterate' f* ( $si$  @  $[i]$ ,  $s$ )

= (*Fst* ( $f$  ( $i$ , *Fst*  $sso$ )),

*Snd*  $sso$  @ [*Snd* ( $f$  ( $i$ , *Fst*  $sso$ ))]))))

**Lemma4**  $\vdash$  *Lemma4*

**Lemma5**  $\vdash$  *Lemma5*

**main\_thm1**  $\vdash$  *Lemma1*  $\Rightarrow$  *behaviours* *SSQLam*  $\in$  *secure*

## 14 THE THEORY fef010

### 14.1 Parents

*fef009*

### 14.2 Children

*fef011* *fef040*

### 14.3 Constants

**secureHideR** (*Class*  $\times$  *StateR*  $\rightarrow$  *StateR*)  $\mathbb{P}$

### 14.4 Definitions

**secureHideR**  $\vdash \forall h$

•  $h \in$  *secureHideR*

$\Leftrightarrow (\forall c_1 c_2 s_1 s_2$

•  $h(c_1, s_1) = h(c_1, s_2) \wedge c_1$  *dominates*  $c_2$

$\Rightarrow h(c_2, s_1) = h(c_2, s_2))$

---

## 14.5 Theorems

**repState\_consistent****absState\_consistent** $\vdash$  *Consistent* $(\lambda (repState', absState')$ 

- $(\forall a \bullet absState' (repState' a) = a)$

 $\wedge (\forall r$ 

- $isState r \Rightarrow repState' (absState' r) = r)$

 $\wedge (\forall a_1 a_2$ 

- $repState' a_1 = repState' a_2 \Leftrightarrow a_1 = a_2)$

 $\wedge (\forall r_1 r_2$ 

- $isState r_1 \wedge isState r_2$

 $\Rightarrow (absState' r_1 = absState' r_2$  $\Leftrightarrow r_1 = r_2))$  $\wedge (\forall s \bullet isState (repState' s)))$ **dir\_components** $\vdash \forall dir_1 dir_2$ 

- $dir_1 = dir_2$

 $\Leftrightarrow Dir\_tables dir_1 = Dir\_tables dir_2$  $\wedge Dir\_exist dir_1 = Dir\_exist dir_2$  $\wedge Dir\_class dir_1 = Dir\_class dir_2$ **tab\_components** $\vdash \forall t_1 t_2$ 

- $t_1 = t_2$

 $\Leftrightarrow TS\_class t_1 = TS\_class t_2$  $\wedge TS\_maxRow t_1 = TS\_maxRow t_2$  $\wedge TS\_colspecs t_1 = TS\_colspecs t_2$  $\wedge TS\_cons t_1 = TS\_cons t_2$  $\wedge TS\_rows t_1 = TS\_rows t_2$ **row\_components** $\vdash \forall r_1 r_2$ 

- $r_1 = r_2$

 $\Leftrightarrow R\_exist r_1 = R\_exist r_2$  $\wedge R\_data r_1 = R\_data r_2$ **data\_components** $\vdash \forall d_1 d_2$ 

- $d_1 = d_2$

 $\Leftrightarrow Dat\_class d_1 = Dat\_class d_2$  $\wedge Dat\_item d_1 = Dat\_item d_2$ **hideR\_hide\_lemma** $\vdash hideR \in secureHideR$  $\wedge (\forall clear s$ 

- $isState s \Rightarrow isState (hideR (clear, s)))$

 $\Rightarrow hide \in secureHide$ **replaceData\_lemma** $\vdash \forall c_1 c_2$ 

- $c_1$  dominates  $c_2$

$$\Rightarrow (\forall d_1 d_2$$

- $\text{replaceData } c_1 d_1 = \text{replaceData } c_1 d_2$

$$\Rightarrow \text{replaceData } c_2 d_1 = \text{replaceData } c_2 d_2)$$
**cleanColCons\_lemma**

$$\vdash \forall c_1 c_2 t_1 t_2$$

- $c_1$  dominates  $c_2$

$$\wedge \text{cleanColCons } c_1 t_1 = \text{cleanColCons } c_1 t_2$$

$$\Rightarrow \text{cleanColCons } c_2 t_1 = \text{cleanColCons } c_2 t_2$$
 **$\subseteq$ \_cleanColCons\_lemma**

$$\vdash \forall c_1 c_2 t$$

- $c_1$  dominates  $c_2$

$$\Rightarrow \text{Fst } (\text{cleanColCons } c_2 t)$$

$$\subseteq \text{Fst } (\text{cleanColCons } c_1 t)$$

$$\wedge \text{Snd } (\text{cleanColCons } c_2 t)$$

$$\subseteq \text{Snd } (\text{cleanColCons } c_1 t)$$
 **$\subseteq$ \_cleanColCons\_lemma1**

$$\vdash \forall c_1 c_2 t \text{ col}$$

- $c_1$  dominates  $c_2$

$$\Rightarrow \text{col} \in \text{Snd } (\text{cleanColCons } c_2 t)$$

$$\Rightarrow \text{col} \in \text{Snd } (\text{cleanColCons } c_1 t)$$
**cleanRow\_lemma**

$$\vdash \forall c_1 c_2$$

- $c_1$  dominates  $c_2$

$$\Rightarrow (\forall t_1 t_2 r_1 r_2$$

- $\text{cleanColCons } c_1 t_1 = \text{cleanColCons } c_1 t_2$

$$\wedge \text{cleanColCons } c_2 t_1$$

$$= \text{cleanColCons } c_2 t_2$$

$$\wedge \text{cleanRow}$$

$$\begin{array}{l} c_1 \\ (\text{Snd } (\text{cleanColCons } c_1 t_1)) \\ r_1 \\ = \text{cleanRow} \\ c_1 \\ (\text{Snd } (\text{cleanColCons } c_1 t_2)) \\ r_2 \end{array}$$

$$\Rightarrow \text{cleanRow}$$

$$\begin{array}{l} c_2 \\ (\text{Snd } (\text{cleanColCons } c_2 t_1)) \\ r_1 \\ = \text{cleanRow} \\ c_2 \\ (\text{Snd } (\text{cleanColCons } c_2 t_2)) \\ r_2 \end{array})$$
**cleanTable\_lemma**

$$\vdash \forall c_1 c_2$$

- $c_1$  dominates  $c_2$

$$\Rightarrow (\forall t_1 t_2$$

- $\text{cleanTable } c_1 t_1 = \text{cleanTable } c_1 t_2$

$$\Rightarrow \text{cleanTable } c_2 \ t_1 = \text{cleanTable } c_2 \ t_2)$$
**cleanDirectory\_lemma**

$$\vdash \forall c_1 \ c_2$$

- $c_1$  dominates  $c_2$

$$\Rightarrow (\forall d_1 \ d_2$$

- $\text{cleanDirectory } c_1 \ d_1 = \text{cleanDirectory } c_1 \ d_2$

$$\Rightarrow \text{cleanDirectory } c_2 \ d_1$$

$$= \text{cleanDirectory } c_2 \ d_2)$$
 **$\subseteq$ \_dir\_lemma**  $\vdash \forall c_1 \ c_2 \ s \ x \ y$ 

- $c_1$  dominates  $c_2$

$$\Rightarrow (x, y) \in s \triangleright \{\text{dir} | c_2 \text{ dominates } \text{Dir\_exist } \text{dir}\}$$

$$\Rightarrow (x, y) \in s \triangleright \{\text{dir} | c_1 \text{ dominates } \text{Dir\_exist } \text{dir}\}$$
**dir\_class\_preserved\_lemma**

$$\vdash \forall c \ c_1 \ s \ s_1 \ x \ z \ z_1$$

- $\text{cleanDirectory } c_1 \ z = \text{cleanDirectory } c_1 \ z_1$

$$\wedge (x, z)$$

$$\in s \triangleright \{\text{dir} | c_1 \text{ dominates } \text{Dir\_exist } \text{dir}\}$$

$$\wedge (x, z_1)$$

$$\in s_1 \triangleright \{\text{dir} | c \text{ dominates } \text{Dir\_exist } \text{dir}\}$$

$$\Rightarrow (x, z_1)$$

$$\in s_1 \triangleright \{\text{dir} | c_1 \text{ dominates } \text{Dir\_exist } \text{dir}\}$$
**secureHideR\_lemma**

$$\vdash \text{hideR} \in \text{secureHideR}$$
**cleanDir\_fun\_thm**

$$\vdash \forall \text{clear} \bullet \text{Graph } (\text{cleanDirectory } \text{clear}) \in \text{Functional}$$
**replaceData\_fun\_thm**

$$\vdash \forall \text{clear} \bullet \text{Graph } (\text{replaceData } \text{clear}) \in \text{Functional}$$
**cleanColCons\_fun\_thm**

$$\vdash \forall \text{clear } \text{tab}$$

- $\text{TS\_cons } \text{tab} \in \text{Functional}$

$$\Rightarrow \text{Fst } (\text{cleanColCons } \text{clear } \text{tab}) \in \text{Functional}$$
**tables\_fun\_thm**

$$\vdash \forall \text{clear } \text{dir}$$

- $\text{Dir\_tables } \text{dir} \in \text{Functional}$

$$\Rightarrow \text{Dir\_tables } (\text{cleanDirectory } \text{clear } \text{dir})$$

$$\in \text{Functional}$$

$$\text{hideR\_lemma } \vdash \forall \text{clear } s \bullet \text{isState } s \Rightarrow \text{isState } (\text{hideR } (\text{clear}, s))$$
**secureHide\_lemma**

$$\vdash \text{hide} \in \text{secureHide}$$

## 15 THE THEORY fef011

### 15.1 Parents

$$\text{fef010}$$

## 15.2 Children

*fef012*

## 15.3 Theorems

**destInsert\_consistent**  
**destDelete\_consistent**  
**destUpdate\_consistent**  
**destSelect\_consistent**

$\vdash$  *Consistent*  
( $\lambda$   
  (*destInsert'*, *destDelete'*, *destUpdate'*,  
   *destSelect'*)  
  •  $\forall i d u s$   
  • *destInsert'* (*InsertEffect* *i*) = *i*  
     $\wedge$  *destDelete'* (*DeleteEffect* *d*) = *d*  
     $\wedge$  *destUpdate'* (*UpdateEffect* *u*) = *u*  
     $\wedge$  *destSelect'* (*SelectEffect* *s*) = *s*)

**tabFromEffect\_consistent**

$\vdash$  *Consistent*  
( $\lambda$  *tabFromEffect'*  
  •  $\forall i d u$   
  • *tabFromEffect'* (*InsertEffect* *i*) = *Fst* *i*  
     $\wedge$  *tabFromEffect'* (*DeleteEffect* *d*) = *Fst* *d*  
     $\wedge$  *tabFromEffect'* (*UpdateEffect* *u*) = *Fst* *u*)

**giveVal\_consistent**  
**giveError\_consistent**  
**destVal\_consistent**  
**destError\_consistent**  
**isVal\_consistent**  
**isError\_consistent**

$\vdash$  *Consistent*  
( $\lambda$   
  (*giveVal'*, *giveError'*, *destVal'*, *destError'*,  
   *isVal'*, *isError'*)  
  •  $\forall v e ve$   
  • *giveVal'* *v* = *InL* *v*  
     $\wedge$  *giveError'* *e* = *InR* *e*  
     $\wedge$  *destVal'* (*giveVal'* *v*) = *v*  
     $\wedge$  *destError'* (*giveError'* *e*) = *e*  
     $\wedge$  (*isVal'* *ve*  $\Leftrightarrow$  ( $\exists v_1 \bullet ve = giveVal' v_1$ ))  
     $\wedge$  (*isError'* *ve*  
       $\Leftrightarrow$  ( $\exists e_1 \bullet ve = giveError' e_1$ )))

**giveVal\_eq\_thm**

$\vdash \forall x y \bullet giveVal x = giveVal y \Leftrightarrow x = y$

**giveError\_eq\_thm**

$\vdash \forall x y \bullet giveError x = giveError y \Leftrightarrow x = y$

**¬isError\_giveVal\_thm** $\vdash \forall v \bullet \neg \text{isError } (\text{giveVal } v)$ **¬isVal\_giveError\_thm** $\vdash \forall e \bullet \neg \text{isVal } (\text{giveError } e)$ **query\_type** $\vdash \forall q$ 

- $\text{isInsert } q$ 
  - $\wedge \neg (\text{isDelete } q \vee \text{isUpdate } q \vee \text{isSelect } q)$
  - $\vee \text{isDelete } q$
  - $\wedge \neg (\text{isInsert } q \vee \text{isUpdate } q \vee \text{isSelect } q)$
  - $\vee \text{isUpdate } q$
  - $\wedge \neg (\text{isInsert } q \vee \text{isDelete } q \vee \text{isSelect } q)$
  - $\vee \text{isSelect } q$
  - $\wedge \neg (\text{isInsert } q \vee \text{isDelete } q \vee \text{isUpdate } q)$

**update\_type** $\vdash \forall u$ 

- $\text{isItem } u \wedge \neg (\text{isClass } u \vee \text{isData } u)$ 
  - $\vee \text{isClass } u \wedge \neg (\text{isItem } u \vee \text{isData } u)$
  - $\vee \text{isData } u \wedge \neg (\text{isItem } u \vee \text{isClass } u)$

**val\_or\_error\_type** $\vdash \forall v \bullet \text{isVal } v \wedge \neg \text{isError } v \vee \text{isError } v \wedge \neg \text{isVal } v$ **conjunct4** $\vdash \forall c \ s \ e \bullet \text{Fst } (\text{Snd } (\text{updateState } (c, e, s))) = c$ **isState\_lemma3** $\vdash \forall c \ s \bullet \text{isState } (\text{hideR } (c, \text{repState } s))$ **hide\_eq\_lemma** $\vdash \forall c \ s_1 \ s_2$ 

- $\text{hide } (c, s_1) = \text{hide } (c, s_2)$ 
  - $\Leftrightarrow \text{hideR } (c, \text{repState } s_1)$
  - $= \text{hideR } (c, \text{repState } s_2)$

**tabExists\_lemma** $\vdash \forall c \ s_1 \ s_2$ 

- $\text{hide } (c, s_1) = \text{hide } (c, s_2)$ 
  - $\Rightarrow (\forall i$ 
    - $\text{tabExists } c \ i \ (\text{repState } s_1)$
    - $\Leftrightarrow \text{tabExists } c \ i \ (\text{repState } s_2)$

**cleanTable\_insertQuery\_lemma** $\vdash \forall c \ t_1 \ t_2 \ s_1 \ s_2 \ i$ 

- $c \text{ dominates } \text{TS\_class } t_1$ 
  - $\wedge c \text{ dominates } \text{TS\_class } t_2$
  - $\wedge \text{cleanTable } c \ t_1 = \text{cleanTable } c \ t_2$
  - $\Rightarrow \text{Snd } (\text{insertQuery } (c, i, s_1, t_1))$
  - $= \text{Snd } (\text{insertQuery } (c, i, s_2, t_2))$

**updateRow\_lemma** $\vdash \forall c \ t_1 \ t_2$ 

- $\text{cleanTable } c \ t_1 = \text{cleanTable } c \ t_2$ 
  - $\Rightarrow c \text{ dominates } \text{TS\_class } t_1$
  - $\wedge c \text{ dominates } \text{TS\_class } t_2$
  - $\Rightarrow \text{updateRow } c \ (\text{TS\_class } t_1)$
  - $= \text{updateRow } c \ (\text{TS\_class } t_2)$

**isError\_updateField\_lemma**

$$\begin{aligned} & \vdash \forall c \, tc \, d_1 \, d_2 \, u \\ & \bullet \text{isError} (\text{updateField} \, c \, tc \, (u, \, d_1)) \\ & \quad \wedge \text{replaceData} \, c \, d_1 = \text{replaceData} \, c \, d_2 \\ & \quad \Rightarrow \text{updateField} \, c \, tc \, (u, \, d_1) \\ & \quad = \text{updateField} \, c \, tc \, (u, \, d_2) \end{aligned}$$

**isError\_updateRow\_lemma**

$$\begin{aligned} & \vdash \forall c \, r_1 \, r_2 \, t \, u \\ & \bullet \text{isError} (\text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_1)) \\ & \quad \wedge \text{isError} (\text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_2)) \\ & \quad \wedge \text{cleanRow} \, c \, (Snd \, (\text{cleanColCons} \, c \, t)) \, r_1 \\ & \quad = \text{cleanRow} \, c \, (Snd \, (\text{cleanColCons} \, c \, t)) \, r_2 \\ & \quad \wedge \text{Dom} \, u \\ & \quad \subseteq \{n\} \\ & \quad |\exists c' \bullet c' \in \text{visibleCols} \, c \, t \wedge CS\_posn \, c' = n\} \\ & \Rightarrow \text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_1) \\ & = \text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_2) \end{aligned}$$

**isVal\_updateRow\_lemma**

$$\begin{aligned} & \vdash \forall c \, r_1 \, r_2 \, t \, u \\ & \bullet \text{isVal} (\text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_1)) \\ & \quad \wedge \text{cleanRow} \, c \, (Snd \, (\text{cleanColCons} \, c \, t)) \, r_1 \\ & \quad = \text{cleanRow} \, c \, (Snd \, (\text{cleanColCons} \, c \, t)) \, r_2 \\ & \quad \wedge \text{Dom} \, u \\ & \quad \subseteq \{n\} \\ & \quad |\exists c' \bullet c' \in \text{visibleCols} \, c \, t \wedge CS\_posn \, c' = n\} \\ & \Rightarrow \text{isVal} (\text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_2)) \end{aligned}$$

**isError\_⇔\_updateRow\_lemma**

$$\begin{aligned} & \vdash \forall c \, r_1 \, r_2 \, t \, u \\ & \bullet \text{cleanRow} \, c \, (Snd \, (\text{cleanColCons} \, c \, t)) \, r_1 \\ & \quad = \text{cleanRow} \, c \, (Snd \, (\text{cleanColCons} \, c \, t)) \, r_2 \\ & \quad \wedge \text{Dom} \, u \\ & \quad \subseteq \{n\} \\ & \quad |\exists c' \bullet c' \in \text{visibleCols} \, c \, t \wedge CS\_posn \, c' = n\} \\ & \Rightarrow (\text{isError} (\text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_1))) \\ & \quad \Leftrightarrow \text{isError} (\text{updateRow} \, c \, (TS\_class \, t) \, (u, \, r_2)) \end{aligned}$$

**dom\_∪\_ran\_lemma**

$$\begin{aligned} & \vdash \forall x \, y \, u \, c \, t \\ & \bullet (x, \, y) \in u \\ & \quad \wedge \text{Dom} \, (\cup \, (\text{Ran} \, u)) \\ & \quad \subseteq \{n\} \\ & \quad |\exists c' \\ & \quad \bullet c' \in \text{Snd} \, (\text{cleanColCons} \, c \, t) \\ & \quad \quad \wedge \text{CS\_posn} \, c' = n\} \\ & \Rightarrow \text{Dom} \, y \\ & \quad \subseteq \{n\} \\ & \quad |\exists c' \bullet c' \in \text{visibleCols} \, c \, t \wedge CS\_posn \, c' = n\} \end{aligned}$$

**doms\_null\_lemma1**

$$\begin{aligned} & \vdash \forall l \, last \, s \, u \\ & \bullet \text{Dom} \, (\text{Squash} \, (\text{Id} \, (\text{Dom} \, (\text{ListRel} \, l \, \triangleright \, s))) \, \sim \, \% \, u) \end{aligned}$$

$$\begin{aligned} & \cap \text{Dom } \{(\# l + 1, \text{last})\} \\ & = \{\} \end{aligned}$$

**doms\_null\_lemma2**

$$\begin{aligned} & \vdash \forall l s u \\ & \bullet \text{Dom} \\ & \quad (\{(\# (\text{Squash } (\text{Id } (\text{Dom } (\text{ListRel } l \triangleright s)))) + 1, \\ & \quad \quad \# l + 1)\} \sim \\ & \quad \quad \circlearrowleft u) \\ & \quad \cap \text{Dom } (\text{ListRel } l) \\ & = \{\} \end{aligned}$$

**squash\_doms\_lemma**

$$\begin{aligned} & \vdash \forall l \text{last } s u n_1 n_2 \\ & \bullet n_1 \\ & \quad \in \text{Dom} \\ & \quad \quad (((\text{RelCombine} \\ & \quad \quad \quad (\text{Squash} \\ & \quad \quad \quad \quad (\text{Id} \\ & \quad \quad \quad \quad \quad (\text{Dom} \\ & \quad \quad \quad \quad \quad \quad (\text{ListRel } l \\ & \quad \quad \quad \quad \quad \quad \quad \triangleright s)))) \sim \\ & \quad \quad \quad \quad \circlearrowleft u) \\ & \quad \quad \quad (\text{ListRel } l) \\ & \quad \quad \quad \circlearrowleft \text{Graph} \\ & \quad \quad \quad (\text{updateRow } c (\text{TS\_class } t_2))) \\ & \quad \quad \quad \triangleright \{x | \text{isError } x\} \\ & \quad \quad \quad \circlearrowleft \text{Graph } \text{destError}) \\ & \quad \wedge n_2 \\ & \quad \in \text{Dom} \\ & \quad \quad (((\text{RelCombine} \\ & \quad \quad \quad (\{(\# \\ & \quad \quad \quad \quad (\text{Squash} \\ & \quad \quad \quad \quad \quad (\text{Id} \\ & \quad \quad \quad \quad \quad \quad (\text{Dom } (\text{ListRel } l \triangleright s)))) \\ & \quad \quad \quad \quad \quad \quad \quad + 1, \# l + 1)\} \sim \\ & \quad \quad \quad \quad \circlearrowleft u) \\ & \quad \quad \quad \{(\# l + 1, \text{last})\} \\ & \quad \quad \quad \circlearrowleft \text{Graph} \\ & \quad \quad \quad (\text{updateRow } c (\text{TS\_class } t_2))) \\ & \quad \quad \quad \triangleright \{x | \text{isError } x\} \\ & \quad \quad \quad \circlearrowleft \text{Graph } \text{destError}) \\ & \Rightarrow n_2 > n_1 \end{aligned}$$

**cleanRows\_size\_lemma**

$$\begin{aligned} & \vdash \forall c t_1 t_2 \\ & \bullet \text{cleanRows} \\ & \quad c \\ & \quad (\text{Snd } (\text{cleanColCons } c t_2)) \\ & \quad (\text{TS\_rows } t_1) \\ & = \text{cleanRows} \end{aligned}$$

$$\begin{aligned}
& c \\
& (Snd (cleanColCons c t_2)) \\
& (TS\_rows t_2) \\
\Rightarrow & \# \\
& (ListRel (TS\_rows t_1) \\
& \triangleright \{r | c \text{ dominates } R\_exist r\}) \\
= & \# \\
& (ListRel (TS\_rows t_2) \\
& \triangleright \{r | c \text{ dominates } R\_exist r\})
\end{aligned}$$

**cleanRows\_errors\_or\_vals\_lemma**

$$\begin{aligned}
& \vdash \forall c t_1 t_2 u \\
& \bullet \text{ Dom } (\cup (\text{Ran } u)) \\
& \subseteq \{n \\
& \mid \exists c' \\
& \bullet c' \in \text{visibleCols } c t_2 \wedge \text{CS\_posn } c' = n\} \\
& \wedge \text{cleanRows} \\
& c \\
& (Snd (cleanColCons c t_2)) \\
& (TS\_rows t_1) \\
& = \text{cleanRows} \\
& c \\
& (Snd (cleanColCons c t_2)) \\
& (TS\_rows t_2) \\
\Rightarrow & ((\text{RelCombine} \\
& (\text{revealRow } c t_1 \sim \% u) \\
& (\text{ListRel } (TS\_rows t_1)) \\
& \% \text{Graph } (\text{updateRow } c (TS\_class t_2))) \\
& \triangleright \{x | \text{isError } x\}) \\
& \% \text{Graph } \text{destError} \\
& = \{\} \\
\Leftrightarrow & ((\text{RelCombine} \\
& (\text{revealRow } c t_2 \sim \% u) \\
& (\text{ListRel } (TS\_rows t_2)) \\
& \% \text{Graph } (\text{updateRow } c (TS\_class t_2))) \\
& \triangleright \{x | \text{isError } x\}) \\
& \% \text{Graph } \text{destError} \\
& = \{\})
\end{aligned}$$

**fun\_updateRow\_thm**

$$\vdash \forall c t \bullet \text{Graph } (\text{updateRow } c (TS\_class t)) \in \text{Functional}$$

**fun\_destError\_thm**

$$\vdash \text{Graph } \text{destError} \in \text{Functional}$$

**fin\_lemma1**

$$\begin{aligned}
& \vdash \forall l r c t u \\
& \bullet u \in \text{Functional} \\
& \Rightarrow ((\text{RelCombine} \\
& (\text{Squash} \\
& (\text{Id} \\
& (\text{Dom} \\
& (\text{ListRel } l
\end{aligned}$$

---


$$\begin{aligned} & \triangleright \{r \\ & \quad |c \\ & \quad \text{dominates } R\_exist \\ & \quad r\}) \sim \\ & \quad \% u) \\ & \quad (ListRel\ l) \\ & \quad \% Graph\ (updateRow\ c\ (TS\_class\ t)) \\ & \quad \triangleright \{x|isError\ x\} \\ & \quad \% Graph\ destError \\ & \quad \in\ Finite \\ \mathbf{fin\_lemma2} \quad & \vdash \forall\ l\ last\ r\ c\ t\ u \\ & \quad \bullet\ u \in\ Functional \\ & \quad \Rightarrow ((RelCombine \\ & \quad \quad (\{(\# \\ & \quad \quad \quad (Squash \\ & \quad \quad \quad \quad (Id \\ & \quad \quad \quad \quad \quad (Dom \\ & \quad \quad \quad \quad \quad \quad (ListRel\ l\ \triangleright\ \{r|c\ \text{dominates}\ R\_exist\ r\}\})) \\ & \quad \quad \quad \quad \quad \quad \quad +\ 1,\ \#\ l + 1\}) \sim \\ & \quad \quad \quad \% u) \\ & \quad \quad \quad \{(\#\ l + 1,\ last)\} \\ & \quad \quad \% Graph\ (updateRow\ c\ (TS\_class\ t)) \\ & \quad \quad \triangleright \{x|isError\ x\} \\ & \quad \% Graph\ destError \\ & \quad \in\ Finite \\ \mathbf{cleanTable\_updateQuery\_lemma} \quad & \vdash \forall\ c\ t_1\ t_2\ s_1\ s_2\ u \\ & \quad \bullet\ c\ \text{dominates}\ TS\_class\ t_1 \\ & \quad \quad \wedge\ c\ \text{dominates}\ TS\_class\ t_2 \\ & \quad \quad \wedge\ cleanTable\ c\ t_1 = cleanTable\ c\ t_2 \\ & \quad \Rightarrow Snd\ (updateQuery\ (c,\ u,\ s_1,\ t_1)) \\ & \quad \quad = Snd\ (updateQuery\ (c,\ u,\ s_2,\ t_2)) \\ \mathbf{tabExists\_cleanTable\_lemma} \quad & \vdash \forall\ c\ s_1\ s_2 \\ & \quad \bullet\ hide\ (c,\ s_1) = hide\ (c,\ s_2) \\ & \quad \Rightarrow (\forall\ i \\ & \quad \bullet\ tabExists\ c\ i\ (repState\ s_1) \\ & \quad \quad \Rightarrow cleanTable\ c\ (getTable\ i\ (repState\ s_1)) \\ & \quad \quad \quad = cleanTable \\ & \quad \quad \quad \quad c \\ & \quad \quad \quad \quad (getTable\ i\ (repState\ s_2)) \\ & \quad \quad \wedge\ (c \\ & \quad \quad \quad \text{dominates}\ TS\_class \\ & \quad \quad \quad \quad (getTable\ i\ (repState\ s_1)) \\ & \quad \quad \Leftrightarrow c \\ & \quad \quad \quad \text{dominates}\ TS\_class \\ & \quad \quad \quad \quad (getTable\ i\ (repState\ s_2)))) \\ \mathbf{conjunct3} \quad & \vdash \forall\ c\ s_1\ s_2\ e \end{aligned}$$


---

- $hide (c, s_1) = hide (c, s_2)$   
 $\Rightarrow Snd (updateState (c, e, s_1))$   
 $= Snd (updateState (c, e, s_2))$

## 16 THE THEORY fef012

### 16.1 Parents

*fef011*

### 16.2 Children

*fef013*

### 16.3 Theorems

**destItem\_consistent**

**destClass\_consistent**

**destData\_consistent**

- ⊢ *Consistent*  
 $(\lambda (destItem', destClass', destData')$   
  - $\forall i c d$
  - $destItem' (ItemUpdate i) = i$   
 $\wedge destClass' (ClassUpdate c) = c$   
 $\wedge destData' (DataUpdate d) = d)$

**¬giveVal\_eq\_giveError\_thm**

- ⊢  $\forall v e \bullet \neg giveVal v = giveError e$

**¬giveError\_eq\_giveVal\_thm**

- ⊢  $\forall e v \bullet \neg giveError e = giveVal v$

**rel\_combine\_one\_lemma**

- ⊢  $\forall l last s u$ 
  - *RelCombine*  
 $(Squash (Id (Dom (ListRel l \triangleright s))) \sim \frac{2}{3} u)$   
 $(ListRel (l @ [last]))$   
 $= RelCombine$   
 $(Squash (Id (Dom (ListRel l \triangleright s))) \sim \frac{2}{3} u)$   
 $(ListRel l)$

**rel\_combine\_null\_lemma**

- ⊢  $\forall l last c t us$ 
  - $((RelCombine$   
 $(Squash$   
 $(Id$   
 $(Dom$   
 $(ListRel (l @ [last])$   
 $\triangleright \{r$   
 $|c$   
 $dominates R\_exist$

$$\begin{aligned}
& r}})) \sim \\
& \quad \% us) \\
& \quad (ListRel (l @ [last])) \\
& \quad \% Graph (updateRow c (TS\_class t))) \\
& \quad \triangleright \{x|isError x\} \\
& \quad \% Graph destError \\
& = \{\} \\
\Rightarrow & ((RelCombine \\
& \quad (Squash \\
& \quad \quad (Id \\
& \quad \quad \quad (Dom \\
& \quad \quad \quad \quad (ListRel l \\
& \quad \quad \quad \quad \quad \triangleright \{r \\
& \quad \quad \quad \quad \quad |c \\
& \quad \quad \quad \quad \quad \quad dominates R\_exist \\
& \quad \quad \quad \quad \quad \quad r}})) \sim
\end{aligned}$$

$$\begin{aligned}
& \% us) \\
& (ListRel l) \\
& \% Graph (updateRow c (TS\_class t))) \\
& \triangleright \{x|isError x\} \\
& \% Graph destError \\
& = \{\}
\end{aligned}$$

**dom\_rel\_combine\_null\_⊆\_lemma** $\vdash \forall l s u c t$ • *Dom*

$$\begin{aligned}
& ((RelCombine \\
& \quad (Squash \\
& \quad \quad (Id \\
& \quad \quad \quad (Dom \\
& \quad \quad \quad \quad (ListRel l \\
& \quad \quad \quad \quad \quad \triangleright \{r \\
& \quad \quad \quad \quad \quad |c \\
& \quad \quad \quad \quad \quad \quad dominates R\_exist \\
& \quad \quad \quad \quad \quad \quad r}})) \sim
\end{aligned}$$

$$\begin{aligned}
& \% u) \\
& (ListRel l) \\
& \% Graph (updateRow c (TS\_class t))) \\
& \% Graph destVal) \\
& \subseteq Dom (ListRel l)
\end{aligned}$$

**destVal\_fun\_thm** $\vdash Graph destVal \in Functional$ **updateRow\_fun\_thm** $\vdash \forall c t \bullet Graph (updateRow c (TS\_class t)) \in Functional$ **conjunct1\_fun\_lemma** $\vdash \forall l s u c t$ •  $u \in Functional$ 

$$\Rightarrow (RelCombine \\
\quad (Squash$$

```

      (Id
        (Dom
          (ListRel l
            ▷ {r
              |c
                dominates R_exist
                r}})) ~
          % u)
      (ListRel l)
      % Graph (updateRow c (TS_class t))
      % Graph destVal
      ∈ Functional
conjunct1_lemma1
      ⊢ ∀ l last c u t
      • u ∈ Functional
      ⇒ RelList
      (ListRel (l @ [last])
        ⊕ (RelCombine
          (Squash
            (Id
              (Dom
                (ListRel l
                  ▷ {r
                    |c
                      dominates R_exist
                      r}})) ~
                % u)
              (ListRel l)
              % Graph (updateRow c (TS_class t))
              % Graph destVal)
            = RelList
              (ListRel l
                ⊕ (RelCombine
                  (Squash
                    (Id
                      (Dom
                        (ListRel l
                          ▷ {r
                            |c
                              dominates R_exist
                              r}})) ~
                        % u)
                      (ListRel l)
                      % Graph (updateRow c (TS_class t))
                      % Graph destVal)
                    @ [last]

```

**conjunct1\_lemma2**

```

      ⊢ ∀ c last l u us t
      • us ∈ Functional

```

$$\begin{aligned}
 & \wedge c \text{ dominates } R\_exist \text{ last} \\
 & \wedge (\# \\
 & \quad (Squash \\
 & \quad \quad (Id \\
 & \quad \quad \quad (Dom \\
 & \quad \quad \quad \quad (ListRel (l @ [last]) \\
 & \quad \quad \quad \quad \triangleright \{r \\
 & \quad \quad \quad \quad |c \\
 & \quad \quad \quad \quad \quad \text{dominates } R\_exist \\
 & \quad \quad \quad \quad \quad \quad r\}))))), u) \\
 & \in us \\
 \Rightarrow RelList \\
 & \quad (ListRel (l @ [last]) \\
 & \quad \oplus (RelCombine \\
 & \quad \quad (Squash \\
 & \quad \quad \quad (Id \\
 & \quad \quad \quad \quad (Dom \\
 & \quad \quad \quad \quad \quad (ListRel (l @ [last]) \\
 & \quad \quad \quad \quad \quad \triangleright \{r \\
 & \quad \quad \quad \quad \quad |c \\
 & \quad \quad \quad \quad \quad \quad \text{dominates } R\_exist \\
 & \quad \quad \quad \quad \quad \quad \quad r\})))))) \sim \\
 & \quad \quad \quad \quad \quad \quad \quad \quad \% us) \\
 & \quad \quad \quad \quad \quad \quad \quad (ListRel (l @ [last])) \\
 & \quad \quad \quad \quad \quad \quad \quad \% Graph (updateRow c (TS\_class t))) \\
 & \quad \quad \quad \quad \quad \quad \quad \% Graph destVal) \\
 = RelList \\
 & \quad (ListRel l \\
 & \quad \oplus (RelCombine \\
 & \quad \quad (Squash \\
 & \quad \quad \quad (Id \\
 & \quad \quad \quad \quad (Dom \\
 & \quad \quad \quad \quad \quad (ListRel l \\
 & \quad \quad \quad \quad \quad \triangleright \{r \\
 & \quad \quad \quad \quad \quad |c \\
 & \quad \quad \quad \quad \quad \quad \text{dominates } R\_exist \text{ } r\})))))) \sim \\
 & \quad \quad \quad \quad \quad \quad \quad \quad \% us) \\
 & \quad \quad \quad \quad \quad \quad \quad (ListRel l) \\
 & \quad \quad \quad \quad \quad \quad \quad \% Graph (updateRow c (TS\_class t))) \\
 & \quad \quad \quad \quad \quad \quad \quad \% Graph destVal) \\
 & \quad @ [destVal \\
 & \quad \quad (updateRow c (TS\_class t) (u, last))]
 \end{aligned}$$

**insertRows\_lemma**

$$\begin{aligned}
 & \vdash \forall c_1 \ c_2 \ t \ ds \\
 & \bullet \neg c_2 \text{ dominates } c_1 \\
 & \quad \Rightarrow cleanTable \ c_2 \ t \\
 & \quad = cleanTable \\
 & \quad \quad c_2
 \end{aligned}$$

$$\begin{aligned}
& (\text{replaceRows} \\
& \quad t \\
& \quad (\text{TS\_rows } t \\
& \quad \quad @ \text{Map} \\
& \quad \quad (\text{MkRow } c_1 \text{ o colDefaults } c_1 t) \\
& \quad \quad ds))
\end{aligned}$$
**insertQuery\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 s i ds \\
& \quad \bullet \neg c_2 \text{ dominates } c_1 \\
& \quad \wedge \text{tabExists} \\
& \quad \quad c_1 \\
& \quad \quad (\text{tabFromEffect } (\text{InsertEffect } (i, ds))) \\
& \quad \quad (\text{repState } s) \\
& \Rightarrow \text{hideR } (c_2, \text{repState } s) \\
& = \text{hideR} \\
& \quad (c_2, \\
& \quad \quad \text{Fst} \\
& \quad \quad (\text{insertQuery} \\
& \quad \quad \quad (c_1, \\
& \quad \quad \quad \quad \text{destInsert} \\
& \quad \quad \quad \quad (\text{InsertEffect } (i, ds)), \\
& \quad \quad \quad \quad \text{repState } s, \\
& \quad \quad \quad \quad \text{getTable} \\
& \quad \quad \quad \quad (\text{tabFromEffect} \\
& \quad \quad \quad \quad \quad (\text{InsertEffect} \\
& \quad \quad \quad \quad \quad \quad (i, ds))) \\
& \quad \quad \quad \quad (\text{repState } s))))))
\end{aligned}$$
**deleteRows\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 s t ns \\
& \quad \bullet \neg c_2 \text{ dominates } c_1 \\
& \quad \Rightarrow \text{cleanTable } c_2 t \\
& \quad = \text{cleanTable} \\
& \quad \quad c_2 \\
& \quad \quad (\text{replaceRows} \\
& \quad \quad \quad t \\
& \quad \quad \quad (\text{Extract} \\
& \quad \quad \quad \quad (1 .. \# (\text{TS\_rows } t) \\
& \quad \quad \quad \quad \quad \setminus \text{revealRow } c_1 t \text{ Image } ns \\
& \quad \quad \quad \quad \quad \cap \{i \\
& \quad \quad \quad \quad \quad \quad |R\_exist (\text{Nth } (\text{TS\_rows } t) i) \\
& \quad \quad \quad \quad \quad \quad = c_1\}) \\
& \quad \quad \quad \quad (\text{TS\_rows } t)))
\end{aligned}$$
**deleteQuery\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 s i ns \\
& \quad \bullet \neg c_2 \text{ dominates } c_1 \\
& \quad \wedge \text{tabExists} \\
& \quad \quad c_1 \\
& \quad \quad (\text{tabFromEffect } (\text{DeleteEffect } (i, ns)))
\end{aligned}$$

$$\begin{aligned}
& (\text{repState } s) \\
\Rightarrow & \text{hideR } (c_2, \text{repState } s) \\
= & \text{hideR} \\
& (c_2, \\
& \quad \text{deleteQuery} \\
& \quad (c_1, \\
& \quad \quad \text{destDelete} \\
& \quad \quad (\text{DeleteEffect } (i, ns)), \\
& \quad \quad \text{repState } s, \\
& \quad \quad \text{getTable} \\
& \quad \quad (\text{tabFromEffect} \\
& \quad \quad \quad (\text{DeleteEffect } (i, ns)))) \\
& \quad (\text{repState } s)))
\end{aligned}$$
**replaceData\_updateField\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 d tc u \\
& \bullet c_1 \text{ dominates } tc \wedge c_2 \text{ dominates } tc \\
& \Rightarrow \neg c_2 \text{ dominates } c_1 \\
& \quad \wedge \text{isVal } (\text{updateField } c_1 tc (u, d)) \\
& \Rightarrow \text{replaceData } c_2 d \\
& = \text{replaceData} \\
& \quad c_2 \\
& \quad (\text{destVal } (\text{updateField } c_1 tc (u, d)))
\end{aligned}$$
**cleanRow\_updateRow\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 r t u \\
& \bullet c_1 \text{ dominates } \text{TS\_class } t \wedge c_2 \text{ dominates } \text{TS\_class } t \\
& \Rightarrow \neg c_2 \text{ dominates } c_1 \\
& \quad \wedge \text{isVal } (\text{updateRow } c_1 (\text{TS\_class } t) (u, r)) \\
& \Rightarrow \text{cleanRow } c_2 (\text{Snd } (\text{cleanColCons } c_2 t)) r \\
& = \text{cleanRow} \\
& \quad c_2 \\
& \quad (\text{Snd } (\text{cleanColCons } c_2 t)) \\
& \quad (\text{destVal } (\text{updateRow } c_1 (\text{TS\_class } t) (u, r)))
\end{aligned}$$
**updateRows\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 s t us \\
& \bullet us \in \text{Functional} \\
& \quad \wedge \neg c_2 \text{ dominates } c_1 \\
& \quad \wedge c_1 \text{ dominates } \text{TS\_class } t \\
& \quad \wedge ((\text{RelCombine} \\
& \quad \quad (\text{revealRow } c_1 t \sim \text{\% } us) \\
& \quad \quad (\text{ListRel } (\text{TS\_rows } t)) \\
& \quad \quad \text{\% } \text{Graph } (\text{updateRow } c_1 (\text{TS\_class } t))) \\
& \quad \quad \triangleright \{x \mid \text{isError } x\} \\
& \quad \quad \text{\% } \text{Graph } \text{destError} \\
& \quad = \{\}) \\
& \Rightarrow \text{cleanTable } c_2 t \\
& = \text{cleanTable} \\
& \quad c_2 \\
& \quad (\text{replaceRows}
\end{aligned}$$

$$\begin{aligned}
& t \\
& (\text{RelList} \\
& \quad (\text{ListRel} (\text{TS\_rows } t) \\
& \quad \oplus (\text{RelCombine} \\
& \quad \quad (\text{revealRow } c_1 \ t \ \sim \ \% \ us) \\
& \quad \quad (\text{ListRel} (\text{TS\_rows } t)) \\
& \quad \quad \% \ \text{Graph} \\
& \quad \quad (\text{updateRow } c_1 \ (\text{TS\_class } t))) \\
& \quad \% \ \text{Graph } \text{destVal}))
\end{aligned}$$
**updateQuery\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 \ c_2 \ s \ i \ us \\
& \quad \bullet \neg c_2 \ \text{dominates } c_1 \\
& \quad \wedge \text{tabExists} \\
& \quad \quad c_1 \\
& \quad \quad (\text{tabFromEffect} (\text{UpdateEffect} (i, us))) \\
& \quad \quad (\text{repState } s) \\
& \quad \wedge c_1 \\
& \quad \quad \text{dominates } \text{TS\_class} \\
& \quad \quad (\text{getTable} \\
& \quad \quad \quad (\text{tabFromEffect} (\text{UpdateEffect} (i, us))) \\
& \quad \quad \quad (\text{repState } s)) \\
& \Rightarrow \text{hideR} (c_2, \text{repState } s) \\
& \quad = \text{hideR} \\
& \quad \quad (c_2, \\
& \quad \quad \quad \text{Fst} \\
& \quad \quad \quad (\text{updateQuery} \\
& \quad \quad \quad \quad (c_1, \\
& \quad \quad \quad \quad \quad \text{destUpdate} \\
& \quad \quad \quad \quad \quad \quad (\text{UpdateEffect} (i, us)), \\
& \quad \quad \quad \quad \quad \quad \text{repState } s, \\
& \quad \quad \quad \quad \quad \quad \text{getTable} \\
& \quad \quad \quad \quad \quad \quad \quad (\text{tabFromEffect} \\
& \quad \quad \quad \quad \quad \quad \quad \quad (\text{UpdateEffect} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad (i, us))) \\
& \quad \quad \quad \quad \quad \quad \quad (\text{repState } s))))
\end{aligned}$$
**conjunct1**

$$\begin{aligned}
& \vdash \forall c_1 \ c_2 \ s \ e \\
& \quad \bullet \neg \text{hideR} (c_2, \text{repState } s) \\
& \quad \quad = \text{hideR} \\
& \quad \quad \quad (c_2, \\
& \quad \quad \quad \quad \text{Fst} \\
& \quad \quad \quad \quad (\text{updateStateR} \\
& \quad \quad \quad \quad \quad (c_1, e, \text{repState } s))) \\
& \Rightarrow c_2 \ \text{dominates } c_1
\end{aligned}$$

## 17 THE THEORY fef013

### 17.1 Parents

*fef012*

### 17.2 Children

*fef015*

### 17.3 Theorems

#### **cleanTable\_insertRows\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 t_1 t_2 ds \\
& \bullet \text{cleanTable } c_1 t_1 = \text{cleanTable } c_1 t_2 \\
& \quad \wedge c_1 \text{ dominates } c_2 \\
& \Rightarrow \text{cleanTable} \\
& \quad c_1 \\
& \quad (\text{replaceRows} \\
& \quad \quad t_1 \\
& \quad \quad (\text{TS\_rows } t_1 \\
& \quad \quad \quad @ \text{Map} \\
& \quad \quad \quad (\text{MkRow } c_2 \text{ o colDefaults } c_2 t_1) \\
& \quad \quad \quad ds)) \\
& = \text{cleanTable} \\
& \quad c_1 \\
& \quad (\text{replaceRows} \\
& \quad \quad t_2 \\
& \quad \quad (\text{TS\_rows } t_2 \\
& \quad \quad \quad @ \text{Map} \\
& \quad \quad \quad (\text{MkRow } c_2 \text{ o colDefaults } c_2 t_2) \\
& \quad \quad \quad ds))
\end{aligned}$$

#### **extract\_∧\_single\_lemma**

$$\begin{aligned}
& \vdash \forall c l x \\
& \bullet \text{Extract} \\
& \quad (1 .. \# (l @ [x]) \\
& \quad \quad \backslash \text{Squash} \\
& \quad \quad \quad (\text{Id} \\
& \quad \quad \quad \quad (\text{Dom} \\
& \quad \quad \quad \quad \quad (\text{ListRel } (l @ [x]) \\
& \quad \quad \quad \quad \quad \quad \triangleright \{r \\
& \quad \quad \quad \quad \quad \quad \quad |c \\
& \quad \quad \quad \quad \quad \quad \quad \text{dominates } R\_exist \\
& \quad \quad \quad \quad \quad \quad \quad \quad r\}))))) \\
& \quad \quad \quad \text{Image ns} \\
& \quad \quad \quad \cap \{i | R\_exist (Nth (l @ [x]) i) = c\} \\
& \quad \quad \quad (l @ [x])
\end{aligned}$$

---

```

= (if
  # l + 1
  ∈ 1 .. # (l @ [x])
  \ Squash
  (Id
    (Dom
      (ListRel (l @ [x])
        ▷ {r
          |c
            dominates R_exist
              r}))))
  Image ns
  ∩ {i|R_exist (Nth (l @ [x]) i) = c})
then
  Extract
  (1 .. # l
  \ Squash
  (Id
    (Dom
      (ListRel l
        ▷ {r
          |c
            dominates R_exist
              r}))))
  Image ns
  ∩ {i|R_exist (Nth l i) = c})
  l
  @ [x]
else
  Extract
  (1 .. # l
  \ Squash
  (Id
    (Dom
      (ListRel l
        ▷ {r
          |c
            dominates R_exist
              r}))))
  Image ns
  ∩ {i|R_exist (Nth l i) = c})
  l)

```

**map\_cleanRow\_lemma1**

```

⊢ ∀ l1 l2 c1 c2 s
  • c1 dominates c2
  ∧ Map
  (cleanRow c1 s)
  (l1 ↑ {r|c1 dominates R_exist r})

```

$$\begin{aligned}
&= \text{Map} \\
&\quad (\text{cleanRow } c_1 \ s) \\
&\quad (l_2 \upharpoonright \{r \mid c_1 \text{ dominates } R\_exist \ r\}) \\
\Rightarrow &\text{Map} \\
&\quad (\text{cleanRow } c_2 \ s) \\
&\quad (l_1 \upharpoonright \{r \mid c_2 \text{ dominates } R\_exist \ r\}) \\
&= \text{Map} \\
&\quad (\text{cleanRow } c_2 \ s) \\
&\quad (l_2 \upharpoonright \{r \mid c_2 \text{ dominates } R\_exist \ r\})
\end{aligned}$$
**map\_cleanRow\_lemma2**

$$\begin{aligned}
&\vdash \forall l_1 \ l_2 \ c_1 \ c_2 \ s \\
&\quad \bullet \ c_1 \text{ dominates } c_2 \\
&\quad \wedge \text{Map} \\
&\quad\quad (\text{cleanRow } c_1 \ s) \\
&\quad\quad (l_1 \upharpoonright \{r \mid c_1 \text{ dominates } R\_exist \ r\}) \\
&= \text{Map} \\
&\quad (\text{cleanRow } c_1 \ s) \\
&\quad (l_2 \upharpoonright \{r \mid c_1 \text{ dominates } R\_exist \ r\}) \\
\Rightarrow &\# (\text{ListRel } l_1 \triangleright \{r \mid c_2 \text{ dominates } R\_exist \ r\}) \\
&= \# (\text{ListRel } l_2 \triangleright \{r \mid c_2 \text{ dominates } R\_exist \ r\})
\end{aligned}$$
**cleanTable\_deleteRows\_lemma**

$$\begin{aligned}
&\vdash \forall c_1 \ c_2 \ t_1 \ t_2 \ ns \\
&\quad \bullet \ \text{cleanTable } c_1 \ t_1 = \text{cleanTable } c_1 \ t_2 \\
&\quad \wedge \ c_1 \text{ dominates } c_2 \\
&\quad \wedge \ c_2 \text{ dominates } TS\_class \ t_1 \\
\Rightarrow &\text{cleanTable} \\
&\quad c_1 \\
&\quad (\text{replaceRows} \\
&\quad\quad t_1 \\
&\quad\quad (\text{Extract} \\
&\quad\quad\quad (1 \ .. \ \# (TS\_rows \ t_1) \\
&\quad\quad\quad\quad \setminus \text{revealRow } c_2 \ t_1 \ \text{Image } ns \\
&\quad\quad\quad\quad \cap \{i \\
&\quad\quad\quad\quad\quad | R\_exist \ (Nth \ (TS\_rows \ t_1) \ i) \\
&\quad\quad\quad\quad\quad = c_2\}) \\
&\quad\quad\quad (TS\_rows \ t_1))) \\
&= \text{cleanTable} \\
&\quad c_1 \\
&\quad (\text{replaceRows} \\
&\quad\quad t_2 \\
&\quad\quad (\text{Extract} \\
&\quad\quad\quad (1 \ .. \ \# (TS\_rows \ t_2) \\
&\quad\quad\quad\quad \setminus \text{revealRow } c_2 \ t_2 \ \text{Image } ns \\
&\quad\quad\quad\quad \cap \{i \\
&\quad\quad\quad\quad\quad | R\_exist \ (Nth \ (TS\_rows \ t_2) \ i) \\
&\quad\quad\quad\quad\quad = c_2\}) \\
&\quad\quad\quad (TS\_rows \ t_2)))
\end{aligned}$$
**replaceData\_updateField\_lemma**

$$\begin{aligned}
& \vdash \forall c_1 c_2 d_1 d_2 t u \\
& \bullet c_1 \text{ dominates } c_2 \\
& \quad \wedge \text{replaceData } c_1 d_1 = \text{replaceData } c_1 d_2 \\
& \quad \wedge \text{isVal } (\text{updateField } c_2 \text{ (TS\_class } t) (u, d_1)) \\
& \quad \wedge \text{isVal } (\text{updateField } c_2 \text{ (TS\_class } t) (u, d_2)) \\
& \Rightarrow \text{replaceData} \\
& \quad c_1 \\
& \quad (\text{destVal} \\
& \quad \quad (\text{updateField } c_2 \text{ (TS\_class } t) (u, d_1))) \\
& = \text{replaceData} \\
& \quad c_1 \\
& \quad (\text{destVal} \\
& \quad \quad (\text{updateField } c_2 \text{ (TS\_class } t) (u, d_2)))
\end{aligned}$$
**cleanRow\_updateRow\_lemma1**

$$\begin{aligned}
& \vdash \forall c_1 c_2 r_1 r_2 t u \\
& \bullet c_1 \text{ dominates } c_2 \\
& \quad \wedge \text{Dom } u \\
& \quad \subseteq \{n\} \\
& \quad |\exists c' \\
& \quad \bullet c' \in \text{visibleCols } c_2 t \wedge \text{CS\_posn } c' = n\} \\
& \quad \wedge \text{cleanRow } c_1 (\text{Snd } (\text{cleanColCons } c_1 t)) r_1 \\
& \quad = \text{cleanRow } c_1 (\text{Snd } (\text{cleanColCons } c_1 t)) r_2 \\
& \quad \wedge \text{isVal } (\text{updateRow } c_2 \text{ (TS\_class } t) (u, r_1)) \\
& \quad \wedge \text{isVal } (\text{updateRow } c_2 \text{ (TS\_class } t) (u, r_2)) \\
& \Rightarrow \text{cleanRow} \\
& \quad c_1 \\
& \quad (\text{Snd } (\text{cleanColCons } c_1 t)) \\
& \quad (\text{destVal} \\
& \quad \quad (\text{updateRow } c_2 \text{ (TS\_class } t) (u, r_1))) \\
& = \text{cleanRow} \\
& \quad c_1 \\
& \quad (\text{Snd } (\text{cleanColCons } c_1 t)) \\
& \quad (\text{destVal} \\
& \quad \quad (\text{updateRow } c_2 \text{ (TS\_class } t) (u, r_2)))
\end{aligned}$$
**inUpdates\_lemma**

$$\begin{aligned}
& \vdash \forall l_1 l_2 x_1 x_2 c us up \\
& \bullet R\_exist x_1 = R\_exist x_2 \\
& \quad \wedge \# (\text{ListRel } l_1 \triangleright \{r|c \text{ dominates } R\_exist r\}) \\
& \quad = \# (\text{ListRel } l_2 \triangleright \{r|c \text{ dominates } R\_exist r\}) \\
& \Rightarrow ((\# \\
& \quad (\text{Squash} \\
& \quad \quad (\text{Id} \\
& \quad \quad \quad (\text{Dom} \\
& \quad \quad \quad \quad (\text{ListRel } (l_1 @ [x_1]) \\
& \quad \quad \quad \quad \triangleright \{r \\
& \quad \quad \quad \quad |c \\
& \quad \quad \quad \quad \text{dominates } R\_exist \\
& \quad \quad \quad \quad r\}))))), up)
\end{aligned}$$

$$\begin{aligned} & \in us \\ \Leftrightarrow & (\# \\ & \quad (\text{Squash} \\ & \quad \quad (\text{Id} \\ & \quad \quad \quad (\text{Dom} \\ & \quad \quad \quad \quad (\text{ListRel } (l_2 \text{ @ } [x_2]) \\ & \quad \quad \quad \quad \triangleright \{r \\ & \quad \quad \quad \quad \quad |c \\ & \quad \quad \quad \quad \quad \quad \text{dominates } R\_exist \\ & \quad \quad \quad \quad \quad \quad \quad r\}))))), up) \end{aligned}$$

$$\in us)$$
**cleanTable\_updateRows\_lemma**

$$\begin{aligned} & \vdash \forall c_1 c_2 t_1 t_2 us \\ & \bullet us \in \text{Functional} \\ & \quad \wedge \text{Dom } (\bigcup (\text{Ran } us)) \\ & \quad \subseteq \{n \\ & \quad \quad | \exists c \\ & \quad \quad \bullet c \in \text{Snd } (\text{cleanColCons } c_2 t_2) \\ & \quad \quad \quad \wedge \text{CS\_posn } c = n\} \\ & \quad \wedge ((\text{RelCombine} \\ & \quad \quad (\text{revealRow } c_2 t_1 \sim \% us) \\ & \quad \quad (\text{ListRel } (\text{TS\_rows } t_1)) \\ & \quad \quad \% \text{Graph } (\text{updateRow } c_2 (\text{TS\_class } t_2))) \\ & \quad \quad \triangleright \{x | \text{isError } x\} \\ & \quad \quad \% \text{Graph } \text{destError} \\ & \quad \quad = \{\} \\ & \quad \wedge ((\text{RelCombine} \\ & \quad \quad (\text{revealRow } c_2 t_2 \sim \% us) \\ & \quad \quad (\text{ListRel } (\text{TS\_rows } t_2)) \\ & \quad \quad \% \text{Graph } (\text{updateRow } c_2 (\text{TS\_class } t_2))) \\ & \quad \quad \triangleright \{x | \text{isError } x\} \\ & \quad \quad \% \text{Graph } \text{destError} \\ & \quad \quad = \{\} \\ & \quad \wedge \text{cleanTable } c_1 t_1 = \text{cleanTable } c_1 t_2 \\ & \quad \wedge c_1 \text{ dominates } c_2 \\ & \quad \wedge c_2 \text{ dominates } \text{TS\_class } t_1 \\ & \Rightarrow \text{cleanTable} \\ & \quad c_1 \\ & \quad (\text{replaceRows} \\ & \quad \quad t_1 \\ & \quad \quad (\text{RelList} \\ & \quad \quad \quad (\text{ListRel } (\text{TS\_rows } t_1)) \\ & \quad \quad \quad \oplus (\text{RelCombine} \\ & \quad \quad \quad \quad (\text{revealRow } c_2 t_1 \sim \% us) \\ & \quad \quad \quad \quad (\text{ListRel } (\text{TS\_rows } t_1)) \\ & \quad \quad \quad \% \text{Graph} \\ & \quad \quad \quad \quad (\text{updateRow} \\ & \quad \quad \quad \quad \quad c_2 \end{aligned}$$

$$\begin{aligned}
& (TS\_class\ t_1))) \\
& \quad \% Graph\ destVal))) \\
= & cleanTable \\
& \quad c_1 \\
& \quad (replaceRows \\
& \quad \quad t_2 \\
& \quad \quad (RelList \\
& \quad \quad \quad (ListRel\ (TS\_rows\ t_2) \\
& \quad \quad \quad \oplus\ (RelCombine \\
& \quad \quad \quad \quad (revealRow\ c_2\ t_2\ \sim\ \% us) \\
& \quad \quad \quad \quad (ListRel\ (TS\_rows\ t_2)) \\
& \quad \quad \quad \% Graph \\
& \quad \quad \quad (updateRow \\
& \quad \quad \quad \quad c_2 \\
& \quad \quad \quad \quad (TS\_class\ t_2))) \\
& \quad \quad \% Graph\ destVal)))
\end{aligned}$$
**tabExists\_lemma1**

$$\begin{aligned}
& \vdash \forall c_1\ c_2\ i\ s \\
& \quad \bullet\ c_1\ dominates\ c_2 \wedge tabExists\ c_2\ i\ s \\
& \quad \Rightarrow tabExists\ c_1\ i\ s
\end{aligned}$$
**tabExists\_cleanTable\_lemma1**

$$\begin{aligned}
& \vdash \forall c\ s_1\ s_2 \\
& \quad \bullet\ hideR\ (c,\ repState\ s_1) = hideR\ (c,\ repState\ s_2) \\
& \quad \Rightarrow (\forall i \\
& \quad \bullet\ tabExists\ c\ i\ (repState\ s_1) \\
& \quad \quad \Rightarrow cleanTable\ c\ (getTable\ i\ (repState\ s_1)) \\
& \quad \quad = cleanTable\ c\ (getTable\ i\ (repState\ s_2)))
\end{aligned}$$
**colDefaults\_lemma**

$$\begin{aligned}
& \vdash \forall c\ t_1\ t_2 \\
& \quad \bullet\ c\ dominates\ TS\_class\ t_1 \\
& \quad \quad \wedge\ c\ dominates\ TS\_class\ t_2 \\
& \quad \quad \wedge\ cleanTable\ c\ t_1 = cleanTable\ c\ t_2 \\
& \quad \Rightarrow colDefaults\ c\ t_1 = colDefaults\ c\ t_2
\end{aligned}$$
**conjunct2**

$$\begin{aligned}
& \vdash \forall c_1\ c_2\ s_1\ s_2\ e \\
& \quad \bullet\ hideR\ (c_1,\ repState\ s_1) \\
& \quad \quad = hideR\ (c_1,\ repState\ s_2) \\
& \quad \quad \wedge\ c_1\ dominates\ c_2 \\
& \quad \Rightarrow hideR \\
& \quad \quad (c_1, \\
& \quad \quad \quad Fst \\
& \quad \quad \quad (updateStateR \\
& \quad \quad \quad \quad (c_2,\ e,\ repState\ s_1))) \\
= & hideR \\
& \quad (c_1, \\
& \quad \quad Fst \\
& \quad \quad (updateStateR \\
& \quad \quad \quad (c_2,\ e,\ repState\ s_2)))
\end{aligned}$$

## 18 THE THEORY fef014

### 18.1 Parents

*fef005*

### 18.2 Children

*fef006*

### 18.3 Constants

**destNullItem**  $Item \rightarrow Maybe$   
**destValuedItem**  $Item \rightarrow ValuedItem$   
**isNullItem**  $Item \rightarrow Bool$   
**isValuedItem**  $Item \rightarrow Bool$   
**destClassVal**  $Val \rightarrow Class$   
**destCodeVal**  $Val \rightarrow Code$   
**destIntervalVal**  $Val \rightarrow Interval$   
**destTimeVal**  $Val \rightarrow Time$   
**destFloatVal**  $Val \rightarrow Float$   
**destIntVal**  $Val \rightarrow Int$   
**destStringVal**  $Val \rightarrow Op$   
**destBoolVal**  $Val \rightarrow Bool$   
**isClassVal**  $Val \rightarrow Bool$   
**isCodeVal**  $Val \rightarrow Bool$   
**isIntervalVal**  $Val \rightarrow Bool$   
**isTimeVal**  $Val \rightarrow Bool$   
**isFloatVal**  $Val \rightarrow Bool$   
**isIntVal**  $Val \rightarrow Bool$   
**isStringVal**  $Val \rightarrow Bool$   
**isBoolVal**  $Val \rightarrow Bool$   
**G\_group**  $GroupedResult \rightarrow Tuple\ LIST$   
**G\_res**  $GroupedResult \rightarrow Data\ LIST$   
**MkGroupedResult**  $Data\ LIST \rightarrow Tuple\ LIST \rightarrow GroupedResult$   
**E\_group**  $Env \rightarrow Tuple\ LIST$   
**E\_row**  $Env \rightarrow Tuple$   
**MkEnv**  $Tuple \rightarrow Tuple\ LIST \rightarrow Env$   
**timeNow**  $Time$   
**userClearance**  $Class$   
**userDirectory**

---

	<i>Col</i>
<b>userName</b>	<i>Op</i>
<b>lubl</b>	<i>Class LIST → Class</i>
<b>glbl</b>	<i>Class LIST → Class</i>
<b>\$dominates_w</b>	<i>Worth → Worth → Bool</i>
<b>\$lub_w</b>	<i>Worth → Worth → Worth</i>
<b>lub_wl</b>	<i>Errors → Worth</i>
<b>Equal</b>	<i>Op</i>
<b>Plus</b>	<i>Op</i>
<b>Or</b>	<i>Op</i>
<b>And</b>	<i>Op</i>
<b>Not</b>	<i>Op</i>
<b>newData</b>	<i>Class → Worth → Val → Data</i>
<b>lub_data</b>	<i>Data LIST → Class</i>
<b>lub_wdata</b>	<i>Data LIST → Worth</i>
<b>ExceptionData</b>	<i>Data LIST → Data</i>
<b>NullData</b>	<i>Data LIST → Data</i>
<b>applyNot</b>	<i>Data LIST → Data</i>
<b>applyAnd</b>	<i>Data LIST → Data</i>
<b>applyOr</b>	<i>Data LIST → Data</i>
<b>\$intPlus</b>	<i>Int → Int → Int</i>
<b>applyPlus</b>	<i>Data LIST → Data</i>
<b>applyEqual</b>	<i>Data LIST → Data</i>
<b>apply</b>	<i>Op → Data LIST → Data</i>
<b>\$*<sub>1</sub></b>	<i>('x → 'y + Errors)</i> <i>→ ('y → 'z + Errors)</i> <i>→ 'x</i> <i>→ 'z + Errors</i>
<b>\$*<sub>2</sub></b>	<i>'x + Errors → ('x → 'y + Errors) → 'y + Errors</i>
<b>\$*<sub>3</sub></b>	<i>'x + Errors</i> <i>→ ('x → 'y → 'z + Errors)</i> <i>→ 'y</i> <i>→ 'z + Errors</i>
<b>\$*<sub>4</sub></b>	<i>('x → 'y + Errors) → ('y → 'z) → 'x → 'z + Errors</i>
<b>\$*<sub>5</sub></b>	<i>'x + Errors → ('x → 'y) → 'y + Errors</i>
<b>\$*<sub>6</sub></b>	<i>'x + Errors × 'y + Errors</i> <i>→ ('x → 'y → 'z)</i> <i>→ 'z + Errors</i>
<b>\$*<sub>7</sub></b>	<i>'x → ('x → 'y) + Errors → 'y + Errors</i>
<b>\$list_∪</b>	<i>'x LIST → 'x LIST → 'x LIST</i>
<b>\$&amp;<sub>1</sub></b>	<i>'x + Errors → 'x + Errors → 'x LIST + Errors</i>
<b>\$&amp;<sub>2</sub></b>	<i>'x + Errors → 'x LIST + Errors → 'x LIST + Errors</i>
<b>\$&amp;<sub>3</sub></b>	<i>'x LIST + Errors → 'x + Errors → 'x LIST + Errors</i>
<b>\$&amp;<sub>4</sub></b>	<i>'x LIST + Errors → 'x LIST + Errors → 'x LIST + Errors</i>
<b>seq</b>	<i>Worth → 'x → 'x LIST</i>
<b>enseq</b>	<i>(Worth → 'x + Errors) → 'x LIST</i>
<b>promote</b>	<i>('x → 'y + Errors) → 'x LIST → 'y LIST + Errors</i>

---

---

<b>find</b>	$'x \text{ LIST} \rightarrow 'x \rightarrow \text{Worth} + \text{Errors}$
<b>distinct</b>	$'x \text{ LIST} \rightarrow 'x \text{ LIST}$
<b>noErrors</b>	$\text{Errors}$
<b>one_col</b>	$\text{Select} \rightarrow \text{Data LIST} + \text{Errors}$
<b>one_result</b>	$\text{Data LIST} \rightarrow \text{Data} + \text{Errors}$
<b>make_data</b>	$\text{Val} \rightarrow \text{Data}$
<b>fillTab</b>	$\text{Col} \rightarrow \text{Col}$
<b>fillCol</b>	$\text{Col} \rightarrow \text{Col}$
<b>getDir</b>	$\text{State} \rightarrow \text{Col} \rightarrow \text{Directory} + \text{Errors}$
<b>getTab</b>	$\text{Directory} \rightarrow \text{Op} \rightarrow \text{TableSpec} + \text{Errors}$
<b>getColPosn</b>	$\text{TableSpec} \rightarrow \text{Op} \rightarrow \text{Worth} + \text{Errors}$
<b>getData</b>	$\text{Row} \rightarrow \text{Worth} \rightarrow \text{Data} + \text{Errors}$
<b>lookup</b>	$\text{State} \rightarrow \text{Col} \rightarrow \text{TableSpec} + \text{Errors}$
<b>isCleared</b>	$\text{Data} \rightarrow \text{Bool}$
<b>isNotCleared</b>	$\text{Data} \rightarrow \text{Bool}$
<b>checkComplete</b>	$\text{Data LIST} \rightarrow \text{Bool}$
<b>check_where_complete</b>	$\text{Bool} \rightarrow \text{Bool} \rightarrow \text{Data} \rightarrow \text{Data} + \text{Errors}$
<b>resultBool</b>	$\text{Data} \rightarrow \text{Bool} + \text{Errors}$
<b>take_data</b>	$\text{Data} \rightarrow \text{Data} + \text{Errors}$
<b>resultClass</b>	$\text{Data} \rightarrow \text{Class} + \text{Errors}$
<b>extract</b>	$\text{Bool LIST} \rightarrow 'x \text{ LIST} \rightarrow 'x \text{ LIST}$
<b>emptyTuple</b>	$\text{Tuple}$
<b>emptyEnv</b>	$\text{Env}$
<b>\$star1</b>	$\text{Tuple} \rightarrow \text{Tuple} \rightarrow \text{Tuple}$
<b>\$star2</b>	$\text{Tuple} \rightarrow \text{Tuple LIST} \rightarrow \text{Tuple LIST}$
<b>\$star3</b>	$\text{Tuple LIST} \rightarrow \text{Tuple LIST} \rightarrow \text{Tuple LIST}$
<b>jay</b>	$\text{Tuple LIST} \rightarrow \text{Tuple LIST LIST} \rightarrow \text{Tuple LIST}$
<b>join</b>	$\text{Tuple LIST LIST} \rightarrow \text{Tuple LIST}$
<b>\$starstar1</b>	$\text{Tuple} \rightarrow \text{Tuple} \rightarrow \text{Tuple}$
<b>\$starstar2</b>	$\text{Tuple} \rightarrow \text{Tuple LIST} \rightarrow \text{Tuple LIST}$
<b>denote_code</b>	$\text{Code} \rightarrow \text{Value}$
<b>denote_class</b>	$\text{Class} \rightarrow \text{Value}$
<b>denote_interval</b>	$\text{Interval} \rightarrow \text{Value}$
<b>denote_time</b>	$\text{Time} \rightarrow \text{Value}$
<b>denote_float</b>	$\text{Float} \rightarrow \text{Value}$
<b>denote_string</b>	$\text{Op} \rightarrow \text{Value}$
<b>denote_integer</b>	$\text{Int} \rightarrow \text{Value}$
<b>denote_false</b>	$\text{Value}$
<b>denote_true</b>	$\text{Value}$
<b>denote_void</b>	$\text{Value}$
<b>denote_null</b>	$\text{Value}$
<b>classification</b>	$\text{Col\_spec} \rightarrow \text{Value}$

---

---

<b>contents</b>	$Col\_spec \rightarrow Value$
<b>all_binop</b>	$Op \rightarrow Value \rightarrow Tuple\_list \rightarrow Value$
<b>count_all</b>	$Value$
<b>set_func_all</b>	$Op \rightarrow Value \rightarrow Value$
<b>binop</b>	$Op \rightarrow Value \times Value \rightarrow Value$
<b>monop</b>	$Op \rightarrow Value \rightarrow Value$
<b>current_time</b>	$Value$
<b>clearance</b>	$Value$
<b>user</b>	$Value$
<b>denote_col_name</b>	$Op \rightarrow Col\_name$
<b>denote_table_spec</b>	$Col \rightarrow Table\_spec$
<b>denote_col_spec</b>	$Col \rightarrow Col\_spec$
<b>correlate_from</b>	$Op \rightarrow Table\_spec \rightarrow From\_spec$
<b>from</b>	$Table\_spec \rightarrow From\_spec$
<b>set_class_and_value</b>	$Col\_name \rightarrow Value \rightarrow Value \rightarrow Set\_clause$
<b>set_class</b>	$Col\_name \rightarrow Value \rightarrow Set\_clause$
<b>set_value</b>	$Col\_name \rightarrow Value \rightarrow Set\_clause$
<b>evaluate</b>	$Select\_list$ $\rightarrow From\_spec\ LIST$ $\rightarrow Value$ $\rightarrow Col\_spec\ LIST$ $\rightarrow Value$ $\rightarrow Tuple\_list$
<b>distinct_tuples</b>	$Select\_list$ $\rightarrow From\_spec\ LIST$ $\rightarrow Value$ $\rightarrow Col\_spec\ LIST$ $\rightarrow Value$ $\rightarrow Tuple\_list$
<b>all_tuples</b>	$Select\_list$ $\rightarrow From\_spec\ LIST$ $\rightarrow Value$ $\rightarrow Col\_spec\ LIST$ $\rightarrow Value$ $\rightarrow Tuple\_list$
<b>union</b>	$Insert\_list \rightarrow Insert\_list \rightarrow Insert\_list$
<b>tuple</b>	$Classified\_value\ LIST \rightarrow Insert\_list$
<b>insert_tuples</b>	$Tuple\_list \rightarrow Insert\_list$
<b>classify_default</b>	$Value \rightarrow Classified\_value$
<b>classify</b>	$Value \rightarrow Value \rightarrow Classified\_value$

---

---

<b>select_values</b>	<i>Value LIST</i> → <i>Select_list</i>
<b>all_columns</b>	<i>Select_list</i>
<b>select</b>	<i>Tuple_list</i> → <i>Query</i>
<b>update</b>	<i>From_spec</i> → <i>Set_clause LIST</i> → <i>Value</i> → <i>Col_spec LIST</i> → <i>Value</i> → <i>Query</i>
<b>delete</b>	<i>From_spec</i> → <i>Value</i> → <i>Query</i>
<b>insert</b>	<i>Table_spec</i> → <i>Col_name LIST</i> → <i>Insert_list</i> → <i>Query</i>
<b>Col_spec</b>	<i>Col_spec</i> → <i>Col</i>
<b>Table_spec</b>	<i>Table_spec</i> → <i>Col</i>
<b>projectTuples</b>	<i>State</i> → <i>Table_spec</i> → <i>Col</i> → <i>Tuple LIST</i> + <i>Errors</i>
<b>\$&amp;5</b>	( <i>Op</i> → <i>Update</i> + <i>Errors</i> ) → ( <i>Op</i> → <i>Update</i> + <i>Errors</i> ) → <i>Op</i> → <i>Update</i> + <i>Errors</i>
<b>\$&amp;6</b>	( <i>Op</i> → <i>Update</i> + <i>Errors</i> ) → ( <i>Op</i> → <i>Update</i> + <i>Errors</i> ) <i>LIST</i> → <i>Op</i> → <i>Update</i> + <i>Errors</i>
<b>auxapply</b>	<i>Col LIST</i> → <i>Tuple LIST</i> → <i>Env LIST</i> → <i>Env LIST</i>
<b>engroup</b>	<i>Col_spec LIST</i> → <i>Tuple LIST</i> → <i>Env LIST</i>
<b>colsInGroup</b>	<i>TableSpec</i> → <i>Worth</i> → <i>Bool LIST</i>
<b>andBools</b>	<i>Bool LIST</i> → <i>Bool</i>
<b>colposns</b>	<i>TableSpec</i> → <i>Worth</i> → <i>ColSpec</i> + <i>Errors</i>
<b>colspecs</b>	<i>TableSpec</i> → <i>Op</i> → <i>ColSpec</i> + <i>Errors</i>
<b>cons</b>	<i>TableSpec</i> → <i>Worth</i> → <i>ColCon</i> + <i>Errors</i>
<b>\$andb</b>	<i>Bool</i> + <i>Errors</i> → <i>Bool</i> → <i>Bool</i> + <i>Errors</i>
<b>checkGroup</b>	<i>TableSpec</i> → ( <i>ColCon</i> → <i>Bool</i> ) → ( <i>TableSpec</i> → <i>Worth</i> → <i>Bool</i> ) → <i>Bool</i>
<b>dataList</b>	<i>Row</i> → <i>Data LIST</i>
<b>checkUniqueness</b>	<i>TableSpec</i> → <i>Worth</i> → <i>Bool</i>
<b>test</b>	<i>Class LIST LIST</i> → <i>Bool</i>
<b>checkUniform</b>	<i>TableSpec</i> → <i>Worth</i> → <i>Bool</i>
<b>seqErr</b>	<i>Worth</i> → ' <i>x</i> + <i>Errors</i> → ' <i>x</i> <i>LIST</i> + <i>Errors</i>
<b>checkIntegrity</b>	<i>TableSpec</i> → ( <i>TableSpec</i> → <i>Worth</i> → <i>Bool</i> + <i>Errors</i> ) → <i>Bool</i>
<b>inRange</b>	<i>Class LIST</i> → <i>ColSpec LIST</i> → <i>Bool</i>
<b>elem</b>	<i>Worth</i> → ' <i>a</i> <i>LIST</i> → ' <i>a</i>
<b>checkFieldClasses</b>	<i>TableSpec</i> → <i>Worth</i> → <i>Bool</i> + <i>Errors</i>

---

---

<b>classType</b>	<i>Type</i>
<b>intervalType</b>	<i>Type</i>
<b>timeType</b>	<i>Type</i>
<b>floatingType</b>	<i>Type</i>
<b>integerType</b>	<i>Type</i>
<b>charsType</b>	<i>Type</i>
<b>booleanType</b>	<i>Type</i>
<b>monoleanType</b>	<i>Type</i>
<b>rightType</b>	<i>Item LIST</i> $\rightarrow$ <i>ColSpec LIST</i> $\rightarrow$ <i>Bool</i>
<b>checkType</b>	<i>TableSpec</i> $\rightarrow$ <i>Worth</i> $\rightarrow$ <i>Bool</i> + <i>Errors</i>
<b>rightNull</b>	<i>Item LIST</i> $\rightarrow$ <i>ColSpec LIST</i> $\rightarrow$ <i>Bool</i>
<b>checkNulls</b>	<i>TableSpec</i> $\rightarrow$ <i>Worth</i> $\rightarrow$ <i>Bool</i> + <i>Errors</i>
<b>Col_name</b>	<i>Col_name</i> $\rightarrow$ <i>Op</i>
<b>From_spec</b>	<i>From_spec</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Tuple LIST</i> + <i>Errors</i>
<b>From_name</b>	<i>From_spec</i> $\rightarrow$ <i>Col</i>
<b>Num_to_Int</b>	<i>Worth</i> $\rightarrow$ <i>Int</i>
<b>check_where_complete1</b>	<i>Bool</i> + <i>Errors</i> $\rightarrow$ <i>Bool</i> $\rightarrow$ <i>Data</i> $\rightarrow$ <i>Data</i> + <i>Errors</i>
<b>all_false</b>	<i>Bool LIST</i> $\rightarrow$ <i>Bool</i>
<b>same</b>	<i>Bool LIST</i> $\rightarrow$ <i>Bool</i> + <i>Errors</i>
<b>Value_monop</b>	( <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i> ) $\rightarrow$ <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i>
<b>Value_binop</b>	( <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i> ) $\rightarrow$ <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i>
<b>Value_set_func_all</b>	( <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i> ) $\rightarrow$ <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i>
<b>Value_all_binop</b>	( <i>Tuple_list</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Bool</i> + <i>Errors</i> ) $\rightarrow$ ( <i>Tuple_list</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Select</i> + <i>Errors</i> ) $\rightarrow$ ( <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i> ) $\rightarrow$ <i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i>
<b>Value_classification</b>	<i>Value</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Data</i> + <i>Errors</i>
<b>Value<sub>p</sub></b>	( <i>Tuple_list</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Bool</i> + <i>Errors</i> ) $\rightarrow$ ( <i>Tuple_list</i> $\rightarrow$ <i>State</i> $\rightarrow$ <i>Env</i> $\rightarrow$ <i>Select</i> + <i>Errors</i> )

---

→ *Value*  
 → *State*  
 → *Env*  
 → *Data + Errors*

**Tuple\_list\_complete<sub>p</sub>**

(*Value* → *State* → *Env* → *Data + Errors*)  
 → *Tuple\_list*  
 → *State*  
 → *Env*  
 → *Bool + Errors*

**Tuple\_list\_all\_tuples**

(*Value* → *State* → *Env* → *Data + Errors*)  
 → (*Select\_list*  
   → *State*  
   → *Env*  
   → *Col LIST*  
   → *Data LIST + Errors*)  
 → *Tuple\_list*  
 → *State*  
 → *Env*  
 → *Select + Errors*

**Tuple\_list\_evaluate**

(*Value* → *State* → *Env* → *Data + Errors*)  
 → (*Select\_list*  
   → *State*  
   → *Env*  
   → *Col LIST*  
   → *Data LIST + Errors*)  
 → *Tuple\_list*  
 → *State*  
 → *Env*  
 → *Select + Errors*

**Tuple\_list<sub>p</sub>**

(*Value* → *State* → *Env* → *Data + Errors*)  
 → (*Select\_list*  
   → *State*  
   → *Env*  
   → *Col LIST*  
   → *Data LIST + Errors*)  
 → *Tuple\_list*  
 → *State*  
 → *Env*  
 → *Select + Errors*

**Select\_list<sub>p</sub>**

(*Value* → *State* → *Env* → *Data + Errors*)  
 → *Select\_list*  
 → *State*  
 → *Env*  
 → *Col LIST*

---

	$\rightarrow$ Data LIST + Errors
<b>Select_list</b>	Select_list $\rightarrow$ State $\rightarrow$ Env $\rightarrow$ Col LIST $\rightarrow$ Data LIST + Errors
<b>Tuple_list</b>	Tuple_list $\rightarrow$ State $\rightarrow$ Env $\rightarrow$ Select + Errors
<b>Tuple_list_complete</b>	Tuple_list $\rightarrow$ State $\rightarrow$ Env $\rightarrow$ Bool + Errors
<b>Value</b>	Value $\rightarrow$ State $\rightarrow$ Env $\rightarrow$ Data + Errors
<b>Set_clause</b>	Set_clause $\rightarrow$ State $\rightarrow$ Env $\rightarrow$ Op $\rightarrow$ Update + Errors
<b>Classified_value</b>	Classified_value $\rightarrow$ State $\rightarrow$ Env $\rightarrow$ Data + Errors
<b>Insert_list</b>	Insert_list $\rightarrow$ State $\rightarrow$ Select + Errors
<b>processIntegrity</b>	TableSpec $\rightarrow$ Errors
<b>processInsert</b>	Class $\rightarrow$ Table_spec $\rightarrow$ Col_name LIST $\rightarrow$ Insert_list $\rightarrow$ State $\rightarrow$ Effect + Errors
<b>processDelete</b>	From_spec $\rightarrow$ Value $\rightarrow$ State $\rightarrow$ Effect + Errors
<b>convert</b>	(Worth $\rightarrow$ Update + Errors) $\rightarrow$ (Worth $\leftrightarrow$ Update) + Errors
<b>updateRowList</b>	Row LIST $\rightarrow$ Worth $\leftrightarrow$ Worth $\leftrightarrow$ Update $\rightarrow$ Row LIST
<b>processUpdate</b>	From_spec $\rightarrow$ Set_clause LIST $\rightarrow$ Value $\rightarrow$ Col_spec LIST $\rightarrow$ Value $\rightarrow$ State $\rightarrow$ Effect + Errors
<b>processSelect</b>	Tuple_list $\rightarrow$ State $\rightarrow$ Effect $\times$ Errors
<b>nullUpdate</b>	Effect
<b>processQuery</b>	Query $\times$ Class $\times$ State $\rightarrow$ Effect $\times$ Errors

## 18.4 Aliases

<b>true</b>	$T : Bool$
<b>false</b>	$F : Bool$
<b>maybe</b>	$One : Maybe$
<b>*</b>	$\$*_1$

---

```

      : ('b → 'a + Errors)
      → ('a → 'c + Errors)
      → 'b
      → 'c + Errors
*   $*2 : 'a + Errors → ('a → 'b + Errors) → 'b + Errors
*   $*3
      : 'a + Errors
      → ('a → 'b → 'c + Errors)
      → 'b
      → 'c + Errors
*   $*4
      : ('b → 'a + Errors) → ('a → 'c) → 'b → 'c + Errors
*   $*6
      : 'a + Errors × 'b + Errors
      → ('a → 'b → 'c)
      → 'c + Errors
&   $&1 : 'a + Errors → 'a + Errors → 'a LIST + Errors
&   $&2
      : 'a + Errors → 'a LIST + Errors → 'a LIST + Errors
&   $&3
      : 'a LIST + Errors → 'a + Errors → 'a LIST + Errors
&   $&4
      : 'a LIST + Errors
      → 'a LIST + Errors
      → 'a LIST + Errors
**  $**1 : Tuple → Tuple → Tuple
**  $**2 : Tuple → Tuple LIST → Tuple LIST
&   $&5
      : (Op → Update + Errors)
      → (Op → Update + Errors)
      → Op
      → Update + Errors
&   $&6
      : (Op → Update + Errors)
      → (Op → Update + Errors) LIST
      → Op
      → Update + Errors

```

## 18.5 Types

**GroupedResult**

**Env**

**Value**

**Col\_spec**

**Table\_spec**

**Col\_name**

**From\_spec**

**Set\_clause**

**Tuple\_list**  
**Insert\_list**  
**Classified\_value**  
**Select\_list**  
**Query**

## 18.6 Type Abbreviations

<b>Col</b>	<i>Col</i>
<b>Tuple</b>	<i>Tuple</i>
<b>Maybe</b>	<i>Maybe</i>
<b>MaybeResult</b>	<i>MaybeResult</i>
<b>Op</b>	<i>Op</i>

## 18.7 Fixity

*Right Infix 40:*

**andb**

*Right Infix 150:*

**dominates\_w starstar2**     $\&$      $\&_4$   
**list\_∪ star1**     $\&_1$      $\&_5$   
**lub\_w star2**     $\&_2$      $\&_6$   
**starstar1**    **star3**     $\&_3$     **\*\***

*Right Infix 300:*

**intPlus**     $*_2$      $*_4$      $*_6$   
 $*_1$      $*_3$      $*_5$      $*_7$

## 18.8 Definitions

**destValuedItem**

**destNullItem**     $\vdash$  *ConstSpec*  
 $(\lambda (destValuedItem', destNullItem')$   
 $\bullet \forall v n$   
 $\bullet destValuedItem' (ValuedItemItem v) = v$   
 $\quad \wedge destNullItem' (NullItemItem n) = n)$   
 $(destValuedItem, destNullItem)$

**isValuedItem**

**isNullItem**     $\vdash \forall i$   
 $\bullet (isValuedItem i \Leftrightarrow (\exists v \bullet i = ValuedItemItem v))$   
 $\quad \wedge (isNullItem i \Leftrightarrow (\exists n \bullet i = NullItemItem n))$

**destBoolVal**

**destStringVal**

**destIntVal**

**destFloatVal**

**destTimeVal**

**destIntervalVal**

**destCodeVal**

---

**destClassVal**  $\vdash \text{ConstSpec}$   
 $(\lambda$   
 $(\text{destBoolVal}', \text{destStringVal}', \text{destIntVal}',$   
 $\text{destFloatVal}', \text{destTimeVal}',$   
 $\text{destIntervalVal}', \text{destCodeVal}',$   
 $\text{destClassVal}')$   

- $\forall b s i f t int c cl$ 
  - $(\text{destBoolVal}' (\text{BoolVal } b) \Leftrightarrow b)$   
 $\wedge \text{destStringVal}' (\text{StringVal } s) = s$   
 $\wedge \text{destIntVal}' (\text{IntVal } i) = i$   
 $\wedge \text{destFloatVal}' (\text{FloatVal } f) = f$   
 $\wedge \text{destTimeVal}' (\text{TimeVal } t) = t$   
 $\wedge \text{destIntervalVal}' (\text{IntervalVal } int) = int$   
 $\wedge \text{destCodeVal}' (\text{CodeVal } c) = c$   
 $\wedge \text{destClassVal}' (\text{ClassVal } cl) = cl)$

 $(\text{destBoolVal}, \text{destStringVal}, \text{destIntVal},$   
 $\text{destFloatVal}, \text{destTimeVal}, \text{destIntervalVal},$   
 $\text{destCodeVal}, \text{destClassVal})$

**isBoolVal**  
**isStringVal**  
**isIntVal**  
**isFloatVal**  
**isTimeVal**  
**isIntervalVal**  
**isCodeVal**  
**isClassVal**  $\vdash \forall v$   

- $(\text{isBoolVal } v \Leftrightarrow (\exists b \bullet v = \text{BoolVal } b))$   
 $\wedge (\text{isStringVal } v \Leftrightarrow (\exists s \bullet v = \text{StringVal } s))$   
 $\wedge (\text{isIntVal } v \Leftrightarrow (\exists i \bullet v = \text{IntVal } i))$   
 $\wedge (\text{isFloatVal } v \Leftrightarrow (\exists f \bullet v = \text{FloatVal } f))$   
 $\wedge (\text{isTimeVal } v \Leftrightarrow (\exists t \bullet v = \text{TimeVal } t))$   
 $\wedge (\text{isIntervalVal } v$   
 $\Leftrightarrow (\exists int \bullet v = \text{IntervalVal } int))$   
 $\wedge (\text{isCodeVal } v \Leftrightarrow (\exists c \bullet v = \text{CodeVal } c))$   
 $\wedge (\text{isClassVal } v \Leftrightarrow (\exists cl \bullet v = \text{ClassVal } cl))$

**GroupedResult**  $\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet \text{true}) f$

**MkGroupedResult**  
**G\_res**  
**G\_group**  $\vdash \forall t x1 x2$   

- $G\_res (\text{MkGroupedResult } x1 x2) = x1$   
 $\wedge G\_group (\text{MkGroupedResult } x1 x2) = x2$   
 $\wedge \text{MkGroupedResult } (G\_res t) (G\_group t) = t$

**Env**  $\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet \text{true}) f$   
**MkEnv**  
**E\_row**  
**E\_group**  $\vdash \forall t x1 x2$   

- $E\_row (\text{MkEnv } x1 x2) = x1$

---

---


$$\wedge E\_group (MkEnv x1 x2) = x2$$

$$\wedge MkEnv (E\_row t) (E\_group t) = t$$

**userName**  
**userDirectory**  
**userClearance**  
**timeNow**  $\vdash true$   
**lubl**  $\vdash \forall cl \bullet lubl\ cl = Fold \$lub\ cl\ lattice\_bottom$   
**glbl**  $\vdash \forall cl \bullet glbl\ cl = Fold \$glb\ cl\ lattice\_top$   
**lub\_wl**  
**lub\_w**  
**dominates\_w**  $\vdash true$   
**Not**  
**And**  
**Or**  
**Plus**  
**Equal**  $\vdash Not = "Not"$   
 $\wedge And = "And"$   
 $\wedge Or = "Or"$   
 $\wedge Plus = "Plus"$   
 $\wedge Equal = "Equal"$

**newData**  $\vdash \forall c\ w\ v$   
 $\bullet newData\ c\ w\ v$   
 $= MkData\ c\ (ValuedItemItem\ (MkValuedItem\ w\ v))$

**lub\_data**  $\vdash \forall dl$   
 $\bullet lub\_data\ dl$   
 $= userClearance\ lub\ lubl\ (Map\ Dat\_class\ dl)$

**lub\_wdata**  $\vdash \forall dl$   
 $\bullet lub\_wdata\ dl$   
 $= lub\_wl$   
 $(Map\ (VI\_worth\ o\ destValuedItem\ o\ Dat\_item)\ dl)$

**ExceptionData**  
 $\vdash \forall dl$   
 $\bullet ExceptionData\ dl$   
 $= newData\ (lub\_data\ dl)\ worthless\ ExceptionVal$

**NullData**  $\vdash \forall dl$   
 $\bullet NullData\ dl$   
 $= MkData\ (lub\_data\ dl)\ (NullItemItem\ maybe)$

**applyNot**  $\vdash \forall dl$   
 $\bullet applyNot\ dl$   
 $= (if\ \neg\ \# dl = 1$   
 $then\ ExceptionData\ dl$   
 $else\ if\ isNullItem\ (Dat\_item\ (Head\ dl))$   
 $then\ Head\ dl$   
 $else$   
 $(let\ v$   
 $= VI\_val$   
 $(destValuedItem\ (Dat\_item\ (Head\ dl)))$   
 $in\ if\ isBoolVal\ v$

---

---

```

    then
      newData
        (lub_data dl)
        (lub_wdata dl)
        (BoolVal (¬ destBoolVal v))
    else ExceptionData dl)
applyAnd ⊢ ∀ dl
  • applyAnd dl
    = (if ¬ # dl = 2
      then ExceptionData dl
      else
        (let d1 = Head dl and d2 = Head (Tail dl)
          in if isValuedItem (Dat_item d1)
            then
              let v1
                = VI_val
                  (destValuedItem (Dat_item d1))
              in if isValuedItem (Dat_item d2)
                then
                  let v2
                    = VI_val
                      (destValuedItem
                        (Dat_item d2))
                  in if isBoolVal v1 ∧ isBoolVal v2
                    then
                      newData
                        (lub_data dl)
                        (lub_wdata dl)
                        (BoolVal
                          (destBoolVal v1
                            ∧ destBoolVal v2))
                    else if
                      isBoolVal v1
                        ∧ (destBoolVal v1 ⇔ false)
                    then
                      newData
                        (lub_data dl)
                        (lub_wdata dl)
                        (BoolVal false)
                    else if
                      isBoolVal v2
                        ∧ (destBoolVal v2 ⇔ false)
                    then
                      newData
                        (lub_data dl)
                        (lub_wdata dl)
                        (BoolVal false)
                    else ExceptionData dl)

```

---

---

```

        else NullData dl
        else NullData dl))
applyOr    ⊢ ∀ dl
    • applyOr dl
      = (if ¬ # dl = 2
        then ExceptionData dl
        else
          (let d1 = Head dl and d2 = Head (Tail dl)
            in if isValuedItem (Dat_item d1)
              then
                let v1
                  = VI_val
                    (destValuedItem (Dat_item d1))
                in if isValuedItem (Dat_item d2)
                  then
                    let v2
                      = VI_val
                        (destValuedItem
                          (Dat_item d2))
                    in if isBoolVal v1 ∧ isBoolVal v2
                      then
                        newData
                          (lub_data dl)
                          (lub_wdata dl)
                          (BoolVal
                            (destBoolVal v1
                              ∨ destBoolVal v2))
                      else if
                        isBoolVal v1
                          ∧ (destBoolVal v1 ⇔ true)
                      then
                        newData
                          (lub_data dl)
                          (lub_wdata dl)
                          (BoolVal true)
                      else if
                        isBoolVal v2
                          ∧ (destBoolVal v2 ⇔ true)
                      then
                        newData
                          (lub_data dl)
                          (lub_wdata dl)
                          (BoolVal true)
                      else ExceptionData dl
                  else NullData dl
              else NullData dl))
intPlus    ⊢ true
applyPlus  ⊢ ∀ dl

```

• *applyPlus dl*  
 = (if  $\neg \# dl = 2$   
 then *ExceptionData dl*  
 else  
 (let  $d_1 = \text{Head } dl$  and  $d_2 = \text{Head } (\text{Tail } dl)$   
 in if  
   *isNullItem* (*Dat\_item*  $d_1$ )  
    $\vee$  *isNullItem* (*Dat\_item*  $d_2$ )  
 then *NullData dl*  
 else  
 (let  $v_1$   
   = *VI\_val*  
   (*destValuedItem* (*Dat\_item*  $d_1$ ))  
 and  $v_2$   
   = *VI\_val*  
   (*destValuedItem* (*Dat\_item*  $d_2$ ))  
 in if *isIntVal*  $v_1 \wedge$  *isIntVal*  $v_2$   
 then  
   *newData*  
   (*lub\_data*  $dl$ )  
   (*lub\_wdata*  $dl$ )  
   (*IntVal*  
     (*destIntVal*  $v_1$   
       *intPlus* *destIntVal*  $v_2$ ))  
   else *ExceptionData dl*)))

**applyEqual**     $\vdash \forall dl$   
 • *applyEqual dl*  
 = (if  $\neg \# dl = 2$   
 then *ExceptionData dl*  
 else  
 (let  $d_1 = \text{Head } dl$  and  $d_2 = \text{Head } (\text{Tail } dl)$   
 in if  
   *isNullItem* (*Dat\_item*  $d_1$ )  
    $\vee$  *isNullItem* (*Dat\_item*  $d_2$ )  
 then *NullData dl*  
 else  
 (let  $v_1$   
   = *VI\_val*  
   (*destValuedItem* (*Dat\_item*  $d_1$ ))  
 and  $v_2$   
   = *VI\_val*  
   (*destValuedItem* (*Dat\_item*  $d_2$ ))  
 in *newData*  
   (*lub\_data*  $dl$ )  
   (*lub\_wdata*  $dl$ )  
   (*BoolVal* ( $v_1 = v_2$ ))))

**apply**             $\vdash \text{ConstSpec}$   
                   ( $\lambda \text{ apply}'$ )

---

```

    •  $\forall dl$ 
      • apply' Not  $dl = applyNot dl$ 
         $\wedge$  apply' And  $dl = applyAnd dl$ 
         $\wedge$  apply' Or  $dl = applyOr dl$ 
         $\wedge$  apply' Plus  $dl = applyPlus dl$ 
         $\wedge$  apply' Equal  $dl = applyEqual dl$ )

    apply
*1  $\vdash \forall f g x$ 
    •  $(f * g) x$ 
      = (if isVal (f x)
        then g (destVal (f x))
        else giveError (destError (f x)))

*2  $\vdash \forall x f$ 
    •  $x * f$ 
      = (if isVal x
        then f (destVal x)
        else giveError (destError x))

*3  $\vdash \forall x f y$ 
    •  $(x * f) y$ 
      = (if isVal x
        then f (destVal x) y
        else giveError (destError x))

*4  $\vdash \forall f g x$ 
    •  $(f * g) x$ 
      = (if isVal (f x)
        then giveVal (g (destVal (f x)))
        else giveError (destError (f x)))

*5  $\vdash \forall x f$ 
    •  $x *_5 f$ 
      = (if isVal x
        then giveVal (f (destVal x))
        else giveError (destError x))

*6  $\vdash \forall f x y$ 
    •  $(x, y) * f$ 
      = (if isVal x
        then
          if isVal y
            then giveVal (f (destVal x) (destVal y))
            else giveError (destError y)
          else if isVal y
            then giveError (destError x)
            else giveError (destError x @ destError y))

*7  $\vdash \forall f x$ 
    •  $x *_7 f$ 
      = (if isVal f
        then giveVal (destVal f x)
        else giveError (destError f))

list_U  $\vdash ConstSpec$ 

```

---

---

```

      (λ $"list_⊔'"
        • ∀ l1
          • $"list_⊔'" l1 [] = l1
            ∧ (∀ l2 x
              • $"list_⊔'" l1 (l2 @ [x])
                = (if x ∈ Elems l1
                  then $"list_⊔'" l1 l2
                  else $"list_⊔'" l1 l2 @ [x]))))
    $list_⊔
&₁ ⊢ ∀ x1 x2
    • (x1 & x2)
      = (if isVal x1
        then
          if isVal x2
            then giveVal ([destVal x1] @ [destVal x2])
            else giveError (destError x2)
          else if isVal x2
            then giveError (destError x1)
            else giveError (destError x1 @ destError x2))
&₂ ⊢ ∀ x xl
    • (x & xl)
      = (if isVal x
        then
          if isVal xl
            then giveVal (Cons (destVal x) (destVal xl))
            else giveError (destError xl)
          else if isVal xl
            then giveError (destError x)
            else giveError (destError x @ destError xl))
&₃ ⊢ ∀ xl x
    • (xl & x)
      = (if isVal x
        then
          if isVal xl
            then giveVal (destVal xl @ [destVal x])
            else giveError (destError xl)
          else if isVal xl
            then giveError (destError x)
            else giveError (destError xl @ destError x))
&₄ ⊢ ∀ xl1 xl2
    • (xl1 & xl2)
      = (if isVal xl1
        then
          if isVal xl2
            then giveVal (destVal xl1 @ destVal xl2)
            else giveError (destError xl2)
          else if isVal xl2
            then giveError (destError xl1)

```

---

```

      else
      giveError (destError xl1 @ destError xl2)
seq      ⊢ ConstSpec
      (λ seq'
        • ∀ n x
          • seq' n x
            = (if n = 0
              then []
              else Cons x (seq' (n - 1) x)))

      seq
enseq    ⊢ ∀ f
        • enseq f
          = RelList
            {(i, j)
             | 1 ≤ i
               ∧ (∀ k • 1 ≤ k ∧ k ≤ i ⇒ isVal (f k))
               ∧ j = destVal (f i)}

      promote
promote  ⊢ ConstSpec
      (λ promote'
        • ∀ f s
          • promote' f s
            = (if s = []
              then giveVal []
              else f (Head s) & promote' f (Tail s)))

      promote
find     ⊢ ConstSpec
      (λ find'
        • ∀ s e
          • find' s e
            = (if s = []
              then giveError (seq 1 error)
              else if Head s = e
                then giveVal 1
                else if isVal (find' (Tail s) e)
                  then
                    giveVal
                      (1 + destVal (find' (Tail s) e))
                  else find' (Tail s) e))

      find
distinct ⊢ ConstSpec
      (λ distinct'
        • distinct' [] = []
          ∧ (∀ x l
            • distinct' (l @ [x])
              = (if x ∈ Elems l
                then distinct' l
                else distinct' l @ [x])))
distinct

```

---

---

<b>noErrors</b>	$\vdash noErrors = []$
<b>one_col</b>	$\vdash one\_col$ $= promote$ $(\lambda ds$ <ul style="list-style-type: none"> <li>• <math>if \# ds = 1</math>  <math>then giveVal (Head ds)</math></li> <li>• <math>else giveError (seq 1 tooWide))</math></li> </ul>
<b>one_result</b>	$\vdash \forall ds$ <ul style="list-style-type: none"> <li>• <math>one\_result ds</math>  <math>= (if \# ds = 1</math>  <math>then giveVal (Head ds)</math>  <math>else giveError (seq 1 tooTall))</math></li> </ul>
<b>make_data</b>	$\vdash \forall v \bullet make\_data v = newData userClearance worthless v$
<b>fillTab</b>	$\vdash \forall i$ <ul style="list-style-type: none"> <li>• <math>fillTab i</math>  <math>= (if \# i = 1 then userDirectory @ i else i)</math></li> </ul>
<b>fillCol</b>	$\vdash \forall i$ <ul style="list-style-type: none"> <li>• <math>fillCol i</math>  <math>= (if \# i = 2 then userDirectory @ i else i)</math></li> </ul>
<b>getDir</b>	$\vdash \forall st tab$ <ul style="list-style-type: none"> <li>• <math>getDir st tab</math>  <math>= (if tab \in Dom (repState st)</math>  <math>then giveVal (repState st @ tab)</math>  <math>else giveError (seq 1 noSuchDirectory))</math></li> </ul>
<b>getTab</b>	$\vdash \forall dir i$ <ul style="list-style-type: none"> <li>• <math>getTab dir i</math>  <math>= (if i \in Dom (Dir\_tables dir)</math>  <math>then giveVal (Dir\_tables dir @ i)</math>  <math>else giveError (seq 1 noSuchTable))</math></li> </ul>
<b>getColPosn</b>	$\vdash \forall ts i$ <ul style="list-style-type: none"> <li>• <math>getColPosn ts i</math>  <math>= (if</math>  <math>i</math>  <math>\in \{name</math>  <math> \exists c \bullet c \in TS\_colspecs ts \wedge CS\_ide c = name\}</math>  <math>then</math>  <math>giveVal</math>  <math>(\epsilon n</math>  <ul style="list-style-type: none"> <li>• <math>\exists c</math>  <ul style="list-style-type: none"> <li>• <math>c \in TS\_colspecs ts</math>  <math>\wedge CS\_ide c = i</math>  <math>\wedge CS\_posn c = n)</math></li> </ul> </li> <li>• <math>else giveError (seq 1 noSuchColumn))</math></li> </ul> </li> </ul>
<b>getData</b>	$\vdash \forall r n$ <ul style="list-style-type: none"> <li>• <math>getData r n</math>  <math>= (if n \in Dom (R\_data r)</math>  <math>then giveVal (R\_data r @ n)</math>  <math>else giveError (seq 1 noSuchColumn))</math></li> </ul>

---

---

**lookup**       $\vdash \forall st\ tab$   
                   • *lookup st tab*  
                   = (let *dir* = *getDir st (Front tab)*  
                   in if *isVal dir*  
                   then *getTab (destVal dir) (Last tab)*  
                   else *giveError (destError dir)*)

**isCleared**       $\vdash \forall d$   
                   • *isCleared d*  $\Leftrightarrow$  *userClearance dominates Dat\_class d*

**isNotCleared**    $\vdash \forall d$  • *isNotCleared d*  $\Leftrightarrow$   $\neg$  *isCleared d*

**checkComplete**  
                    $\vdash$  (*checkComplete []*  $\Leftrightarrow$  *true*)  
                    $\wedge$  ( $\forall d\ tl$   
                   • *checkComplete (Cons d tl)*  
                    $\Leftrightarrow$  (if *isNotCleared d*  
                   then *false*  
                   else *checkComplete tl*))

**check\_where\_complete**  
                    $\vdash \forall comp\ b\ d$   
                   • *check\_where\_complete comp b d*  
                   = (if *comp*  
                   then *giveVal d*  
                   else if *isNotCleared d*  
                   then *giveError (seq 1 notCleared)*  
                   else if *isNullItem (Dat\_item d)*  
                   then *giveVal d*  
                   else  
                   (let *v* = *VI\_val (destValuedItem (Dat\_item d))*  
                   in if *isBoolVal v*  
                   then  
                   if *destBoolVal v*  $\Leftrightarrow$  *b*  
                   then *giveError (seq 1 notCleared)*  
                   else *giveVal d*  
                   else *giveError (seq 1 wrongType)*))

**resultBool**       $\vdash \forall d$   
                   • *resultBool d*  
                   = (if *isNotCleared d*  
                   then *giveVal false*  
                   else if *isNullItem (Dat\_item d)*  
                   then *giveVal false*  
                   else  
                   (let *v* = *VI\_val (destValuedItem (Dat\_item d))*  
                   in if *isBoolVal v*  
                   then *giveVal (destBoolVal v)*  
                   else *giveError (seq 1 wrongType)*))

**take\_data**       $\vdash \forall d$   
                   • *take\_data d*  
                   = (if *isCleared d*  
                   then *giveVal d*

---

---

```

    else giveError (seq 1 notCleared))
resultClass    ⊢ ∀ d
    • resultClass d
      = (let vi i
          = (if isValuedItem i
              then giveVal (destValuedItem i)
              else giveError (seq 1 nullValue))
          and valClass v
          = (if isClassVal v
              then giveVal (destClassVal v)
              else giveError (seq 1 wrongType))
          in ((take_data d *5 Dat_item) * vi *5 VI_val)
             * valClass)
extract       ⊢ ConstSpec
    (λ extract'
    • ∀ bs s
      • extract' bs s
        = (if bs = []
            then s
            else if s = []
              then s
              else if Head bs ⇔ true
                then
                  Cons
                    (Head s)
                    (extract' (Tail bs) (Tail s))
                else extract' (Tail bs) (Tail s)))
    extract
emptyTuple   ⊢ ∀ c • emptyTuple c = giveError (seq 1 noSuchColumn)
emptyEnv     ⊢ emptyEnv = MkEnv emptyTuple (seq 1 emptyTuple)
star1        ⊢ ∀ a b c
    • (a star1 b) c
      = (if a c = giveError (seq 1 noSuchColumn)
          then b c
          else if b c = giveError (seq 1 noSuchColumn)
            then a c
            else if isVal (a c) ∧ isVal (b c)
              then giveError (seq 1 ambiguousColumn)
              else if isError (a c) ∧ isError (b c)
                then
                  giveError
                    (destError (a c) list_∪ destError (b c))
                else if isError (a c)
                  then a c
                  else b c)
star2       ⊢ ConstSpec
    (λ star2'
    • ∀ a b

```

---

---

```

      • star2' a b
      = (if b = []
         then []
         else
           Cons
            (a star1 Head b)
            (star2' a (Tail b))))
$star2
star3  ⊢ ConstSpec
      (λ star3'
       • ∀ a b
       • star3' a b
       = (if a = []
          then []
          else
            (Head a star2 b) @ star3' (Tail a) b))
$star3
jay    ⊢ ConstSpec
      (λ jay'
       • ∀ t s
       • jay' t s
       = (if s = []
          then t
          else jay' (t star3 Head s) (Tail s)))
jay
join   ⊢ join = jay []
starstar1 ⊢ ∀ a b c
      • (a ** b) c
      = (if b c = giveError (seq 1 noSuchColumn)
         then a c
         else b c)
starstar2 ⊢ ConstSpec
      (λ starstar2'
       • ∀ a b
       • starstar2' a b
       = (if b = []
          then []
          else
            Cons
             (a ** Head b)
             (starstar2' a (Tail b))))
$**
insert
delete
update
select
all_columns
select_values

```

---

**classify**  
**classify\_default**  
**insert\_tuples**  
**tuple**  
**union**  
**all\_tuples**  
**distinct\_tuples**  
**evaluate**  
**set\_value**  
**set\_class**  
**set\_class\_and\_value**  
**from**  
**correlate\_from**  
**denote\_col\_spec**  
**denote\_table\_spec**  
**denote\_col\_name**  
**user**  
**clearance**  
**current\_time**  
**monop**  
**binop**  
**set\_func\_all**  
**count\_all**  
**all\_binop**  
**contents**  
**classification**  
**denote\_null**  
**denote\_void**  
**denote\_true**  
**denote\_false**  
**denote\_integer**  
**denote\_string**  
**denote\_float**  
**denote\_time**  
**denote\_interval**  
**denote\_class**  
**denote\_code**    $\vdash$  *true*  
**Col\_spec**        $\vdash$  *ConstSpec*  
                   $(\lambda$  *Col\_spec'*  
                  •  $\forall$  *il*  
                  • *Col\_spec'* (*denote\_col\_spec il*) = *fillCol il*)  
                  *Col\_spec*  
**Table\_spec**      $\vdash$  *ConstSpec*  
                   $(\lambda$  *Table\_spec'*  
                  •  $\forall$  *il*  
                  • *Table\_spec'* (*denote\_table\_spec il*)  
                  = *fillTab il*)  
                  *Table\_spec*

---

**projectTuples**

$$\vdash \forall st\ ts\ cn$$

- *projectTuples st ts cn*
  - = (let table = Table\_spec ts
  - in let spec = lookup st table
  - in if isError spec
  - then giveError (destError spec)
  - else
  - (let coln c
  - = (if Front c = table
  - then
  - getColPosn
  - (destVal spec)
  - (Last c)
  - else if Front c = cn
  - then
  - getColPosn
  - (destVal spec)
  - (Last c)
  - else
  - giveError (seq 1 noSuchColumn))
  - in let entuple r
  - = giveVal ( $\lambda c \bullet \text{coln } c * \text{getData } r$ )
  - in promote
  - entuple
  - (TS\_rows (destVal spec))))

&amp;5

$$\vdash \forall f\ g\ c$$

- (f & g) c
  - = (if
  - isError (f c)  $\wedge$  destError (f c) = seq 1 error
  - $\vee$  isError (g c)
  - $\wedge$  destError (g c) = seq 1 error
  - then f c
  - else if
  - isClass (destVal (f c))
  - $\wedge$  isItem (destVal (g c))
  - then
  - giveVal
  - (DataUpdate
  - (MkData
  - (destClass (destVal (f c)))
  - (destItem (destVal (g c))))))
  - else if
  - isClass (destVal (g c))
  - $\wedge$  isItem (destVal (f c))
  - then
  - giveVal
  - (DataUpdate
  - (MkData
  - (destClass (destVal (g c)))
  - (destItem (destVal (f c))))))

---

```

      (MkData
        (destClass (destVal (g c)))
        (destItem (destVal (f c))))
    else giveError (seq 1 ambiguousUpdate))
&₆ ⊢ ConstSpec
  (λ &'₆
    • ∀ f s
    • &'₆ f s
      = (if s = []
        then f
        else &'₆ (f & Head s) (Tail s)))
  $&
auxapply ⊢ ConstSpec
  (λ auxapply'
    • ∀ cs ts es
    • auxapply' cs ts es
      = (let add cs t e
        = (if
          promote t cs
            = promote (E_row e) cs
          then
            MkEnv (E_row e) (E_group e @ [t])
          else e)
        in let adds cs t = Map (add cs t)
        in if ts = []
        then es
        else
          auxapply'
            cs
            (Tail ts)
            (adds cs (Head ts) es)))
    auxapply
engroup ⊢ ∀ g ts
  • engroup g ts
    = auxapply
      (Map (λ c • Col_spec c) g)
      ts
      (Map (λ t • MkEnv t []) ts)
colsInGroup
andBools ⊢ true
  ⊢ ConstSpec
    (λ andBools'
      • ∀ bs
      • andBools' bs
        ⇔ (if # bs = 0
          then true
          else Head bs ∧ andBools' (Tail bs)))
    andBools
colposns ⊢ ∀ t n

```

---

---

```

    • colposns t n
      = (if ∃1 c • c ∈ TS_colspecs t ∧ CS_posn c = n
        then
          giveVal
            (ε c • c ∈ TS_colspecs t ∧ CS_posn c = n)
        else giveError (seq 1 noSuchColumn))
colspecs ⊢ ∀ t i
    • colspecs t i
      = (if ∃1 c • c ∈ TS_colspecs t ∧ CS_ide c = i
        then
          giveVal
            (ε c • c ∈ TS_colspecs t ∧ CS_ide c = i)
        else giveError (seq 1 noSuchColumn))
cons ⊢ ∀ t n
    • cons t n
      = (if ∃1 cc • (n, cc) ∈ TS_cons t
        then giveVal (TS_cons t @ n)
        else giveError (seq 1 error))
andb ⊢ ∀ be b
    • (be andb b)
      = (if isVal be
        then giveVal (destVal be ∧ b)
        else giveError (destError be))
checkGroup ⊢ ∀ ts cond check
    • checkGroup ts cond check
      ⇔ andBools
        (enseq
          (λ n • cons ts n *5 cond andb check ts n))
dataList ⊢ ∀ r • dataList r = RelList (Squash (R_data r))
checkUniqueness ⊢ ∀ ts n
    • checkUniqueness ts n
      ⇔ (let key
        = (let ii
          = dataList
            % extract (colsInGroup ts n)
            % Map Dat_item in TS_rows % Map ii)
        in key ts = distinct (key ts))
test ⊢ ConstSpec
  (λ test'
    • ∀ css
      • test' css
        ⇔ (if # css = 0
          then true
          else if # (distinct (Head css)) = 1
            then test' (Tail css)
            else false))
    test
  
```

---

---

**checkUniform**  $\vdash \forall ts\ n$

- *checkUniform* *ts n*

$$\Leftrightarrow (\text{let } \textit{classes} \\ = (\text{let } \textit{cc} \\ = \textit{dataList} \\ \quad \textit{\% extract (colsInGroup ts n)} \\ \quad \textit{\% Map Dat\_class} \\ \textit{in TS\_rows \% Map cc)} \\ \textit{in test (classes ts)})$$

**seqErr**  $\vdash \forall n\ xe$

- *seqErr* *n xe*

$$= (\text{if } \textit{isVal } \textit{xe} \\ \textit{then giveVal (seq n (destVal xe))} \\ \textit{else giveError (destError xe)})$$

**checkIntegrity**  $\vdash \forall ts\ \textit{check}$

- *checkIntegrity* *ts check*

$$\Leftrightarrow \textit{andBools (enseq (\lambda n \bullet \textit{check } \textit{ts } \textit{n}))}$$

**inRange**  $\vdash \textit{ConstSpec}$

$$(\lambda \textit{inRange}' \\ \bullet \forall \textit{cs } \textit{css} \\ \bullet \textit{inRange}' \textit{cs } \textit{css} \\ \Leftrightarrow (\text{let } \textit{inRan } \textit{cl } \textit{c} \\ \Leftrightarrow \textit{cl dominates CS\_min } \textit{c} \\ \quad \wedge \textit{CS\_max } \textit{c dominates cl} \\ \textit{in if } \# \textit{cs} = 0 \vee \# \textit{css} = 0 \\ \textit{then true} \\ \textit{else} \\ \textit{inRan (Head cs) (Head css)} \\ \quad \wedge \textit{inRange}' (\textit{Tail cs}) (\textit{Tail css}))) \\ \textit{inRange}$$

**elem**  $\vdash \forall n\ \textit{xl} \bullet \textit{elem } \textit{n } \textit{xl} = \textit{Nth } \textit{xl } \textit{n}$

**checkFieldClasses**  $\vdash \forall ts\ n$

- *checkFieldClasses* *ts n*

$$= (\text{let } \textit{classes} \\ = (\textit{TS\_rows} \\ \quad \textit{\% Map (dataList \% elem n \% Dat\_class)} \\ \textit{ts} \\ \textit{and } \textit{cs} = \textit{colposns } \textit{ts } \textit{n} \\ \textit{in seqErr } \textit{n } \textit{cs } *_{\textit{5}} \textit{inRange } \textit{classes})$$

**monoleanType**

**booleanType**

**charsType**

**integerType**

**floatingType**

**timeType**

**intervalType**

---

---

```

classType    ⊢ monoleanType = InL maybe
                ∧ booleanType = InR (InL maybe)
                ∧ charsType = InR (InR (InL maybe))
                ∧ integerType = InR (InR (InR (InL maybe)))
                ∧ floatingType = InR (InR (InR (InR (InL maybe))))
                ∧ timeType
                  = InR (InR (InR (InR (InR (InL maybe)))))
                ∧ intervalType
                  = InR (InR (InR (InR (InR (InR (InL maybe))))))
                ∧ classType
                  = InR (InR (InR (InR (InR (InR (InR maybe))))))

rightType   ⊢ ConstSpec
                (λ rightType'
                 • ∀ is css
                 • rightType' is css
                 ⇔ (let rightT i cs
                     ⇔ (let right t v
                         ⇔ t = monoleanType
                           ∧ v = VoidVal
                           ∨ t = booleanType
                           ∧ isBoolVal v
                           ∨ t = charsType
                           ∧ isStringVal v
                           ∨ t = integerType
                           ∧ isIntVal v
                           ∨ t = floatingType
                           ∧ isFloatVal v
                           ∨ t = timeType ∧ isTimeVal v
                           ∨ t = intervalType
                           ∧ isIntervalVal v
                           ∨ t = classType
                           ∧ isClassVal v
                         in if isNullItem i
                           then true
                           else
                             (let vi = destValuedItem i
                               in if VI_worth vi = sterling
                                 then
                                   right
                                   (CS_sterlingType cs)
                                   (VI_val vi)
                                 else
                                   right
                                   (CS_dinaryType cs)
                                   (VI_val vi)))
                       in if # is = 0 ∨ # css = 0
                           then true
                           else

```

---

---

$rightT (Head\ is) (Head\ css)$   
 $\wedge\ rightType' (Tail\ is) (Tail\ css)))$

**checkType**  $\vdash \forall ts\ n$

- $checkType\ ts\ n$   
 $= (let\ itemList$   
 $= (TS\_rows$   
 $\% Map (dataList\ \% elem\ n\ \% Dat\_item))$   
 $ts$   
 $and\ cs = colposns\ ts\ n$   
 $in\ seqErr\ n\ cs\ *5\ rightType\ itemList)$

**rightNull**  $\vdash ConstSpec$

$(\lambda\ rightNull'$

- $\forall is\ css$
- $rightNull'\ is\ css$   
 $\Leftrightarrow (let\ rightN\ i\ cs$   
 $\Leftrightarrow isNullItem\ i\ \wedge\ CS\_nullType\ cs$   
 $in\ if\ \# is = 0\ \vee\ \# css = 0$   
 $then\ true$   
 $else$   
 $rightN (Head\ is) (Head\ css)$   
 $\wedge\ rightNull' (Tail\ is) (Tail\ css)))$

$rightNull$

**checkNulls**  $\vdash \forall ts\ n$

- $checkNulls\ ts\ n$   
 $= (let\ itemList$   
 $= (TS\_rows$   
 $\% Map (dataList\ \% elem\ n\ \% Dat\_item))$   
 $ts$   
 $and\ cs = colposns\ ts\ n$   
 $in\ seqErr\ n\ cs\ *5\ rightNull\ itemList)$

**Col\_name**  $\vdash ConstSpec$

$(\lambda\ Col\_name'$

- $\forall i$
- $Col\_name' (denote\_col\_name\ i) = i$

$Col\_name$

**From\_spec**  $\vdash ConstSpec$

$(\lambda\ From\_spec'$

- $\forall t\ cn\ st$
- $From\_spec' (from\ t)\ st$   
 $= projectTuples\ st\ t\ []$   
 $\wedge\ From\_spec' (correlate\_from\ cn\ t)\ st$   
 $= projectTuples\ st\ t\ (seq\ 1\ cn))$

$From\_spec$

**From\_name**  $\vdash ConstSpec$

$(\lambda\ From\_name'$

- $\forall t\ cn$
- $From\_name' (from\ t) = Table\_spec\ t$   
 $\wedge\ From\_name' (correlate\_from\ cn\ t)$

---

```

      = Table_spec t)
    From_name
Num_to_Int   ⊢ true
check_where_complete1
  ⊢ ∀ be b d
    • check_where_complete1 be b d
      = (if isVal be
         then check_where_complete (destVal be) b d
         else giveError (destError be))
all_false   ⊢ ConstSpec
    (λ all_false'
     • ∀ bs
       • all_false' bs
         ⇔ (if # bs = 0
            then true
            else if Head bs ⇔ true
              then false
              else all_false' (Tail bs)))
same        ⊢ ConstSpec
    (λ same'
     • ∀ bs
       • same' bs
         = (if # bs = 0
            then giveVal true
            else if # bs = 1
              then giveVal (Head bs)
              else if
                Head bs ⇔ destVal (same' (Tail bs))
              then giveVal (Head bs)
              else giveError (seq 1 ambiguousHaving)))
Value_monop ⊢ ConstSpec
    (λ Value_monop'
     • ∀ value st e op v
       • Value_monop' value (monop op v) st e
         = (let v' = value v st e
            in if isVal v'
              then
                giveVal (apply op (seq 1 (destVal v')))
              else giveError (destError v')))
Value_binop ⊢ ConstSpec
    (λ Value_binop'
     • ∀ value st e op v1 v2
       • Value_binop' value (binop op (v1, v2)) st e
         = (let v'1 = value v1 st e
            and v'2 = value v2 st e

```

---

```

in if isVal v'_1
then
  if isVal v'_2
  then
    giveVal
      (apply
        op
        [destVal v'_1; destVal v'_2])
      else giveError (destError v'_2)
  else if isVal v'_2
  then giveError (destError v'_1)
  else
    giveError
      (destError v'_1 @ destError v'_2)))

```

*Value\_binop*

**Value\_set\_func\_all**

```

⊢ ConstSpec
(λ Value_set_func_all'
  • ∀ value st e op v
  • Value_set_func_all'
    value
    (set_func_all op v)
    st
    e
  = (let all_res
      = promote
      (λ t
        • value
          v
          st
          (MkEnv t (E_group e)))
      (E_group e)
    in if isVal all_res
    then giveVal (apply op (destVal all_res))
    else giveError (destError all_res)))

```

*Value\_set\_func\_all*

**Value\_all\_binop**

```

⊢ ConstSpec
(λ Value_all_binop'
  • ∀ t_l_c t_l value st e op v tl
  • Value_all_binop'
    t_l_c
    t_l
    value
    (all_binop op v tl)
    st
    e
  = (let tlc = t_l_c tl st e

```

```

and calc
= (let ans = value v st e
   in promote
    (λ d
     • if isVal ans
       then
         giveVal
           (apply
            op
            ([destVal ans]
             @ [d])))
       else
         giveError (destError ans)))
in (((t_l tl st e * one_col) * calc
     *5 apply And)
    * take_data)
* check_where_complete1 tlc true))

```

Value\_all\_binop

### Value\_classification

```

⊢ ConstSpec
(λ Value_classification'
 • ∀ st e cs
 • Value_classification'
   (classification cs)
   st
   e
= (let ce
   = E_row e (Col_spec cs) *5 Dat_class
   in if isVal ce
   then
     giveVal
       (make_data (ClassVal (destVal ce)))
   else giveError (destError ce)))

```

Value\_classification

### Value<sub>p</sub>

```

⊢ ConstSpec
(λ Value'_p
 • ∀ t_l_c t_l st e i s r m int cl c op v v1 v2
   tl cs
 • Value'_p t_l_c t_l denote_null st e
   = giveVal
     (MkData
      userClearance
      (NullItemItem maybe))
 ∧ Value'_p t_l_c t_l denote_void st e
   = giveVal (make_data VoidVal)
 ∧ Value'_p t_l_c t_l denote_true st e
   = giveVal (make_data (BoolVal true))
 ∧ Value'_p t_l_c t_l denote_false st e

```

---


$$\begin{aligned}
&= \text{giveVal } (\text{make\_data } (\text{BoolVal } \text{false})) \\
\wedge &\text{Value}'_p \\
&\quad t\_l\_c \\
&\quad t\_l \\
&\quad (\text{denote\_integer } i) \\
&\quad st \\
&\quad e \\
&= \text{giveVal } (\text{make\_data } (\text{IntVal } i)) \\
\wedge &\text{Value}'_p \ t\_l\_c \ t\_l \ (\text{denote\_string } s) \ st \ e \\
&= \text{giveVal } (\text{make\_data } (\text{StringVal } s)) \\
\wedge &\text{Value}'_p \ t\_l\_c \ t\_l \ (\text{denote\_float } r) \ st \ e \\
&= \text{giveVal } (\text{make\_data } (\text{FloatVal } r)) \\
\wedge &\text{Value}'_p \ t\_l\_c \ t\_l \ (\text{denote\_time } m) \ st \ e \\
&= \text{giveVal } (\text{make\_data } (\text{TimeVal } m)) \\
\wedge &\text{Value}'_p \\
&\quad t\_l\_c \\
&\quad t\_l \\
&\quad (\text{denote\_interval } \text{int}) \\
&\quad st \\
&\quad e \\
&= \text{giveVal } (\text{make\_data } (\text{IntervalVal } \text{int})) \\
\wedge &\text{Value}'_p \ t\_l\_c \ t\_l \ (\text{denote\_class } \text{cl}) \ st \ e \\
&= \text{giveVal } (\text{make\_data } (\text{ClassVal } \text{cl})) \\
\wedge &\text{Value}'_p \ t\_l\_c \ t\_l \ (\text{denote\_code } c) \ st \ e \\
&= \text{giveVal } (\text{make\_data } (\text{CodeVal } c)) \\
\wedge &\text{Value}'_p \ t\_l\_c \ t\_l \ (\text{monop } \text{op } v) \ st \ e \\
&= \text{Value\_monop} \\
&\quad (\text{Value}'_p \ t\_l\_c \ t\_l) \\
&\quad (\text{monop } \text{op } v) \\
&\quad st \\
&\quad e \\
\wedge &\text{Value}'_p \\
&\quad t\_l\_c \\
&\quad t\_l \\
&\quad (\text{binop } \text{op } (v_1, v_2)) \\
&\quad st \\
&\quad e \\
&= \text{Value\_binop} \\
&\quad (\text{Value}'_p \ t\_l\_c \ t\_l) \\
&\quad (\text{binop } \text{op } (v_1, v_2)) \\
&\quad st \\
&\quad e \\
\wedge &\text{Value}'_p \\
&\quad t\_l\_c \\
&\quad t\_l \\
&\quad (\text{set\_func\_all } \text{op } v) \\
&\quad st
\end{aligned}$$


---

---


$$\begin{aligned}
& e \\
& = \text{Value\_set\_func\_all} \\
& \quad (\text{Value}'_p \ t\_l\_c \ t\_l) \\
& \quad (\text{set\_func\_all} \ op \ v) \\
& \quad st \\
& e \\
\wedge & \text{Value}'_p \ t\_l\_c \ t\_l \ \text{count\_all} \ st \ e \\
& = \text{giveVal} \\
& \quad (\text{make\_data} \\
& \quad \quad (\text{IntVal} \\
& \quad \quad \quad (\text{Num\_to\_Int} \ (\# \ (\text{E\_group} \ e)))))) \\
\wedge & \text{Value}'_p \\
& \quad t\_l\_c \\
& \quad t\_l \\
& \quad (\text{all\_binop} \ op \ v \ tl) \\
& \quad st \\
& e \\
& = \text{Value\_all\_binop} \\
& \quad t\_l\_c \\
& \quad t\_l \\
& \quad (\text{Value}'_p \ t\_l\_c \ t\_l) \\
& \quad (\text{all\_binop} \ op \ v \ tl) \\
& \quad st \\
& e \\
\wedge & \text{Value}'_p \ t\_l\_c \ t\_l \ (\text{contents} \ cs) \ st \ e \\
& = \text{E\_row} \ e \ (\text{Col\_spec} \ cs) \\
\wedge & \text{Value}'_p \\
& \quad t\_l\_c \\
& \quad t\_l \\
& \quad (\text{classification} \ cs) \\
& \quad st \\
& e \\
& = \text{Value\_classification} \\
& \quad (\text{classification} \ cs) \\
& \quad st \\
& e \\
\wedge & \text{Value}'_p \ t\_l\_c \ t\_l \ \text{user} \ st \ e \\
& = \text{giveVal} \\
& \quad (\text{make\_data} \ (\text{StringVal} \ \text{userName})) \\
\wedge & \text{Value}'_p \ t\_l\_c \ t\_l \ \text{clearance} \ st \ e \\
& = \text{giveVal} \\
& \quad (\text{make\_data} \ (\text{ClassVal} \ \text{userClearance})) \\
\wedge & \text{Value}'_p \ t\_l\_c \ t\_l \ \text{current\_time} \ st \ e \\
& = \text{giveVal} \ (\text{make\_data} \ (\text{TimeVal} \ \text{timeNow}))
\end{aligned}$$

*Value<sub>p</sub>*

**Tuple\_list\_complete<sub>p</sub>**  
 $\vdash \text{ConstSpec}$

---

```

(λ Tuple_list_complete'_p
  • ∀ value st e sel from_list where group having
  • Tuple_list_complete'_p
    value
    (all_tuples
      sel
      from_list
      where
      group
      having)
    st
    e
  = (let froms
      = promote (λ fr • From_spec fr st)
      * join
      and ws g
      = promote
      (λ t
        • value where st (MkEnv t g)
        *5 isNotCleared)
      g
      in froms from_list * ws *5 all_false)
  ∧ Tuple_list_complete'_p
    value
    (distinct_tuples
      sel
      from_list
      where
      group
      having)
  = Tuple_list_complete'_p
    value
    (all_tuples
      sel
      from_list
      where
      group
      having)
  ∧ Tuple_list_complete'_p
    value
    (evaluate
      sel
      from_list
      where
      group
      having)
  = Tuple_list_complete'_p

```

---

---

```

      value
      (all_tuples
        sel
        from_list
        where
        group
        having))
    Tuple_list_complete_p
Tuple_list_all_tuples
  ⊢ ConstSpec
  (λ Tuple_list_all_tuples'
    • ∀ value s_l st e sel from_list where group
      having
    • Tuple_list_all_tuples'
      value
      s_l
      (all_tuples
        sel
        from_list
        where
        group
        having)
      st
      e
    = (let froms
      = promote (λ fr • From_spec fr st)
      * join
      and selr env
      = (let tabs = Map From_name from_list
        in s_l sel st env tabs)
      and outer e
      = Map
      (λ ei
        • MkEnv
          (E_row e ** E_row ei)
          (E_row e ** E_group ei))
      and where_filter ts
      = (let ws g
        = (let ev t
          = (let elim
            = (let eliminate
              s
              env
              h
              = promote
                (((λ t • value h s (MkEnv t (E_group env))) * seq 1)
                  * applyNot)
                  * resultBool)

```

---

```

(E_group env)
* same in eliminate st (MkEnv t g) having)
    in if isError elim
    then elim
    else if
        destVal elim ⇔ true
    then giveVal false
    else
        value
        where
            st
            (MkEnv t g)
            * resultBool)
        in promote ev g)
    in ts *7 ws ts *5 extract)
in ((froms from_list * where_filter
    *5 engroup group)
    *5 outer e)
    * promote selr))
Tuple_list_all_tuples

```

**Tuple\_list\_evaluate**

```

⊢ ConstSpec
(λ Tuple_list_evaluate'
    • ∀ value s_l st e sel from_list where group
        having
        • Tuple_list_evaluate'
            value
            s_l
            (evaluate
                sel
                from_list
                where
                group
                having)
            st
            e
    = (let froms
        = promote (λ fr • From_spec fr st)
        * join
        and selg env
        = (let tabs = Map From_name from_list
            in if isVal (s_l sel st env tabs)
            then
                giveVal
                (MkGroupedResult
                    (destVal
                        (s_l sel st env tabs))
                    (E_group env))

```

---

```

    else
      giveError
        (destError
          (s_l sel st env tabs)))
  and outer e
  = Map
    (λ ei
      • MkEnv
        (E_row e ** E_row ei)
        (E_row e ** E_group ei))
  and ev grs
  = (if
    distinct (Map G_group grs)
      = Map G_group (distinct grs)
    then giveVal (Map G_res grs)
    else
      giveError
        (seq 1 ambiguousEvaluate))
  and where_filter ts
  = (let ws g
      = promote
        (λ t
          • value
            where
              st
              (MkEnv t g)
            * resultBool)

          g
          in ts *7 ws ts *5 extract)
    in (((froms from_list * where_filter
      *5 engroup group)
      *5 outer e)
      * promote selg)
      * ev))
  Tuple_list_evaluate
Tuple_listp ⊢ ConstSpec
  (λ Tuple_list'p
    • ∀ value s_l st e sel from_list where group
      having
    • Tuple_list'p
      value
      s_l
      (all_tuples
        sel
        from_list
        where
        group
        having))

```

---

```

      st
      e
    = Tuple_list_all_tuples
      value
      s_l
      (all_tuples
        sel
        from_list
        where
        group
        having)
      st
      e
    ∧ Tuple_list'_p
      value
      s_l
      (distinct_tuples
        sel
        from_list
        where
        group
        having)
      st
      e
    = Tuple_list'_p
      value
      s_l
      (all_tuples
        sel
        from_list
        where
        group
        having)
      st
      e
    *5 distinct
    ∧ Tuple_list'_p
      value
      s_l
      (evaluate
        sel
        from_list
        where
        group
        having)
      st
      e
    = Tuple_list_evaluate

```

---

```

      value
      s_l
      (evaluate
        sel
        from_list
        where
        group
        having)
      st
      e)
  Tuple_list_p
Select_list_p
  ⊢ ConstSpec
  (λ Select_list'_p
    • ∀ value st e vl ts
    • Select_list'_p value all_columns st e
      = (let vals
          = denote_col_spec
            % contents
            % (λ v • value v st e)
          and col t
          = enseq
            (((lookup st t * colposns)
              * CS_ide)
              * (λ i • t @ [i]))
            in (Map col % Flat) % promote vals)
    ∧ Select_list'_p
      value
      (select_values vl)
      st
      e
      ts
      = promote (λ v • value v st e) vl)
  Select_list_p

```

**Value****Tuple\_list\_complete****Tuple\_list****Select\_list**

```

  ⊢ ConstSpec
  (λ
    (Value', Tuple_list_complete', Tuple_list',
     Select_list')
    • Value'
      = Value_p Tuple_list_complete' Tuple_list'
    ∧ Tuple_list_complete'
      = Tuple_list_complete_p Value'
    ∧ Tuple_list'
      = Tuple_list_p Value' Select_list'
    ∧ Select_list' = Select_list_p Value')

```

---

*(Value, Tuple\_list\_complete, Tuple\_list, Select\_list)*

**Set\_clause**      $\vdash$  *ConstSpec*  
                    $(\lambda$  *Set\_clause'*  
                   •  $\forall$  *st e col vv vc id*  
                   • *Set\_clause'* (*set\_value col vv*) *st e id*  
                   = (*let column = Col\_name col*  
                   and *exp = Value vv st e*  
                   in if *id = column*  
                   then *exp \* take\_data \*<sub>5</sub> DataUpdate*  
                   else *giveError (seq 1 error)*)  
                    $\wedge$  *Set\_clause'* (*set\_class col vc*) *st e id*  
                   = (*let column = Col\_name col*  
                   and *exp = Value vc st e*  
                   in if *id = column*  
                   then *exp \* resultClass \*<sub>5</sub> ClassUpdate*  
                   else *giveError (seq 1 error)*)  
                    $\wedge$  *Set\_clause'*  
                   (*set\_class\_and\_value col vv vc*)  
                   *st*  
                   *e*  
                   *id*  
                   = (*let column = Col\_name col*  
                   and *value*  
                   = *Value vv st e \* take\_data*  
                    $*_{5}$  *Dat\_item*  
                   and *clas = Value vc st e \* resultClass*  
                   in if *id = column*  
                   then  
                   (*clas, value*)  $*$  *MkData \*<sub>5</sub> DataUpdate*  
                   else *giveError (seq 1 error)*))  
                   *Set\_clause*

**Classified\_value**  
                    $\vdash$  *ConstSpec*  
                    $(\lambda$  *Classified\_value'*  
                   •  $\forall$  *st e v vc*  
                   • *Classified\_value'* (*classify v vc*) *st e*  
                   = (*let resv = Value v st e*  
                   and *resc = Value vc st e*  
                   in (*resc \* resultClass,*  
                   *resv \* take\_data \*<sub>5</sub> Dat\_item*)  
                    $*$  *MkData*)  
                    $\wedge$  *Classified\_value'*  
                   (*classify\_default v*)  
                   *st*  
                   *e*  
                   = (*let combine c d*  
                   = *MkData c (Dat\_item d)*

---

---

```

    and newDat
      = Value v st e
      * (λ d
        • if isCleared d
          then giveVal d
          else
            giveError (seq 1 notCleared))
    and newClass = userClearance
    in newDat *5 combine newClass))
  Classified_value
Insert_list ⊢ ConstSpec
  (λ Insert_list'
    • ∀ st tl cvl il1 il2
    • Insert_list' (insert_tuples tl) st
    = (let check d
      = (let tlc
        = Tuple_list_complete
          tl
          st
          emptyEnv
        in if isVal tlc
        then
          if destVal tlc ⇔ true
          then giveVal d
          else
            giveError (seq 1 notCleared)
        else
          giveError (seq 1 notCleared))
      in Tuple_list tl st emptyEnv
      * promote (promote take_data))
    ∧ Insert_list' (tuple cvl) st
    = (let d_or_err
      = promote
        (λ cv
          • Classified_value
            cv
            st
            emptyEnv)
        cvl
      in if isVal d_or_err
      then giveVal (seq 1 (destVal d_or_err))
      else giveError (destError d_or_err))
    ∧ Insert_list' (union il1 il2) st
    = (Insert_list' il1 st
      & Insert_list' il2 st))
  Insert_list
processIntegrity
  ⊢ ∀ t

```

- *processIntegrity t*  
 = (if *checkGroup t CC\_uniform checkUniform*  $\Leftrightarrow$  true  
 then *noErrors*  
 else *seq 1 nonUniformValues*)  
 @ (if  
   *checkGroup t CC\_unique checkUniqueness*  
    $\Leftrightarrow$  true  
 then *noErrors*  
 else *seq 1 nonUniqueValues*)  
 @ (if *checkIntegrity t checkFieldClasses*  $\Leftrightarrow$  true  
 then *noErrors*  
 else *seq 1 fieldClassOutOfRange*)  
 @ (if *checkIntegrity t checkType*  $\Leftrightarrow$  true  
 then *noErrors*  
 else *seq 1 wrongType*)  
 @ (if *checkIntegrity t checkNulls*  $\Leftrightarrow$  true  
 then *noErrors*  
 else *seq 1 noNulls*)

**processInsert**

- $\vdash \forall c t cns il st$
- *processInsert c t cns il st*  
 = (let *spec = lookup st (Table\_spec t)*  
 and *givenNames = Map Col\_name cns*  
 in let *defaults*  
   = (*spec \* colposns*) \* *CS\_default*  
 in let *givenNums*  
   = *promote*  
   ((*spec \* colspecs*) \* *CS\_posn*)  
   *givenNames*  
 in let *colNums*  
   = *Map*  
   *CS\_posn*  
   (*enseq (spec \* colposns)*)  
 in let *map nums ds n*  
   = (if *isVal (find nums n)*  
   then  
     *giveVal*  
     (*n,*  
       *elem*  
       (*destVal*  
       (*find nums n*))  
       *ds*)  
   else if *isVal (defaults n)*  
   then  
     *giveVal*  
     (*n, destVal (defaults n)*)  
   else  
     *giveError*

---

```

      (destError (defaults n)))
in let addDefaults ds
  = (if # givenNames = 0
    then
      promote
        (map colNums ds)
        colNums
    else if isVal givenNums
    then
      promote
        (map
          (destVal givenNums)
          ds)
        colNums
    else
      giveError
        (destError givenNums))
in let width ds
  = (if # givenNames = 0
    then
      if # colNums = # ds
      then giveVal ds
      else
        giveError (seq 1 tooWide)
    else if # givenNames = # ds
    then giveVal ds
    else
      giveError (seq 1 tooWide))
in let effect
  = (if isVal spec
    then
      if
        TS_maxRow
          (destVal spec)
          dominates userClearance
      then
        giveError
          (seq
            1
            rowClassTooLow)
      else
        Insert_list il st
          * promote
            (width
              * addDefaults
              * Elems)
    else
      giveError

```

---

---

```

        (destError spec))
    in if isError effect
    then giveError (destError effect)
    else
      (let newSpec
        = replaceRows
          (destVal spec)
          (TS_rows (destVal spec)
            @ Map
              (MkRow c)
              (destVal effect)))
      in let integrity
        = processIntegrity
          newSpec
      in if ¬ # integrity = 0
      then giveError integrity
      else
        giveVal
          (InsertEffect
            (Table_spec t,
              destVal
                effect))))
processDelete
  ⊢ ∀ f w st
  • processDelete f w st
    = (let table = From_name f
      in let spec = lookup st table
      in let ts = From_spec f st
      in let where
        = (if isVal ts
          then
            promote
              (λ t
                • Value
                  w
                  st
                  (MkEnv t (destVal ts)))
              (destVal ts)
            else giveError (destError ts))
      in let whereBools
        = where * promote resultBool
      in let complete
        ⇔ (if isError where
          then true
          else
            checkComplete
              (destVal where))
      in let doomed

```

---

---

```

= (if isError whereBools
  then
    giveError
      (destError whereBools)
  else
    giveVal
      (Dom
        (ListRel
          (destVal
            whereBools)
            ▷ {true})))
in if isError doomed
then giveError (destError doomed)
else if complete
then
  giveVal
    (DeleteEffect
      (From_name f,
        destVal doomed))
else
  giveError
    (seq 1 mayNotBeComplete))

```

**convert**  $\vdash \forall f$

- *convert f*

```

= (if
  ∃ n
  • isError (f n)
    ∧ ¬ destError (f n) = seq 1 error
  then
    giveError
      (RelList
        (Enumerate
          (∪
            {x
              | ∃ y
                • isError (f y)
                  ∧ ¬ destError (f y)
                    = seq 1 error
                  ∧ x
                    = Elms
                      (destError
                        (f y))})))
    else giveVal {(id, upd)|f id = giveVal upd})

```

**updateRowList**  $\vdash \forall rl\ nnu$

- *updateRowList rl nnu*

```

= (let updateD (u, d)
  = (if isItem u

```

---

---

```

    then MkData (Dat_class d) (destItem u)
    else if isClass u
    then MkData (destClass u) (Dat_item d)
    else destData u
  in let updateR (nu, r)
    = MkRow
      (R_exist r)
      (R_data r
        ⊕ RelCombine nu (R_data r)
        § Graph updateD)
  in RelList
    (ListRel rl
      ⊕ RelCombine nnu (ListRel rl)
      § Graph updateR))

```

**processUpdate**

```

⊢ ∀ f ss w g h st
  • processUpdate f ss w g h st
    = (let table = From_name f
      in let spec = lookup st table
        in let ts = From_spec f st
          in let whereResults
            = (let ev e
              = (let elim
                = (let eliminate s env h
                  = promote
                    (((λ t
                      • Value
                    h
                    s
                    (MkEnv t (E_group env)))
                    * seq 1)
                    * applyNot)
                    (E_group env)
                    * same
                    in eliminate st e h)
                in if isError elim
                  then
                    giveError (destError elim)
                  else if destVal elim ⇔ true
                  then
                    giveVal
                      (make_data
                        (BoolVal false))
                    else Value w st e)
            in if isVal ts
              then
                promote

```

---

```

      ev
      (engroup g (destVal ts))
      else giveError (destError ts))
in let whereBools
  = whereResults * promote resultBool
in let complete
  ⇔ (if isError whereResults
    then true
    else
      checkComplete
      (destVal whereResults))
in let sets e
  = ((λ c
    • giveError (seq 1 error))
    & Map
    (λ s • Set_clause s st e)
    ss)
in let getEnv n
  = (if isVal ts
    then
      giveVal
      (MkEnv
      (Nth
      (destVal ts)
      n)
      (destVal ts))
    else
      giveError (destError ts))
in let rowUpd n
  = convert
  (((spec * colposns)
   * CS_ide)
   * getEnv n
   * sets)
in let where
  = (if isError whereBools
    then
      giveError
      (destError
      whereBools)
    else
      giveVal
      (Dom
      (ListRel
      (destVal
      whereBools)
      ▷ {true}))))
in if isError where

```

---

---

```

then
  giveError (destError where)
else if complete
then
  let us
    = {(n, nu)
      | n ∈ destVal where
        ∧ nu = rowUpd n}
  in if
    ∃ pr
    • pr ∈ us
      ∧ isError (Snd pr)
  then
    giveError
      (RelList
        (Enumerate
          (∪
            {x|∃ y• y ∈ us ∧ x = Elems (destError (Snd y))}))
        else
          (let ups
            = {(n, nu)
              | n
                ∈ destVal where
                  = destVal (rowUpd n)}
              ∧ nu
                ∈ destVal (rowUpd n)}
            in let oldRows
              = TS_rows
                (destVal
                  spec)
              in let newSpec
                = replaceRows
                  (destVal spec)
                  (updateRowList oldRows ups)
              in let integrity
                in if
                  ¬ # integrity
                then
                  giveError
                else
                  giveVal
                  (UpdateEffect (table, ups)))
            else
              giveError
                (seq 1 mayNotBeComplete))

```

**processSelect**

---

```

    ⊢ ∀ tl st
      • processSelect tl st
        = (let retrieved = Tuple_list tl st emptyEnv
           and tlc = Tuple_list_complete tl st emptyEnv
           in if isError retrieved
            then (SelectEffect [], destError retrieved)
            else if isVal tlc
                then
                  if destVal tlc ⇔ true
                  then (SelectEffect (destVal retrieved), [])
                  else
                     (SelectEffect (destVal retrieved),
                      seq 1 mayNotBeComplete)
            else (SelectEffect [], destError tlc))

nullUpdate ⊢ true
processQuery ⊢ ConstSpec
  (λ processQuery'
    • ∀ c st t cns il f w ss g h tl
      • processQuery' (insert t cns il, c, st)
        = (let iorerr
           = processInsert c t cns il st
           in if isVal iorerr
            then (destVal iorerr, [])
            else (nullUpdate, destError iorerr))
      ∧ processQuery' (delete f w, c, st)
        = (let dorerr = processDelete f w st
           in if isVal dorerr
            then (destVal dorerr, [])
            else (nullUpdate, destError dorerr))
      ∧ processQuery'
        (update f ss w g h, c, st)
        = (let uorerr
           = processUpdate f ss w g h st
           in if isVal uorerr
            then (destVal uorerr, [])
            else (nullUpdate, destError uorerr))
      ∧ processQuery' (select tl, c, st)
        = processSelect tl st)
    processQuery
  
```

## 19 THE THEORY fef015

### 19.1 Parents

*fef013*

## 19.2 Theorems

**updateStateR\_updateState\_lemma**

$$\begin{aligned} &\vdash (\forall \text{ clear } s \ u \\ &\quad \bullet \text{ isState } s \\ &\quad \Rightarrow \text{ isState } (Fst \ (\text{updateStateR } (\text{clear}, u, s))) \\ &\quad \Rightarrow (\text{hide}, \text{updateState}) \in \text{secureUpdate} \end{aligned}$$
**isState\_insertQuery\_lemma**

$$\begin{aligned} &\vdash \forall c \ i \ ns \ s \ ts \\ &\quad \bullet \text{ isState } s \wedge \text{tabExists } c \ i \ s \\ &\quad \Rightarrow \text{ isState} \\ &\quad \quad (Fst \\ &\quad \quad \quad (\text{insertQuery} \\ &\quad \quad \quad \quad (c, (i, ds), s, \text{getTable } i \ s))) \end{aligned}$$
**isState\_deleteQuery\_lemma**

$$\begin{aligned} &\vdash \forall c \ i \ ns \ s \ ts \\ &\quad \bullet \text{ isState } s \wedge \text{tabExists } c \ i \ s \\ &\quad \Rightarrow \text{ isState} \\ &\quad \quad (\text{deleteQuery } (c, (i, ns), s, \text{getTable } i \ s)) \end{aligned}$$
**isState\_updateRow\_lemma**

$$\begin{aligned} &\vdash \forall r \ c \ tc \ u \\ &\quad \bullet r \in \text{RowS} \wedge \text{isVal } (\text{updateRow } c \ tc \ (u, r)) \\ &\quad \Rightarrow \text{destVal } (\text{updateRow } c \ tc \ (u, r)) \in \text{RowS} \end{aligned}$$
**isState\_updateRows\_lemma**

$$\begin{aligned} &\vdash \forall r \ c \ t \ u \\ &\quad \bullet u \in \text{Functional} \\ &\quad \wedge ((\text{RelCombine} \\ &\quad \quad (\text{revealRow } c \ t \ \sim \ \S \ u) \\ &\quad \quad (\text{ListRel } (TS\_rows \ t)) \\ &\quad \quad \S \ \text{Graph } (\text{updateRow } c \ (TS\_class \ t))) \\ &\quad \quad \triangleright \{x \mid \text{isError } x\} \\ &\quad \quad \S \ \text{Graph } \text{destError} \\ &\quad \quad = \{\} \\ &\quad \wedge (\forall r \bullet r \in_l \ TS\_rows \ t \Rightarrow r \in \text{RowS}) \\ &\Rightarrow r \\ &\quad \in_l \ \text{RelList} \\ &\quad \quad (\text{ListRel } (TS\_rows \ t) \\ &\quad \quad \oplus (\text{RelCombine} \\ &\quad \quad \quad (\text{revealRow } c \ t \ \sim \ \S \ u) \\ &\quad \quad \quad (\text{ListRel } (TS\_rows \ t)) \\ &\quad \quad \quad \S \ \text{Graph } (\text{updateRow } c \ (TS\_class \ t))) \\ &\quad \quad \quad \S \ \text{Graph } \text{destVal}) \\ &\Rightarrow r \in \text{RowS} \end{aligned}$$
**isState\_updateQuery\_lemma**

$$\begin{aligned} &\vdash \forall c \ i \ us \ s \ ts \\ &\quad \bullet \text{ isState } s \wedge \text{tabExists } c \ i \ s \\ &\quad \Rightarrow \text{ isState} \\ &\quad \quad (Fst \end{aligned}$$

$$(updateQuery \\ (c, (i, us), s, getTable i s))$$
**updateStateR\_lemma**

$$\begin{aligned} &\vdash \forall clear\ s \\ &\quad \bullet\ isState\ s \\ &\quad \Rightarrow isState\ (Fst\ (updateStateR\ (clear, u, s))) \end{aligned}$$
**secureSSQL**

$$\vdash behaviours\ SSQLam \in secure$$
**20 THE THEORY fef021****20.1 Parents**

$$fef024$$
**20.2 Children**

$$fef025$$
**20.3 Constants****item\_sterling**

$$Item \rightarrow Item$$
**class\_bottom**

$$Data \rightarrow Data$$
**set\_bottom<sub>d</sub>**

$$Worth \leftrightarrow Data \rightarrow Worth \leftrightarrow Data$$
**set\_bottom<sub>u</sub>**

$$Worth \leftrightarrow Update \rightarrow Worth \leftrightarrow Update$$
**State<sub>tS</sub>**

$$State\ \mathbb{P}$$
**isState<sub>t</sub>**

$$State \rightarrow Bool$$
**absState<sub>t</sub>**

$$State \rightarrow State_t$$
**repState<sub>t</sub>**

$$State_t \rightarrow State$$
**processQuery<sub>t</sub>**

$$Process_t$$
**updateState<sub>t</sub>**

$$Ustate_t$$
**MkTf<sub>t</sub>**

$$Process_t \rightarrow Ustate_t \rightarrow tf_t$$
**TSQLtf**

$$tf_t$$
**20.4 Types**

$$State_t$$
**20.5 Type Abbreviations**

$$\mathbf{Process}_t \quad Process_t$$

$$\mathbf{Ustate}_t \quad Ustate_t$$

$$\mathbf{tf}_t \quad tf_t$$

## 20.6 Definitions

### item\_sterling

$$\vdash \forall i$$

- *item\_sterling* *i*  
 = (if *isNullItem* *i*  
 then *i*  
 else  
 (let *v* = *destValuedItem* *i*  
 in *ValuedItemItem*  
 (*MkValuedItem* *sterling* (*VI\_val* *v*))))

### class\_bottom

$$\vdash \forall d$$

- *class\_bottom* *d*  
 = (let *i* = *Dat\_item* *d*  
 in *MkData* *lattice\_bottom* (*item\_sterling* *i*))

### set\_bottom<sub>d</sub>

$$\vdash \forall nd$$

- *set\_bottom<sub>d</sub>* *nd*  
 = {(*n*, *d*) | *n* ∈ *Dom* *nd* ∧ *d* = *class\_bottom* (*nd* @ *n*)}

### set\_bottom<sub>u</sub>

$$\vdash \forall nu$$

- *set\_bottom<sub>u</sub>* *nu*  
 = {(*n*, *u*)  
 | *n* ∈ *Dom* *nu*  
 ∧ *u*  
 = (let *u'* = *nu* @ *n*  
 in if *isItem* *u'*  
 then  
   *ItemUpdate*  
   (*item\_sterling* (*destItem* *u'*))  
 else if *isClass* *u'*  
 then *ClassUpdate* *lattice\_bottom*  
 else  
   *DataUpdate*  
   (*class\_bottom* (*destData* *u'*)))}

### State<sub>tS</sub>

$$\vdash State_{tS}$$

$$= \{st$$

$$| \forall dir$$

- *dir* ∈ *Ran* (*repState* *st*)  
 ⇒ *Dir\_exist* *dir* = *lattice\_bottom*  
 ∧ *Dir\_class* *dir* = *lattice\_bottom*  
 ∧ (∀ *tab*
- *tab* ∈ *Ran* (*Dir\_tables* *dir*)  
 ⇒ *TS\_class* *tab* = *lattice\_bottom*  
 ∧ *TS\_maxRow* *tab* = *lattice\_bottom*  
 ∧ (∀ *col*
- *col* ∈ *TS\_colspecs* *tab*  
 ⇒ *CS\_min* *col* = *lattice\_bottom*  
 ∧ *CS\_max* *col* = *lattice\_bottom*  
 ∧ (∀ *cc*

---


$$\begin{aligned}
& \bullet cc \in \text{Ran } (TS\_cons \ tab) \\
& \Rightarrow CC\_exist \ cc \\
& \quad = \text{lattice\_bottom} \\
& \quad \wedge (\forall \ row \\
& \quad \bullet \ row \\
& \quad \quad \in \text{Elms} \\
& \quad \quad (TS\_rows \ tab) \\
& \quad \Rightarrow R\_exist \ row \\
& \quad \quad = \text{lattice\_bottom} \\
& \quad \quad \wedge (\forall \ data \\
& \quad \quad \bullet \ data \\
& \quad \quad \quad \in \text{Ran} \\
& \quad \quad (R\_data \ row) \\
& \quad \quad \quad \Rightarrow \text{Dat\_class} \\
& \quad \quad \quad \quad data \\
& \quad \quad = \text{lattice\_bottom} \\
& \quad \quad \quad \quad \wedge \text{isValuedItem} \\
& \quad \quad (Dat\_item \ data) \\
& \quad \quad \quad \Rightarrow \text{VI\_worth} \\
& \quad \quad (destValuedItem \ (Dat\_item \ data)) \\
& \quad \quad \quad = \text{sterling}})))))\}
\end{aligned}$$

**isState<sub>t</sub>**      $\vdash \forall s \bullet \text{isState}_t \ s \Leftrightarrow s \in \text{State}_tS$

**state<sub>t</sub>-type-def**      $\vdash \exists f \bullet \text{TypeDefn } \text{isState}_t \ f$

**repState<sub>t</sub>**  
**absState<sub>t</sub>**      $\vdash \text{ConstSpec}$   
 $(\lambda (\text{repState}'_t, \text{absState}'_t)$   
 $\bullet (\forall a \bullet \text{absState}'_t (\text{repState}'_t \ a) = a)$   
 $\quad \wedge (\forall r$   
 $\quad \bullet \text{isState}_t \ r$   
 $\quad \quad \Rightarrow \text{repState}'_t (\text{absState}'_t \ r) = r)$   
 $\quad \wedge (\forall a_1 \ a_2$   
 $\quad \bullet \text{repState}'_t \ a_1 = \text{repState}'_t \ a_2$   
 $\quad \quad \Leftrightarrow a_1 = a_2)$   
 $\quad \wedge (\forall r_1 \ r_2$   
 $\quad \bullet \text{isState}_t \ r_1 \wedge \text{isState}_t \ r_2$   
 $\quad \quad \Rightarrow (\text{absState}'_t \ r_1 = \text{absState}'_t \ r_2$   
 $\quad \quad \quad \Leftrightarrow r_1 = r_2))$   
 $\quad \wedge (\forall s \bullet \text{isState}_t (\text{repState}'_t \ s)))$   
 $(\text{repState}_t, \text{absState}_t)$

**processQuery<sub>t</sub>**      $\vdash \forall q \ s$   
 $\bullet \text{processQuery}_t (q, s)$   
 $\quad = (\text{let } (ef, es)$   
 $\quad \quad = \text{processQuery}$   
 $\quad \quad \quad (q, \text{lattice\_bottom}, \text{repState}_t \ s)$   
 $\quad \quad \text{in let } ef'$   
 $\quad \quad \quad = (\text{if } \text{isInsert } ef$

---

---

```

then
  let (t, ndl) = destInsert ef
    in InsertEffect
      (t, Map set_bottom_d ndl)
else if isDelete ef
then ef
else if isUpdate ef
then
  let (t, nnu) = destUpdate ef
    in UpdateEffect
      (t,
        {(n, nu)
         | n ∈ Dom nnu
           ∧ nu
             = set_bottom_u
               (nnu @ n)})
else
  (let dll = destSelect ef
    in SelectEffect
      (Map (Map class_bottom) dll))
in (ef', es))

```

**updateState<sub>t</sub>**

```

⊢ ∀ efes s
  • updateStatet (efes, s)
    = (let (s', c, out)
      = updateState
        (lattice_bottom, efes, repStatet s)
      in (absStatet s', out))

```

**MkTf<sub>t</sub>**

```

⊢ ∀ p u q s • MkTft p u (q, s) = u (p (q, s), s)

```

**TSQLtf**

```

⊢ TSQLtf = MkTft processQueryt updateStatet

```

## 20.7 Theorems

**repState\_consistent****absState\_consistent**

```

⊢ Consistent
  (λ (repState', absState')
    • (∀ a • absState' (repState' a) = a)
      ∧ (∀ r
        • isState r ⇒ repState' (absState' r) = r)
      ∧ (∀ a1 a2
        • repState' a1 = repState' a2 ⇔ a1 = a2)
      ∧ (∀ r1 r2
        • isState r1 ∧ isState r2
          ⇒ (absState' r1 = absState' r2
            ⇔ r1 = r2))
      ∧ (∀ s • isState (repState' s)))

```

**repState<sub>t</sub>\_consistent**

**absState<sub>t</sub>\_consistent**⊢ *Consistent*

$$\begin{aligned}
& (\lambda (\text{repState}'_t, \text{absState}'_t) \\
& \bullet (\forall a \bullet \text{absState}'_t (\text{repState}'_t a) = a) \\
& \quad \wedge (\forall r \\
& \quad \bullet \text{isState}_t r \\
& \quad \quad \Rightarrow \text{repState}'_t (\text{absState}'_t r) = r) \\
& \quad \wedge (\forall a_1 a_2 \\
& \quad \bullet \text{repState}'_t a_1 = \text{repState}'_t a_2 \\
& \quad \quad \Leftrightarrow a_1 = a_2) \\
& \quad \wedge (\forall r_1 r_2 \\
& \quad \bullet \text{isState}_t r_1 \wedge \text{isState}_t r_2 \\
& \quad \quad \Rightarrow (\text{absState}'_t r_1 = \text{absState}'_t r_2 \\
& \quad \quad \quad \Leftrightarrow r_1 = r_2)) \\
& \quad \wedge (\forall s \bullet \text{isState}_t (\text{repState}'_t s))
\end{aligned}$$
**21 THE THEORY fef022****21.1 Parents***fef006***21.2 Children***fef024***21.3 Constants**

**mkTff** ('TSQL\_QUERY, 'ST) DBMS\_TYPE  
→ ('TSQL\_QUERY, 'PARS) STP\_TYPE  
→ 'PARS FILTER\_TYPE  
→ 'ST TF\_TYPE

**istate<sub>f</sub>** (State → 'ST) → 'ST

**FE\_SWORD** (State → 'ST)  
→ ('TSQL\_QUERY, 'ST) DBMS\_TYPE  
→ ('TSQL\_QUERY, 'PARS) STP\_TYPE  
→ 'PARS FILTER\_TYPE  
→ (Query, ANSWER) BEHAVIOURS

**ok\_to\_proceed** ('TSQL\_QUERY, 'ST) DBMS\_TYPE  
→ ('TSQL\_QUERY  
× 'TSQL\_QUERY + Maybe  
× 'PARS)  
+ Errors  
→ 'ST  
→ Bool

**subsys\_secureA**

(*State* → '*ST*)  
 → ('*TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ↔ ('*TSQL\_QUERY*, '*PARS*) *STP\_TYPE*

**subsys\_secureB**

(*State* → '*ST*)  
 → ('*TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ↔ ('*TSQL\_QUERY*, '*PARS*) *STP\_TYPE*

**subsys\_secureC**

(*State* → '*ST*)  
 → ('*TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ↔ ('*TSQL\_QUERY*, '*PARS*) *STP\_TYPE*

**subsys\_secureD**

(*State* → '*ST*)  
 → ('*TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ↔ ('*TSQL\_QUERY*, '*PARS*) *STP\_TYPE*

**subsys\_secureE**

(*State* → '*ST*)  
 → ('*TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ↔ ((''*TSQL\_QUERY*, '*PARS*) *STP\_TYPE*  
 × '*PARS\_FILTER\_TYPE*)

**subsys\_secure**

(*State* → '*ST*)  
 → ('*TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ↔ ((''*TSQL\_QUERY*, '*PARS*) *STP\_TYPE*  
 × '*PARS\_FILTER\_TYPE*)

**21.4 Type Abbreviations**

**ANSWER**      *ANSWER*  
 ('*TSQL\_QUERY*, '*ST*) **DBMS\_TYPE**  
                   (*'TSQL\_QUERY*, '*ST*) *DBMS\_TYPE*  
 ('*TSQL\_QUERY*, '*PARS*) **STP\_TYPE**  
                   (*'TSQL\_QUERY*, '*PARS*) *STP\_TYPE*  
 '*PARS\_FILTER\_TYPE*  
                   '*PARS\_FILTER\_TYPE*  
 '*ST TF\_TYPE*    '*ST TF\_TYPE*

**21.5 Definitions**

**mkTf<sub>f</sub>**      ⊢ ∀ *dbms stp filter q c st*  
                   • *mkTf<sub>f</sub> dbms stp filter ((q, c), st)*  
                   = (*let res = stp (q, c)*  
                   *in if isError res*  
                   *then (st, c, [], destError res)*  
                   *else*  
                   (*let (dq, ocq, pars) = destVal res*  
                   *in let (st', cks, errs)*  
                   = (*if IsL ocq*

---

```

      then dbms (OutL ocq, st)
      else (st, [], [])
    in if cks = [] ^ errs = []
    then
      let (st'', ans2) = dbms (dq, st')
      in (st'', c, filter (c, ans2, pars))
      else (st', c, [], [notCleared]))
istatef      ⊢ ∀ repr • istatef repr = repr isstate
FE_SWORD    ⊢ ∀ repr dbms stp filter
  • FE_SWORD repr dbms stp filter
    = behaviours
    (mkTff dbms stp filter, istatef repr)
ok_to_proceed
  ⊢ ∀ dbms st stp_res
  • ok_to_proceed dbms stp_res st
    ⇔ ¬ isError stp_res
    ∧ (let (dq, ocq, pars) = destVal stp_res
      in IsL ocq
      ⇒ (let (cks, errs)
          = Snd (dbms (OutL ocq, st))
          in errs = [] ^ cks = []))
subsys_secureA
  ⊢ ∀ repr dbms stp
  • (dbms, stp) ∈ subsys_secureA repr
    ⇔ (∀ q c
  • (let res = stp (q, c)
    in let (dq, ocq, pars) = destVal (stp (q, c))
      in let ok s ⇔ ok_to_proceed dbms res s
        in ∀ s
        • ∃1 s'
          • ok (repr s)
            ⇒ repr s'
            = Fst (dbms (dq, repr s))))
subsys_secureB
  ⊢ ∀ repr dbms stp
  • (dbms, stp) ∈ subsys_secureB repr
    ⇔ (∀ q c
  • (let res = stp (q, c)
    in let (dq, ocq, pars) = destVal (stp (q, c))
      in ∀ st1
        • ¬ isError res ^ IsL ocq
          ⇒ (let (st2, cks, errs)
              = dbms (OutL ocq, st1)
              in st2 = st1 ^ errs = []))
subsys_secureC
  ⊢ ∀ repr dbms stp
  • (dbms, stp) ∈ subsys_secureC repr
    ⇔ (∀ q c

```

---

- (let res = stp (q, c)  
in let (dq, ocq, pars) = destVal (stp (q, c))  
in let ok s  $\Leftrightarrow$  ok\_to\_proceed dbms res s  
in  $\forall$  c' s' s d
  - (repr s', d) = dbms (dq, repr s)  
 $\wedge \neg$  hide (c', s) = hide (c', s')  
 $\wedge \neg$  c' dominates c  
 $\Rightarrow \neg$  ok (repr s)))

**subsys\_secureD**

- $\vdash \forall$  repr dbms stp
- (dbms, stp)  $\in$  subsys\_secureD repr  
 $\Leftrightarrow (\forall$  q c
    - (let res = stp (q, c)  
in let (dq, ocq, pars) = destVal (stp (q, c))  
in let ok s  $\Leftrightarrow$  ok\_to\_proceed dbms res s  
in  $\forall$  c' s<sub>1</sub> s<sub>2</sub> s'<sub>1</sub> s'<sub>2</sub> d<sub>1</sub> d<sub>2</sub>
      - hide (c, s<sub>1</sub>) = hide (c, s<sub>2</sub>)  
 $\wedge$  (repr s'<sub>1</sub>, d<sub>1</sub>)  
= dbms (dq, repr s<sub>1</sub>)  
 $\wedge$  (repr s'<sub>2</sub>, d<sub>2</sub>)  
= dbms (dq, repr s<sub>2</sub>)  
 $\wedge$  c' dominates c  
 $\wedge \neg$  hide (c', s'<sub>1</sub>)  
= hide (c', s'<sub>2</sub>)  
 $\Rightarrow \neg$  ok (repr s<sub>1</sub>)))

**subsys\_secureE**

- $\vdash \forall$  repr dbms stp filter
- (dbms, stp, filter)  $\in$  subsys\_secureE repr  
 $\Leftrightarrow (\forall$  q c
    - (let res = stp (q, c)  
in let (dq, ocq, pars) = destVal (stp (q, c))  
in let ok s  $\Leftrightarrow$  ok\_to\_proceed dbms res s  
in  $\forall$  s<sub>1</sub> s<sub>2</sub> s'<sub>1</sub> s'<sub>2</sub> d<sub>1</sub> d<sub>2</sub>
      - hide (c, s<sub>1</sub>) = hide (c, s<sub>2</sub>)  
 $\wedge$  (repr s'<sub>1</sub>, d<sub>1</sub>)  
= dbms (dq, repr s<sub>1</sub>)  
 $\wedge$  (repr s'<sub>2</sub>, d<sub>2</sub>)  
= dbms (dq, repr s<sub>2</sub>)  
 $\wedge \neg$  filter (c, d<sub>1</sub>, pars)  
= filter (c, d<sub>2</sub>, pars)  
 $\Rightarrow \neg$  ok (repr s<sub>1</sub>)))

**subsys\_secure**

- $\vdash \forall$  repr dbms stp filter
- (dbms, stp, filter)  $\in$  subsys\_secure repr  
 $\Leftrightarrow$  (dbms, stp)  
 $\in$  subsys\_secureA repr  
 $\cap$  subsys\_secureB repr  
 $\cap$  subsys\_secureC repr

$$\begin{aligned} & \cap \text{subsys\_secureD repr} \\ & \wedge (\text{dbms}, \text{stp}, \text{filter}) \in \text{subsys\_secureE repr} \end{aligned}$$

## 22 THE THEORY fef024

### 22.1 Parents

*fef022*

### 22.2 Children

*fef021*

### 22.3 Constants

#### class\_of\_item

*Item*  $\rightarrow$  *Class*

**fill\_cols** *Class*  $\rightarrow$  *Data LIST*  $\times$  *Bool LIST*  $\rightarrow$  (*Item*  $\times$  *Class*) *LIST*

**fill\_row** *Bool*  $\times$  *Class*  $+$  *Maybe*  $\times$  *Bool LIST*  $\times$  *Class*

$\rightarrow$  *Data LIST*

$\rightarrow$  *Class*  $\times$  *Class*  $\times$  (*Item*  $\times$  *Class*) *LIST*

**fill\_table** *Bool*  $\times$  *Class*  $+$  *Maybe*  $\times$  *Bool LIST*  $\times$  *Class*  $\times$  *Select*

$\rightarrow$  (*Class*  $\times$  *Class*  $\times$  (*Item*  $\times$  *Class*) *LIST*) *LIST*

#### filter\_where\_row

*Class*  $\times$  *Class*  $\times$  (*Item*  $\times$  *Class*) *LIST*

$\rightarrow$  (*Item*  $\times$  *Class*) *LIST*  $\times$  *Bool*

**filter\_table** (*Class*  $\times$  *Class*  $\times$  (*Item*  $\times$  *Class*) *LIST*) *LIST*  $\times$  *Class*

$\rightarrow$  (*Item*  $\times$  *Class*) *LIST LIST*  $\times$  *Bool*

**filter\_cols** *Class*  $\rightarrow$  (*Item*  $\times$  *Class*) *LIST*  $\rightarrow$  *Data LIST*

#### filter\_select

(*Class*  $\times$  *Class*  $\times$  (*Item*  $\times$  *Class*) *LIST*) *LIST*  $\times$  *Class*

$\rightarrow$  *Select*  $\times$  *Bool*

**outputFilter** *FILTER\_PARS FILTER\_TYPE*

### 22.4 Type Abbreviations

**FILTER\_PARS** *FILTER\_PARS*

## 22.5 Definitions

### class\_of\_item

$$\vdash \forall i$$

- $class\_of\_item\ i$   
 $= destClassVal\ (VI\_val\ (destValuedItem\ i))$

### fill\_cols

$$\vdash ConstSpec$$

$$(\lambda\ fill\_cols'$$

- $\forall d\ ds\ b\ bs\ clear$
- $fill\_cols'\ clear\ ([], []) = []$
- $\wedge fill\_cols'\ clear\ (Cons\ d\ ds,\ Cons\ b\ bs)$   
 $= (if\ b$   
 $\quad then$   
 $\quad\quad let\ c$   
 $\quad\quad\quad = class\_of\_item$   
 $\quad\quad\quad\quad (Dat\_item\ (Head\ ds))$   
 $\quad\quad\quad in\ Cons$   
 $\quad\quad\quad\quad (Dat\_item\ d,\ c)$   
 $\quad\quad\quad\quad (fill\_cols'\ clear\ (Tail\ ds,\ bs))$   
 $\quad\quad else$   
 $\quad\quad\quad Cons$   
 $\quad\quad\quad\quad (Dat\_item\ d,\ clear)$   
 $\quad\quad\quad\quad (fill\_cols'\ clear\ (ds,\ bs))))$

$$fill\_cols$$

### fill\_row

$$\vdash \forall check\_where\ check\_rc\ check\_cols\ clear\ dl$$

- $fill\_row$   
 $(check\_where,\ check\_rc,\ check\_cols,\ clear)$   
 $dl$   
 $= (let\ (where\_c,\ rest_1)$   
 $\quad = (if\ check\_where$   
 $\quad\quad then$   
 $\quad\quad\quad (class\_of\_item\ (Dat\_item\ (Head\ dl)),$   
 $\quad\quad\quad\quad Tail\ dl)$   
 $\quad\quad else\ (clear,\ dl))$   
 $\quad in\ let\ (rc,\ rest_2)$   
 $\quad = (if\ IsR\ check\_rc$   
 $\quad\quad then$   
 $\quad\quad\quad (class\_of\_item$   
 $\quad\quad\quad\quad (Dat\_item\ (Head\ rest_1)),$   
 $\quad\quad\quad\quad Tail\ rest_1)$   
 $\quad\quad else\ (OutL\ check\_rc,\ rest_1))$   
 $\quad in\ (where\_c,\ rc,$   
 $\quad\quad fill\_cols\ clear\ (rest_2,\ check\_cols)))$

### fill\_table

$$\vdash \forall check\_where\ check\_rc\ check\_cols\ clear\ dll$$

- $fill\_table$   
 $(check\_where,\ check\_rc,\ check\_cols,\ clear,\ dll)$   
 $= Map$   
 $(fill\_row$

```

      (check_where, check_rc, check_cols, clear))
    dll
filter_where_row
  ⊢ ∀ clear where_c icl
    • filter_where_row (clear, where_c, icl)
      = (icl, (¬ clear dominates where_c))
filter_table
  ⊢ ∀ h rest clear
    • filter_table ([], clear) = ([], false)
      ∧ filter_table (Cons h rest, clear)
        = (let (where_c, rc, ics) = h
            in if ¬ clear dominates rc
              then filter_table (rest, clear)
              else
                (let (fics, msg)
                  = filter_where_row
                    (clear, where_c, ics)
                  in let (frest, msgs)
                      = filter_table (rest, clear)
                      in if msg
                        then (frest, true)
                        else (Cons fics frest, msgs)))
filter_cols
  ⊢ ∀ clear ic ics
    • filter_cols clear [] = []
      ∧ filter_cols clear (Cons ic ics)
        = (let (i, c) = ic
            in if clear dominates c
              then
                Cons (MkData c i) (filter_cols clear ics)
              else
                Cons
                  (MkData
                    c
                    (ValuedItemItem
                      (MkValuedItem
                        sterling
                        dummyVal))))
                  (filter_cols clear ics))
filter_select
  ⊢ ∀ table clear
    • filter_select (table, clear)
      = (let (ftable, check)
          = filter_table (table, clear)
          in (Map (filter_cols clear) ftable, check))
outputFilter
  ⊢ ∀ cl dl errs cw cr cc
    • outputFilter (cl, (dl, errs), cw, cr, cc)
      = (if ¬ errs = []
          then ([], errs)
          else

```

---

```
(let (ans, mnc)
  = filter_select
  (fill_table (cw, cr, cc, cl, dl), cl)
  in (ans,
    (if mnc
      then [mayNotBeComplete]
      else []))))
```

## 23 THE THEORY fef025

### 23.1 Parents

*fef021*

### 23.2 Children

*fef028*

### 23.3 Constants

<b>MkData<sub>t</sub></b>	<i>Val</i> → <i>Data</i>
<b>MkNullData<sub>t</sub></b>	<i>Data</i>
<b>MkClassVal</b>	<i>Class</i> → <i>Data</i>
<b>repr_data</b>	<i>Data</i> → <i>Data</i> × <i>Data</i> × <i>Data</i>
<b>Repr_colCon</b>	<i>ColCon</i> → <i>ColCon</i>
<b>ClassName</b>	<i>Op</i> → <i>Op</i>
<b>DinaryName</b>	<i>Op</i> → <i>Op</i>
<b>SterlingName</b>	<i>Op</i> → <i>Op</i>
<b>RowExistName</b>	<i>Op</i> → <i>Op</i>
<b>rowExistCol</b>	<i>Op</i> → <i>ColSpec</i>
<b>rowExistColCon</b>	<i>ColCon</i>
<b>f<sub>1</sub></b>	<i>TableSpec</i> → <i>Worth</i> → <i>ColSpec</i> + <i>Maybe</i>
<b>ColNeeds</b>	<i>TableSpec</i> → <i>Worth</i> → <i>Worth</i> + <i>Maybe</i>
<b>NextNum</b>	<i>TableSpec</i> → <i>Worth</i> → <i>Worth</i>
<b>f<sub>2</sub></b>	<i>TableSpec</i> → <i>ColSpec</i> → <i>ColSpec LIST</i>
<b>f<sub>3</sub></b>	<i>TableSpec</i> → <i>Worth</i> × <i>Data</i> → ( <i>Worth</i> × <i>Data</i> ) <i>LIST</i>
<b>repr_row</b>	<i>TableSpec</i> → <i>Row</i> → <i>Row</i>
<b>repr_table</b>	<i>Op</i> → <i>TableSpec</i> → <i>TableSpec</i>
<b>repr_dir</b>	<i>Directory</i> → <i>Directory</i>
<b>reprState</b>	<i>State</i> → <i>State<sub>t</sub></i>

---

**23.4 Definitions**

**MkData<sub>t</sub>**         $\vdash \forall v \bullet \text{MkData}_t v = \text{newData lattice\_bottom sterling } v$   
**MkNullData<sub>t</sub>**     $\vdash \text{MkNullData}_t$   
                        $= \text{MkData lattice\_bottom (NullItemItem maybe)}$   
**MkClassVal**       $\vdash \forall c \bullet \text{MkClassVal } c = \text{MkData}_t (\text{ClassVal } c)$   
**repr\_data**         $\vdash \forall d$   
                       • *repr\_data d*  
                        $= (\text{let } i = \text{Dat\_item } d$   
                        $\text{in let } c = \text{Dat\_class } d$   
                        $\text{in if isNullItem } i$   
                        $\text{then}$   
                        $(\text{MkNullData}_t, \text{MkNullData}_t, \text{MkClassVal } c)$   
                        $\text{else}$   
                        $(\text{let } v = \text{destValuedItem } i$   
                        $\text{in if VI\_worth } v = \text{sterling}$   
                        $\text{then}$   
                        $(\text{MkData}_t (\text{VI\_val } v), \text{MkNullData}_t,$   
                        $\text{MkClassVal } c)$   
                        $\text{else if VI\_worth } v = \text{dinary}$   
                        $\text{then}$   
                        $(\text{MkNullData}_t, \text{MkData}_t (\text{VI\_val } v),$   
                        $\text{MkClassVal } c)$   
                        $\text{else}$   
                        $(\text{MkNullData}_t, \text{MkNullData}_t,$   
                        $\text{MkClassVal } c)))$   
**Repr\_colCon**     $\vdash \forall cc$   
                       • *Repr\_colCon cc*  
                        $= \text{MkColCon}$   
                        $\text{lattice\_bottom}$   
                        $(\text{CC\_uniform } cc)$   
                        $(\text{CC\_unique } cc)$   
                        $(\text{CC\_classLimited } cc)$   
                        $(\text{CC\_primary } cc)$   
                        $(\text{CC\_secondary } cc)$   
                        $(\text{CC\_referential } cc)$   
**RowExistName**  
**SterlingName**  
**DinaryName**  
**ClassName**       $\vdash \text{ConstSpec}$   
                        $(\lambda$   
                        $(\text{RowExistName}', \text{SterlingName}', \text{DinaryName}',$   
                        $\text{ClassName}')$   
                       • *OneOne RowExistName'*  
                        $\wedge \text{OneOne SterlingName}'$   
                        $\wedge \text{OneOne DinaryName}'$   
                        $\wedge \text{OneOne ClassName}'$   
                        $\wedge (\forall f g x$

---

•  $\{f; g\}$   
 $\subseteq \{RowExistName'; SterlingName';$   
 $DinaryName'; ClassName'\}$   
 $\wedge f x = g x$   
 $\Rightarrow f = g)$   
 (*RowExistName, SterlingName, DinaryName, ClassName*)

**rowExistCol**  $\vdash \forall tab$

- *rowExistCol tab*  
 $= MkColSpec$   
 $(RowExistName tab)$   
 $1$   
*monoleanType*  
*classType*  
*false*  
*MkNullData<sub>t</sub>*  
 $1$   
*lattice\_bottom*  
*lattice\_bottom*

**rowExistColCon**

$\vdash rowExistColCon$   
 $= MkColCon$   
*lattice\_bottom*  
*false*  
*false*  
*false*  
*false*  
*false*  
 $\square$

**f<sub>1</sub>**  $\vdash \forall t n$

- $f_1 t n$   
 $= (if \exists_1 c \bullet c \in TS\_colspecs t \wedge CS\_posn c = n$   
*then*  
 $InL (\epsilon c \bullet c \in TS\_colspecs t \wedge CS\_posn c = n)$   
*else InR maybe)*

**ColNeeds**  $\vdash \forall t n$

- *ColNeeds t n*  
 $= (if IsL (f_1 t n)$   
*then*  
 $let c = OutL (f_1 t n)$   
*in let n'*  
 $= (if CS\_dinaryType c = monoleanType$   
*then 1*  
*else 2)*  
*in let ans*  
 $= (if CS\_min c = CS\_max c$   
*then n'*  
 $else n' + 1) in InL ans$   
*else InR (OutR (f\_1 t n))*

---

---

**NextNum**     $\vdash \forall t n$

- *NextNum t n*
  - = 1
  - + #
    - {i
    - |i < n
    - ∧ *IsL (ColNeeds t i)*
    - ∧ *OutL (ColNeeds t i) = 3*}
  - \* 3
  - + #
    - {i
    - |i < n
    - ∧ *IsL (ColNeeds t i)*
    - ∧ *OutL (ColNeeds t i) = 2*}
  - \* 2
  - + #
    - {i
    - |i < n
    - ∧ *IsL (ColNeeds t i)*
    - ∧ *OutL (ColNeeds t i) = 1*}

**f<sub>2</sub>**     $\vdash \forall t cs$

- *f<sub>2</sub> t cs*
  - = (let *n* = *CS\_posn cs*
  - in let *rc*
  - = (if *TS\_class t* = *TS\_maxRow t*
  - then 0
  - else 1)
  - in let *next* = *rc* + *NextNum t n*
  - in let *cs<sub>s</sub>*
  - = *MkColSpec*
  - (*SterlingName (CS\_ide cs)*)
  - next*
  - monoleanType*
  - (*CS\_sterlingType cs*)
  - (*CS\_nullType cs*)
  - (*Fst (repr\_data (CS\_default cs))*)
  - (*rc* + *CS\_consGroup cs*)
  - lattice\_bottom*
  - lattice\_bottom*
  - in let (*l<sub>1</sub>*, *next<sub>1</sub>*)
  - = (if
  - CS\_dinaryType cs* = *monoleanType*
  - then (*[cs<sub>s</sub>]*, *next* + 1)
  - else
  - (*[cs<sub>s</sub>;*
  - MkColSpec*
  - (*DinaryName*
  - (*CS\_ide cs*))

---

---

```

                                (next + 1)
                                monoleanType
                                (CS_dinaryType cs)
                                (CS_nullType cs)
                                (Fst
                                  (Snd
                                    (repr_data
                                      (CS_default cs))))
                                (rc
                                  + CS_consGroup
                                    cs)
                                lattice_bottom
                                lattice_bottom],
                                next + 2))
  in if CS_min cs = CS_max cs
  then l1
  else
    l1
    @ [MkColSpec
      (ClassName (CS_ide cs))
      next1
      monoleanType
      classType
      (CS_nullType cs)
      (Snd
        (Snd
          (repr_data
            (CS_default cs))))
      (rc + CS_consGroup cs)
      lattice_bottom
      lattice_bottom])
f3 ⊢ ∀ t n d
  • f3 t (n, d)
    = (let rc
      = (if TS_class t = TS_maxRow t
        then 0
        else 1)
      in let next = rc + NextNum t n
        in let cs = OutL (f1 t n)
          in let ds = Fst (repr_data d)
            in let (l1, next1)
              = (if
                CS_dinaryType cs = monoleanType
                then [(next, ds), next + 1)
                else
                  [(next, ds);
                    (next + 1,
                      Fst

```

---

```

                                (Snd
                                  (repr_data
                                    d))), next + 2))
                                in if CS_min cs = CS_max cs
                                  then l1
                                  else
                                    l1
                                    @ [(next1,
                                        Snd (Snd (repr_data d)))]
repr_row ⊢ ∀ t r
  • repr_row t r
    = MkRow
      lattice_bottom
      (let prs = R_data r
        in let prs'
          = {pr'
            | ∃ pr
              • pr ∈ prs
                ∧ pr' ∈ Elems (f3 t pr)}
          in if TS_class t = TS_maxRow t
            then prs'
            else
              prs' ∪ {(1, MkClassVal (R_exist r))})
repr_table ⊢ ∀ i t
  • repr_table i t
    = (let css
      = {cs'
        | ∃ cs
          • cs ∈ TS_colspecs t
            ∧ cs' ∈ Elems (f2 t cs)}
      in let ccs
        = {(n, cc)
          | n ∈ Dom (TS_cons t)
            ∧ cc = Repr_colCon (TS_cons t @ n)}
        in let (css', ccs')
          = (if TS_class t = TS_maxRow t
            then (css, ccs)
            else
              (css ∪ {rowExistCol i},
                {(1, rowExistColCon)}
                ∪ {(n', cc')}
                | ∃ n cc
                  • (n, cc) ∈ ccs
                    ∧ n' = n + 1
                    ∧ cc' = cc}))
          in MkTableSpec
            lattice_bottom
            lattice_bottom

```

---

---

$$\begin{aligned} & \text{repr\_dir} \quad \vdash \forall d \\ & \quad \bullet \text{repr\_dir } d \\ & \quad = \text{MkDirectory} \\ & \quad \quad \{(id, tab) \\ & \quad \quad | id \in \text{Dom } (\text{Dir\_tables } d) \\ & \quad \quad \wedge tab \\ & \quad \quad = \text{repr\_table } id (\text{Dir\_tables } d @ id)\} \\ & \quad \text{lattice\_bottom} \\ & \quad \text{lattice\_bottom} \end{aligned}$$

$$\begin{aligned} \text{reprState} \quad & \vdash \forall s \\ & \quad \bullet \text{reprState } s \\ & \quad = (\text{let } s' = \text{repState } s \\ & \quad \quad \text{in } \text{absState}_t \\ & \quad \quad (\text{absState} \\ & \quad \quad \quad \{(i, dir) \\ & \quad \quad \quad | i \in \text{Dom } s' \\ & \quad \quad \quad \wedge dir = \text{repr\_dir } (s' @ i)\})) \end{aligned}$$

## 24 THE THEORY fef026

### 24.1 Parents

*fef029*

### 24.2 Children

*fef031 fef032*

### 24.3 Constants

**DCS\_max**      *DerColSpec* → *Class*  
**DCS\_min**      *DerColSpec* → *Class*  
**DCS\_name**     *DerColSpec* → *Col LIST*  
**MkDerColSpec** *Col LIST* → *Class* → *Class* → *DerColSpec*  
**DTS\_colSpecs** *DerTableSpec* → *DerColSpec LIST*  
**DTS\_maxRow**   *DerTableSpec* → *Class*  
**DTS\_name**      *DerTableSpec* → *Col LIST*  
**MkDerTableSpec**  
                  *Col LIST* → *Class* → *DerColSpec LIST* → *DerTableSpec*  
**DTR\_cols**      *DerTableRow* → (*Class* × *ValuedItem OPT*) *LIST*  
**DTR\_row**        *DerTableRow* → *Class*  
**DTR\_where**     *DerTableRow* → *Class*  
**MkDerTableRow**  
                  *Class*

---

	→ <i>Class</i>
	→ ( <i>Class</i> × <i>ValuedItem</i> <i>OPT</i> ) <i>LIST</i>
	→ <i>DerTableRow</i>
<b>DT_rows</b>	<i>DerTable</i> → <i>DerTableRow</i> <i>LIST</i>
<b>DT_spec</b>	<i>DerTable</i> → <i>DerTableSpec</i>
<b>MkDerTable</b>	<i>DerTableSpec</i> → <i>DerTableRow</i> <i>LIST</i> → <i>DerTable</i>
<b>EM</b>	( <i>State<sub>t</sub></i> → <i>DerTable</i> <i>LIST</i> )
	→ ( <i>Query</i> → <i>DerTable</i> <i>LIST</i> → <i>DerTable</i> × <i>Errors</i> )
	→ ( <i>Query</i> × ( <i>DerTable</i> × <i>Errors</i> ) × <i>State<sub>t</sub></i>
	→ <i>State<sub>t</sub></i> × <i>ANSWER</i> )
	→ <i>tf<sub>t</sub></i>
<b>StateDirs</b>	<i>State</i> → ( <i>Class</i> × <i>Col</i> × <i>Directory</i> ) <i>LIST</i>
<b>DirTables</b>	<i>Directory</i> → ( <i>Op</i> × <i>TableSpec</i> ) <i>LIST</i>
<b>StateTables</b>	<i>State</i> → ( <i>Class</i> × <i>Col</i> × <i>Op</i> × <i>TableSpec</i> ) <i>LIST</i>
<b>ColSpec<sub>d</sub></b>	<i>Col</i> × <i>Op</i> × <i>TableSpec</i> → <i>ColSpec</i> → <i>DerColSpec</i>
<b>TableSpec<sub>d</sub></b>	<i>Class</i> × <i>Col</i> × <i>Op</i> × <i>TableSpec</i> → <i>DerTableSpec</i>
<b>TableRow<sub>d</sub></b>	<i>TableSpec</i> → <i>Row</i> → <i>DerTableRow</i>
<b>Table<sub>d</sub></b>	<i>Class</i> × <i>Col</i> × <i>Op</i> × <i>TableSpec</i> → <i>DerTable</i>
<b>View<sub>s</sub></b>	<i>State</i> → <i>DerTable</i> <i>LIST</i>
<b>View<sub>t</sub></b>	<i>State<sub>t</sub></i> → <i>DerTable</i> <i>LIST</i>
<b>GiveData</b>	<i>DerTable</i> → <i>Select</i>
<b>is_select</b>	<i>Query</i> → <i>Bool</i>
<b>Act<sub>t</sub></b>	( <i>Query</i> × ( <i>DerTable</i> × <i>Errors</i> ) × <i>State<sub>t</sub></i> → <i>State<sub>t</sub></i> )
	→ <i>Query</i> × ( <i>DerTable</i> × <i>Errors</i> ) × <i>State<sub>t</sub></i>
	→ <i>State<sub>t</sub></i> × <i>ANSWER</i>
<b>HideDerTableRow</b>	<i>Class</i> → <i>DerTableRow</i> → <i>DerTableRow</i>
<b>HideDerTableData</b>	<i>Class</i> → <i>DerTableRow</i> <i>LIST</i> → <i>DerTableRow</i> <i>LIST</i>
<b>HideDerTable</b>	<i>Class</i> → <i>DerTable</i> → <i>DerTable</i>
<b>RiskInputs</b>	<i>Class</i>
	→ ( <i>DerTable</i> <i>LIST</i> → <i>DerTable</i> × <i>Errors</i> )
	→ <i>DerTable</i> <i>LIST</i> $\mathbb{P}$
<b>ConditionE</b>	( <i>Query</i> → <i>DerTable</i> <i>LIST</i> → <i>DerTable</i> × <i>Errors</i> )
	→ <i>Class</i>
	→ ( <i>Query</i> × <i>Query</i> <i>OPT</i> × 'PARS) <i>RESULT</i> $\mathbb{P}$
<b>STP_secure_E</b>	( <i>Query</i> → <i>DerTable</i> <i>LIST</i> → <i>DerTable</i> × <i>Errors</i> )
	→ ( <i>Query</i> , 'PARS) <i>STP_TYPE</i> $\mathbb{P}$
<b>EM<sub>1</sub></b>	( <i>Query</i> → <i>DerTable</i> <i>LIST</i> → <i>DerTable</i> × <i>Errors</i> )
	→ ( <i>Query</i> × ( <i>DerTable</i> × <i>Errors</i> ) × <i>State<sub>t</sub></i>
	→ <i>State<sub>t</sub></i> )
	→ <i>tf<sub>t</sub></i>
<b>Correct_Compile</b>	( <i>Query</i> → <i>DerTable</i> <i>LIST</i> → <i>DerTable</i> × <i>Errors</i> )
	↔ ( <i>Query</i> × ( <i>DerTable</i> × <i>Errors</i> ) × <i>State<sub>t</sub></i>
	→ <i>State<sub>t</sub></i> )

---

## 24.4 Types

**DerColSpec**  
**DerTableSpec**  
**DerTableRow**  
**DerTable**

## 24.5 Definitions

**DerColSpec**  $\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet \text{true}) f$

**MkDerColSpec**

**DCS\_name**

**DCS\_min**

**DCS\_max**  $\vdash \forall t \ x1 \ x2 \ x3$

- $\text{DCS\_name } (\text{MkDerColSpec } x1 \ x2 \ x3) = x1$   
 $\wedge \text{DCS\_min } (\text{MkDerColSpec } x1 \ x2 \ x3) = x2$   
 $\wedge \text{DCS\_max } (\text{MkDerColSpec } x1 \ x2 \ x3) = x3$   
 $\wedge \text{MkDerColSpec}$   
 $\quad (\text{DCS\_name } t)$   
 $\quad (\text{DCS\_min } t)$   
 $\quad (\text{DCS\_max } t)$   
 $= t$

**DerTableSpec**  $\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet \text{true}) f$

**MkDerTableSpec**

**DTS\_name**

**DTS\_maxRow**

**DTS\_colSpecs**  $\vdash \forall t \ x1 \ x2 \ x3$

- $\text{DTS\_name } (\text{MkDerTableSpec } x1 \ x2 \ x3) = x1$   
 $\wedge \text{DTS\_maxRow } (\text{MkDerTableSpec } x1 \ x2 \ x3) = x2$   
 $\wedge \text{DTS\_colSpecs } (\text{MkDerTableSpec } x1 \ x2 \ x3) = x3$   
 $\wedge \text{MkDerTableSpec}$   
 $\quad (\text{DTS\_name } t)$   
 $\quad (\text{DTS\_maxRow } t)$   
 $\quad (\text{DTS\_colSpecs } t)$   
 $= t$

**DerTableRow**  $\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet \text{true}) f$

**MkDerTableRow**

**DTR\_where**

**DTR\_row**

**DTR\_cols**  $\vdash \forall t \ x1 \ x2 \ x3$

- $\text{DTR\_where } (\text{MkDerTableRow } x1 \ x2 \ x3) = x1$   
 $\wedge \text{DTR\_row } (\text{MkDerTableRow } x1 \ x2 \ x3) = x2$   
 $\wedge \text{DTR\_cols } (\text{MkDerTableRow } x1 \ x2 \ x3) = x3$   
 $\wedge \text{MkDerTableRow}$   
 $\quad (\text{DTR\_where } t)$   
 $\quad (\text{DTR\_row } t)$   
 $\quad (\text{DTR\_cols } t)$   
 $= t$

---

<b>DerTable</b>	$\vdash \exists f \bullet \text{TypeDefn } (\lambda x \bullet \text{true}) f$
<b>MkDerTable</b>	
<b>DT_spec</b>	
<b>DT_rows</b>	$\vdash \forall t \ x1 \ x2$ <ul style="list-style-type: none"> <li>• <math>\text{DT\_spec } (\text{MkDerTable } x1 \ x2) = x1</math></li> <li>• <math>\text{DT\_rows } (\text{MkDerTable } x1 \ x2) = x2</math></li> <li>• <math>\text{MkDerTable } (\text{DT\_spec } t) (\text{DT\_rows } t) = t</math></li> </ul>
<b>EM</b>	$\vdash \forall \text{view compile act query st}$ <ul style="list-style-type: none"> <li>• <math>\text{EM view compile act } (\text{query}, \text{st})</math>  <math>= (\text{let compute} = \text{compile query}</math>  <math>\text{in let viewed} = \text{view st}</math>  <math>\text{in let computed} = \text{compute viewed}</math>  <math>\text{in act } (\text{query}, \text{computed}, \text{st}))</math></li> </ul>
<b>StateDirs</b>	$\vdash \text{ConstSpec}$ $(\lambda \text{StateDirs}'$ <ul style="list-style-type: none"> <li>• <math>\forall s</math> <ul style="list-style-type: none"> <li>• <math>\text{Elems } (\text{StateDirs}' s)</math>  <math>= \{(c, i, d)</math>  <math>\mid (i, d) \in \text{repState } s</math>  <math>\wedge c = \text{Dir\_exist } d \text{ lub Dir\_class } d\}</math></li> </ul> </li> </ul> $\text{StateDirs}$
<b>DirTables</b>	$\vdash \text{ConstSpec}$ $(\lambda \text{DirTables}'$ <ul style="list-style-type: none"> <li>• <math>\forall d \bullet \text{Elems } (\text{DirTables}' d) = \text{Dir\_tables } d</math></li> </ul> $\text{DirTables}$
<b>StateTables</b>	$\vdash \forall s$ <ul style="list-style-type: none"> <li>• <math>\text{StateTables } s</math>  <math>= (\text{let } (cl, illdl) = \text{Split } (\text{StateDirs } s)</math>  <math>\text{in let } (ill, dl) = \text{Split } illdl</math>  <math>\text{in let } itll = \text{Map } \text{DirTables } dl</math>  <math>\text{in let } f \ c \ i \ it = (c, i, it)</math>  <math>\text{in let } g \ (c, i, its) = \text{Map } (f \ c \ i) \ its</math>  <math>\text{in let } h \ cl \ is \ itss</math>  <math>= \text{Map}</math>  <math>\quad g</math>  <math>\quad (\text{Combine } cl \ (\text{Combine } is \ itss))</math>  <math>\text{in Flat } (h \ cl \ ill \ itll))</math></li> </ul>
<b>ColSpec<sub>d</sub></b>	$\vdash \forall d\_name \ t\_name \ t \ cs$ <ul style="list-style-type: none"> <li>• <math>\text{ColSpec}_d (d\_name, t\_name, t) \ cs</math>  <math>= (\text{let } cc = \text{TS\_cons } t \ @ \ \text{CS\_consGroup } cs</math>  <math>\text{in let } tc\_name = [t\_name; \text{CS\_ide } cs]</math>  <math>\text{in MkDerColSpec}</math>  <math>\quad [[\text{CS\_ide } cs]; tc\_name; d\_name \ @ \ tc\_name]</math>  <math>\quad (\text{CS\_min } cs)</math>  <math>\quad (\text{CS\_max } cs))</math></li> </ul>
<b>TableSpec<sub>d</sub></b>	$\vdash \forall c \ d\_name \ t\_name \ t$ <ul style="list-style-type: none"> <li>• <math>\text{TableSpec}_d (c, d\_name, t\_name, t)</math>  <math>= \text{MkDerTableSpec}</math></li> </ul>

---

---

```

[[t_name]; d_name @ [t_name]]
(TS_maxRow t)
(RelList
  (Squash
    {(n, cs)
     | ∃ cs'
       • cs' ∈ TS_colspecs t
         ∧ cs
           = ColSpec_d
             (d_name, t_name, t)
             cs'
             ∧ n = CS_posn cs'}}))
TableRow_d ⊢ ∀ t r
  • TableRow_d t r
    = (let f d = (Dat_class d, Dat_item d)
      in MkDerTableRow
        lattice_bottom
        (R_exist r)
        (RelList
          (Squash
            {(n, ic)
             | n ∈ Dom (R_data r)
               ∧ ic = f (R_data r @ n)})))
Table_d ⊢ ∀ c d_name t_name t
  • Table_d (c, d_name, t_name, t)
    = MkDerTable
      (TableSpec_d (c, d_name, t_name, t))
      (Map (TableRow_d t) (TS_rows t))
View_s ⊢ ∀ s • View_s s = Map Table_d (StateTables s)
View_t ⊢ ConstSpec
  (λ View'_t
    • ∀ s_t s_s
      • s_t = reprState s_s
        ⇒ View'_t s_t = View_s s_s)
View_t
GiveData ⊢ ∀ dt
  • GiveData dt
    = (let class_item c
      = ValuedItemItem
        (MkValuedItem sterling (ClassVal c))
      in let item_data i = MkData lattice_bottom i
        in let class_data c
          = item_data (class_item c)
          in let cell_cols (c, i)
            = [class_data c; item_data i]
            in let row_data r
              = Flat
                (Cons

```

---

---

[*class\_data* (*DTR\_where* *r*);  
*class\_data* (*DTR\_row* *r*)]  
(*Map cell\_cols* (*DTR\_cols* *r*)))  
in *Map row\_data* (*DT\_rows* *dt*)

**is\_select**     $\vdash \forall q \bullet \text{is\_select } q \Leftrightarrow (\exists t \bullet q = \text{select } t)$

**Act<sub>t</sub>**         $\vdash \forall \text{upd query } dt \text{ errs } st$   
• *Act<sub>t</sub> upd* (*query*, (*dt*, *errs*), *st*)  
= (*if*  $\neg$  *errs* = []  
then (*st*, [], *errs*)  
else *if is\_select query*  
then (*st*, *GiveData dt*, *errs*)  
else (*upd* (*query*, (*dt*, *errs*), *st*), [], []))

**HideDerTableRow**  
 $\vdash \forall cc \ r$   
• *HideDerTableRow* *cc r*  
= (*let d*  
= *ValuedItemItem*  
(*MkValuedItem sterling dummyVal*)  
in *let hc* (*c*, *i*)  
= (*if cc dominates c*  
then (*c*, *i*)  
else (*c*, *d*))  
in *MkDerTableRow*  
(*DTR\_where* *r*)  
(*DTR\_row* *r*)  
(*Map hc* (*DTR\_cols* *r*)))

**HideDerTableData**  
 $\vdash \forall cc \ rs$   
• *HideDerTableData* *cc rs*  
= (*let okr* = {*r* | *cc dominates DTR\_row r*}  
in *Map* (*HideDerTableRow* *cc*) (*rs* | *okr*))

**HideDerTable**  $\vdash \forall cc \ t$   
• *HideDerTable* *cc t*  
= *MkDerTable*  
(*DT\_spec* *t*)  
(*HideDerTableData* *cc* (*DT\_rows* *t*))

**RiskInputs**     $\vdash \forall c \ f$   
• *RiskInputs* *c f*  
= {*ts*  
|  $\exists ts_0$   
• *Map* (*HideDerTable* *c*) *ts\_0*  
= *Map* (*HideDerTable* *c*) *ts*  
 $\wedge (\neg \text{HideDerTable } c \text{ (Fst } (f \ ts_0))$   
= *HideDerTable* *c* (*Fst* (*f* *ts*))  
 $\vee \neg \text{Snd } (f \ ts_0) = \text{Snd } (f \ ts)$ }}

**ConditionE**     $\vdash \forall \text{compile } cc$   
• *ConditionE* *compile cc*  
= {*stp\_res*

---

---


$$\begin{aligned}
& |isError\ stp\_res \\
& \vee (let\ (dq,\ ocq,\ pars) = destVal\ stp\_res \\
& \quad in\ let\ dcomp = compile\ dq \\
& \quad \quad in\ \forall\ ri \\
& \quad \quad \bullet\ ri \in RiskInputs\ cc\ dcomp \\
& \quad \quad \Rightarrow IsL\ ocq \\
& \quad \quad \quad \wedge\ is\_select\ (OutL\ ocq) \\
& \quad \quad \quad \wedge\ (let\ ccomp = compile\ (OutL\ ocq) \\
& \quad \quad \quad \quad in\ \neg\ DT\_rows\ (Fst\ (ccomp\ ri)) \\
& \quad \quad \quad \quad = [] \\
& \quad \quad \quad \vee\ \neg\ Snd\ (ccomp\ ri) = [])) \\
STP\_secure\_E & \vdash \forall\ compile \\
& \quad \bullet\ STP\_secure\_E\ compile \\
& \quad \quad = \{stp | \forall\ q\ c \bullet stp\ (q,\ c) \in ConditionE\ compile\ c\} \\
EM_1 & \vdash \forall\ compile\ upd \\
& \quad \bullet\ EM_1\ compile\ upd = EM\ View_t\ compile\ (Act_t\ upd) \\
Correct\_Compile & \\
& \vdash Correct\_Compile \\
& \quad = \{(compile,\ upd) | EM_1\ compile\ upd = TSQLEtf\}
\end{aligned}$$

## 25 THE THEORY fef028

### 25.1 Parents

*fef025*

### 25.2 Children

*fef029*

### 25.3 Constants

<b>case</b>	$Value\ LIST \rightarrow Value\ LIST \rightarrow Value \rightarrow Value$
<b>fold</b>	$('a \times 'a \rightarrow 'a) \rightarrow 'a\ LIST \rightarrow 'a$
<b>splice</b>	$'a\ LIST \rightarrow 'b\ LIST \rightarrow ('a \times 'b)\ LIST$
<b>at2</b>	$(('a \times 'b)\ LIST \rightarrow 'c) \rightarrow 'a\ LIST \times 'b\ LIST \rightarrow 'c$
<b>at3</b>	$(('a \times 'b \times 'c)\ LIST \rightarrow 'd)$ $\rightarrow 'a\ LIST \times 'b\ LIST \times 'c\ LIST$ $\rightarrow 'd$
<b>at4</b>	$(('a \times 'b \times 'c \times 'd)\ LIST \rightarrow 'e)$ $\rightarrow 'a\ LIST \times 'b\ LIST \times 'c\ LIST \times 'd\ LIST$ $\rightarrow 'e$
<b>\$dom</b>	$Class \rightarrow Class \rightarrow Bool$
<b>invert</b>	$'a\ LIST\ LIST \rightarrow 'a\ LIST\ LIST$
<b>split3</b>	$('a \times 'b \times 'c)\ LIST \rightarrow 'a\ LIST \times 'b\ LIST \times 'c\ LIST$
<b>split4</b>	$('a \times 'b \times 'c \times 'd)\ LIST$ $\rightarrow 'a\ LIST \times 'b\ LIST \times 'c\ LIST \times 'd\ LIST$

---

**split5**  $( 'a \times 'b \times 'c \times 'd \times 'e ) \text{ LIST}$   
 $\rightarrow 'a \text{ LIST} \times 'b \text{ LIST} \times 'c \text{ LIST} \times 'd \text{ LIST} \times 'e \text{ LIST}$

**CASE**  $'a \rightarrow ( 'a \rightarrow 'b \text{ OPT} ) \text{ LIST} \rightarrow 'b$

**OTHERS**  $( 'a, 'b ) \text{ WHEN\_CONST}$

**Lift**  $( 'rep, 'abs ) \text{ MK\_DEST} \rightarrow ( 'rep, 'abs, 'a ) \text{ WHEN}$

**LiftConstant**  $'abs \rightarrow ( 'abs, 'a ) \text{ WHEN\_CONST}$

**ExceptionValue**  
 $'a \text{ RESULT} \rightarrow \text{Errors}$

**Exception**  $\text{Errors} \rightarrow 'a \text{ RESULT}$

**OkValue**  $'a \text{ RESULT} \rightarrow 'a$

**Ok**  $'a \rightarrow 'a \text{ RESULT}$

**\$WHEN\_exception**  
 $( \text{Errors}, 'a \text{ RESULT}, 'b ) \text{ WHEN}$

**\$WHEN\_ok**  $( 'a, 'a \text{ RESULT}, 'b ) \text{ WHEN}$

**ListOk**  $'a \text{ RESULT LIST} \rightarrow 'a \text{ LIST RESULT}$

**Try**  $( 'a \rightarrow 'b \text{ RESULT} ) \rightarrow 'a \text{ RESULT} \rightarrow 'b \text{ RESULT}$

**dest\_default**  $\text{TableSpecification} \rightarrow \text{Worth} \times \text{Col} \times \text{Op}$

**mk\_default**  $\text{Worth} \times \text{Col} \times \text{Op} \rightarrow \text{TableSpecification}$

**dest\_absolute**  
 $\text{TableSpecification} \rightarrow \text{Col} \times \text{Op}$

**mk\_absolute**  $\text{Col} \times \text{Op} \rightarrow \text{TableSpecification}$

**\$WHEN\_default**  
 $( \text{Worth} \times \text{Col} \times \text{Op}, \text{TableSpecification}, 'b ) \text{ WHEN}$

**\$WHEN\_absolute**  
 $( \text{Col} \times \text{Op}, \text{TableSpecification}, 'b ) \text{ WHEN}$

**c\_anyType**  $\text{SwordType}$

**c\_codeType**  $\text{SwordType}$

**c\_classType**  $\text{SwordType}$

**dest\_intervalType**  
 $\text{SwordType} \rightarrow \text{Op}$

**mk\_intervalType**  
 $\text{Op} \rightarrow \text{SwordType}$

**dest\_timeType**  
 $\text{SwordType} \rightarrow \text{Op}$

**mk\_timeType**  $\text{Op} \rightarrow \text{SwordType}$

**dest\_enumType**  
 $\text{SwordType} \rightarrow \text{Fixed} \times \text{TableSpecification}$

**mk\_enumType**  $\text{Fixed} \times \text{TableSpecification} \rightarrow \text{SwordType}$

**dest\_fixedType**  
 $\text{SwordType} \rightarrow \text{Fixed} \times \text{Fixed}$

**mk\_fixedType**  $\text{Fixed} \times \text{Fixed} \rightarrow \text{SwordType}$

**dest\_stringType**  
 $\text{SwordType} \rightarrow \text{Fixed} \times \text{Fixed}$

**mk\_stringType**  
 $\text{Fixed} \times \text{Fixed} \rightarrow \text{SwordType}$

**c\_booleanType**  
 $\text{SwordType}$

**c\_monoleanType**

---

---

*SwordType*  
**c\_nullType** *SwordType*  
**WHEN\_anyType** (*SwordType*, 'a) *WHEN\_CONST*  
**WHEN\_codeType**  
*(SwordType, 'a) WHEN\_CONST*  
**WHEN\_classType**  
*(SwordType, 'a) WHEN\_CONST*  
**\$WHEN\_intervalType**  
*(Op, SwordType, 'a) WHEN*  
**\$WHEN\_timeType**  
*(Op, SwordType, 'a) WHEN*  
**\$WHEN\_enumType**  
*(Fixed × TableSpecification, SwordType, 'a) WHEN*  
**\$WHEN\_fixedType**  
*(Fixed × Fixed, SwordType, 'a) WHEN*  
**\$WHEN\_stringType**  
*(Fixed × Fixed, SwordType, 'a) WHEN*  
**WHEN\_booleanType**  
*(SwordType, 'a) WHEN\_CONST*  
**WHEN\_monoleanType**  
*(SwordType, 'a) WHEN\_CONST*  
**WHEN\_nullType**  
*(SwordType, 'a) WHEN\_CONST*  
**dest\_name<sub>s</sub>** *SsqlName → Op*  
**mk\_name<sub>s</sub>** *Op → SsqlName*  
**c\_anon<sub>s</sub>** *SsqlName*  
**\$WHEN\_name<sub>s</sub>** (*Op, SsqlName, 'a) WHEN*  
**WHEN\_anon<sub>s</sub>** (*SsqlName, 'a) WHEN\_CONST*  
**dest\_name<sub>t</sub>** *TsqlName → Op*  
**mk\_name<sub>t</sub>** *Op → TsqlName*  
**c\_anon<sub>t</sub>** *TsqlName*  
**c\_none<sub>t</sub>** *TsqlName*  
**\$WHEN\_name<sub>t</sub>** (*Op, TsqlName, 'a) WHEN*  
**WHEN\_anon<sub>t</sub>** (*TsqlName, 'a) WHEN\_CONST*  
**WHEN\_none<sub>t</sub>** (*TsqlName, 'a) WHEN\_CONST*  
**dest\_constant**  
*BoundInfo → Class*  
**mk\_constant** *Class → BoundInfo*  
**dest\_upb** *BoundInfo → Class*  
**mk\_upb** *Class → BoundInfo*  
**\$WHEN\_constant**  
*(Class, BoundInfo, 'a) WHEN*  
**\$WHEN\_upb** (*Class, BoundInfo, 'a) WHEN*  
**sc\_col\_class** *SsqlCol → BoundInfo*  
**sc\_col\_exist** *SsqlCol → Class*  
**sc\_type\_field**  
*SsqlCol → ColType*  
**sc\_name** *SsqlCol → SsqlName*

---

---

**MkSsqlCol**  $SsqlName \rightarrow ColType \rightarrow Class \rightarrow BoundInfo \rightarrow SsqlCol$   
**dest\_constant<sub>tc</sub>**  $TsqlClassName \rightarrow Class$   
**mk\_constant<sub>tc</sub>**  $Class \rightarrow TsqlClassName$   
**dest\_name<sub>tc</sub>**  $TsqlClassName \rightarrow Op$   
**mk\_name<sub>tc</sub>**  $Op \rightarrow TsqlClassName$   
**c\_anon<sub>tc</sub>**  $TsqlClassName$   
**\$WHEN\_constant<sub>tc</sub>**  $(Class, TsqlClassName, 'a) WHEN$   
**\$WHEN\_name<sub>tc</sub>**  $(Op, TsqlClassName, 'a) WHEN$   
**WHEN\_anon<sub>tc</sub>**  $(TsqlClassName, 'a) WHEN_CONST$   
**tc\_class\_name**  $TsqlCol \rightarrow TsqlClassName$   
**tc\_dinary\_name**  $TsqlCol \rightarrow TsqlName$   
**tc\_sterling\_name**  $TsqlCol \rightarrow TsqlName$   
**MkTsqlCol**  $TsqlName \rightarrow TsqlName \rightarrow TsqlClassName \rightarrow TsqlCol$   
**ti\_row\_class**  $TableInfo \rightarrow BoundInfo$   
**ti\_table\_class**  $TableInfo \rightarrow Class$   
**ti\_table\_exist\_class**  $TableInfo \rightarrow Class$   
**MkTableInfo**  $Class \rightarrow Class \rightarrow BoundInfo \rightarrow TableInfo$   
**ci\_index**  $ConstraintInfo \rightarrow Errors LIST$   
**ci\_uniform**  $ConstraintInfo \rightarrow Errors LIST$   
**ci\_unique**  $ConstraintInfo \rightarrow Errors LIST$   
**ci\_lwb**  $ConstraintInfo \rightarrow Class LIST$   
**ci\_null\_allowed**  $ConstraintInfo \rightarrow Bool LIST$   
**MkConstraintInfo**  $Bool LIST$   
 $\rightarrow Class LIST$   
 $\rightarrow Errors LIST$   
 $\rightarrow Errors LIST$   
 $\rightarrow Errors LIST$   
 $\rightarrow ConstraintInfo$   
**dest\_specific**  $ColumnSpecification \rightarrow TableSpecification \times Op$   
**mk\_specific**  $TableSpecification \times Op \rightarrow ColumnSpecification$   
**dest\_anonymous\_column**  $ColumnSpecification \rightarrow Op$   
**mk\_anonymous\_column**  $Op \rightarrow ColumnSpecification$

---

---

**\$WHEN\_specific**  $(TableSpecification \times Op, ColumnSpecification, 'a) WHEN$   
**\$WHEN\_anonymous\_column**  $(Op, ColumnSpecification, 'a) WHEN$   
**c\_constant\_null**  $TsqlRepr$   
**dest\_constant\_class**  $TsqlRepr \rightarrow Class$   
**mk\_constant\_class**  $Class \rightarrow TsqlRepr$   
**dest\_column**  $TsqlRepr \rightarrow Op \times Op$   
**mk\_column**  $Op \times Op \rightarrow TsqlRepr$   
**dest\_local\_identifier**  $TsqlRepr \rightarrow Op$   
**mk\_local\_identifier**  $Op \rightarrow TsqlRepr$   
**WHEN\_constant\_null**  $(TsqlRepr, 'a) WHEN\_CONST$   
**\$WHEN\_constant\_class**  $(Class, TsqlRepr, 'a) WHEN$   
**\$WHEN\_column**  $(Op \times Op, TsqlRepr, 'a) WHEN$   
**\$WHEN\_local\_identifier**  $(Op, TsqlRepr, 'a) WHEN$   
**dest\_constant<sub>ec</sub>**  $ExpClass \rightarrow Class$   
**mk\_constant<sub>ec</sub>**  $Class \rightarrow ExpClass$   
**dest\_variable**  $ExpClass \rightarrow Value \times Class$   
**mk\_variable**  $Value \times Class \rightarrow ExpClass$   
**\$WHEN\_constant<sub>ec</sub>**  $(Class, ExpClass, 'a) WHEN$   
**\$WHEN\_variable**  $(Value \times Class, ExpClass, 'a) WHEN$   
**dest\_simple**  $InternalExpClass \rightarrow ExpClass$   
**mk\_simple**  $ExpClass \rightarrow InternalExpClass$   
**dest\_ors**  $InternalExpClass \rightarrow Value LIST \times ExpClass LIST$   
**mk\_ors**  $Value LIST \times ExpClass LIST \rightarrow InternalExpClass$   
**dest\_and**  $InternalExpClass \rightarrow Value LIST \times ExpClass LIST$   
**mk\_and**  $Value LIST \times ExpClass LIST \rightarrow InternalExpClass$   
**\$WHEN\_simple**  $(ExpClass, InternalExpClass, 'a) WHEN$   
**\$WHEN\_ors**  $(Value LIST \times ExpClass LIST, InternalExpClass, 'a) WHEN$   
**\$WHEN\_and**  $(Value LIST \times ExpClass LIST, InternalExpClass, 'a) WHEN$   
**dest\_name<sub>tn</sub>**  $TableName \rightarrow TableSpecification$   
**mk\_name<sub>tn</sub>**  $TableSpecification \rightarrow TableName$   
**c\_anon<sub>tn</sub>**  $TableName$   
**\$WHEN\_name<sub>tn</sub>**

---

---

*(TableSpecification, TableName, 'a) WHEN*  
**WHEN\_anon<sub>tn</sub>** *(TableName, 'a) WHEN\_CONST*  
**td\_constraints** *TableDetail → ConstraintInfo*  
**td\_implementation** *TableDetail → TsqlCol LIST*  
**td\_rowClass** *TableDetail → TsqlClassName*  
**td\_columns** *TableDetail → SsqlCol LIST*  
**td\_info** *TableDetail → TableInfo*  
**td\_genCorr** *TableDetail → Op*  
**td\_corrName** *TableDetail → SsqlName*  
**td\_tableName** *TableDetail → TableName*  
**MkTableDetail** *TableName*  
*→ SsqlName*  
*→ Op*  
*→ TableInfo*  
*→ SsqlCol LIST*  
*→ TsqlClassName*  
*→ TsqlCol LIST*  
*→ ConstraintInfo*  
*→ TableDetail*  
**id\_cName** *IdentDetail → TsqlName*  
**id\_vName** *IdentDetail → Op*  
**id\_lub<sub>id</sub>** *IdentDetail → Class*  
**id\_info** *IdentDetail → ExpType*  
**id\_identName** *IdentDetail → Op*  
**MkIdentDetail** *Op → ExpType → Class → Op → TsqlName → IdentDetail*  
**s\_identifiers** *Scope → IdentDetail LIST*  
**s\_tables** *Scope → TableDetail LIST*  
**MkScope** *TableDetail LIST → IdentDetail LIST → Scope*  
**pi\_clasf** *ParamInfo → Class*  
**pi\_valp** *ParamInfo → Value*  
**pi\_name** *ParamInfo → Op*  
**MkParamInfo** *Op → Value → Class → ParamInfo*  
**parameterTable** *ST\_STACK → ParamInfo LIST*  
**symbolTable** *ST\_STACK → Scope LIST*  
**MkST\_STACK** *Scope LIST → ParamInfo LIST → ST\_STACK*

## 25.4 Types

**SsqlCol**  
**TsqlCol**  
**TableInfo**

**ConstraintInfo**  
**TableDetail**  
**IdentDetail**  
**Scope**  
**ParamInfo**  
**ST\_STACK**

## 25.5 Type Abbreviations

**Enum**            *Fixed*  
**Integer**        *Fixed*  
**Fixed**            *Fixed*  
**'a OPT**         *'a OPT*  
**('rep, 'abs) MK\_DEST**  
                      *('rep, 'abs) MK\_DEST*  
**('rep, 'abs, 'a) WHEN**  
                      *('rep, 'abs, 'a) WHEN*  
**('abs, 'a) WHEN\_CONST**  
                      *('abs, 'a) WHEN\_CONST*  
**'a RESULT**      *'a RESULT*

### TableSpecification

*TableSpecification*  
**SwordType**      *SwordType*  
**SsqlName**        *SsqlName*  
**TsqlName**        *TsqlName*  
**ColType**         *ColType*  
**BoundInfo**      *BoundInfo*  
**TsqlClassName**  
                      *TsqlClassName*

### ColumnSpecification

*ColumnSpecification*  
**TsqlRepr**        *TsqlRepr*  
**ExpType**         *ExpType*  
**ExpClass**        *ExpClass*

### InternalExpClass

*InternalExpClass*  
**TableName**      *TableName*

## 25.6 Fixity

*Binder:*            **WHEN\_absolute**    **WHEN\_local\_identifier**  
                      **WHEN\_ands** **WHEN\_name<sub>s</sub>**  
                      **WHEN\_anonymous\_column**    **WHEN\_name<sub>t</sub>**  
                      **WHEN\_column**      **WHEN\_name<sub>tc</sub>**  
                      **WHEN\_constant**    **WHEN\_name<sub>tn</sub>**  
                      **WHEN\_constant\_class**    **WHEN\_ok**  
                      **WHEN\_constant<sub>ec</sub>** **WHEN\_ors**  
                      **WHEN\_constant<sub>tc</sub>** **WHEN\_simple**

**WHEN\_default**    **WHEN\_specific**  
**WHEN\_enumType** **WHEN\_stringType**  
**WHEN\_exception** **WHEN\_timeType**  
**WHEN\_fixedType** **WHEN\_upb**  
**WHEN\_intervalType**    **WHEN\_variable**

*Right Infix 150:*

**dom**

## 25.7 Definitions

**case**             $\vdash true$   
**fold**             $\vdash \forall f h t$   
                    •  $fold\ f\ (Cons\ h\ t)$   
                    =  $(if\ t = []\ then\ h\ else\ f\ (h,\ fold\ f\ t))$   
**splice**           $\vdash \forall h1\ h2\ list1\ list2$   
                    •  $splice\ []\ [] = []$   
                     $\wedge\ splice\ []\ (Cons\ h2\ list2) = []$   
                     $\wedge\ splice\ (Cons\ h1\ list1)\ [] = []$   
                     $\wedge\ splice\ (Cons\ h1\ list1)\ (Cons\ h2\ list2)$   
                    =  $Cons\ (h1,\ h2)\ (splice\ list1\ list2)$   
**at2**             $\vdash \forall f\ as\ bs \bullet at2\ f\ (as,\ bs) = f\ (Combine\ as\ bs)$   
**at3**             $\vdash \forall f\ as\ bs\ cs$   
                    •  $at3\ f\ (as,\ bs,\ cs) = f\ (Combine\ as\ (Combine\ bs\ cs))$   
**at4**             $\vdash \forall f\ as\ bs\ cs\ ds$   
                    •  $at4\ f\ (as,\ bs,\ cs,\ ds)$   
                    =  $f\ (Combine\ as\ (Combine\ bs\ (Combine\ cs\ ds)))$   
**dom**             $\vdash \$dom = \$dominates$   
**invert**           $\vdash \forall xs\ more$   
                    •  $invert\ (Cons\ xs\ more)$   
                    =  $(if\ more = []$   
                     $\quad then\ Map\ (\lambda x \bullet [x])\ xs$   
                     $\quad else$   
                     $\quad Map$   
                     $\quad (\lambda (x,\ ys) \bullet Cons\ x\ ys)$   
                     $\quad (Combine\ xs\ (invert\ more)))$   
**split3**           $\vdash \forall abc$   
                    •  $split3\ abc$   
                    =  $(let\ (as,\ bcs) = Split\ abc\ in\ (as,\ Split\ bcs))$   
**split4**           $\vdash \forall abcd$   
                    •  $split4\ abcd$   
                    =  $(let\ (as,\ bcde) = Split\ abcd$   
                     $\quad in\ (as,\ split3\ bcde))$   
**split5**           $\vdash \forall abcde$   
                    •  $split5\ abcde$   
                    =  $(let\ (as,\ bcde) = Split\ abcde$   
                     $\quad in\ (as,\ split4\ bcde))$   
**CASE**           $\vdash \forall a\ f\ fs$   
                    •  $CASE\ a\ (Cons\ f\ fs)$



**c\_nullType**  
**c\_monoleanType**  
**c\_booleanType**  
**dest\_stringType**  
**mk\_stringType**  
**dest\_fixedType**  
**mk\_fixedType**  
**dest\_enumType**  
**mk\_enumType**  
**dest\_timeType**  
**mk\_timeType**  
**dest\_intervalType**  
**mk\_intervalType**  
**c\_classType**  
**c\_codeType**  
**c\_anyType**

$$\begin{aligned}
&\vdash c\_nullType = InL \text{ maybe} \\
&\quad \wedge c\_monoleanType = (InR \circ InL) \text{ maybe} \\
&\quad \wedge c\_booleanType = (InR \circ InR \circ InL) \text{ maybe} \\
&\quad \wedge mk\_stringType = InR \circ InR \circ InR \circ InL \\
&\quad \wedge mk\_fixedType = InR \circ InR \circ InR \circ InR \circ InL \\
&\quad \wedge mk\_enumType = InR \circ InR \circ InR \circ InR \circ InR \circ InL \\
&\quad \wedge mk\_timeType \\
&\quad \quad = InR \circ InR \circ InR \circ InR \circ InR \circ InR \circ InL \\
&\quad \wedge mk\_intervalType \\
&\quad \quad = InR \circ InR \circ InR \circ InR \circ InR \circ InR \circ InR \circ InL \\
&\quad \wedge c\_classType \\
&\quad \quad = (InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InL) \\
&\quad \quad \text{maybe} \\
&\quad \wedge c\_codeType \\
&\quad \quad = (InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InR \\
&\quad \quad \quad \circ InL) \\
&\quad \quad \text{maybe}
\end{aligned}$$



---


$$\begin{aligned} & \wedge \$WHEN\_intervalType \\ & = Lift (mk\_intervalType, dest\_intervalType) \\ & \wedge WHEN\_classType = LiftConstant c\_classType \\ & \wedge WHEN\_codeType = LiftConstant c\_codeType \\ & \wedge WHEN\_anyType = LiftConstant c\_anyType \\ \mathbf{c\_anon_s} \\ \mathbf{dest\_name_s} \\ \mathbf{mk\_name_s} & \vdash c\_anon_s = InL \text{ maybe} \\ & \wedge mk\_name_s = InR \\ & \wedge dest\_name_s = OutR \\ \mathbf{WHEN\_anon_s} \\ \mathbf{WHEN\_name_s} & \vdash WHEN\_anon_s = LiftConstant c\_anon_s \\ & \wedge \$WHEN\_name_s = Lift (mk\_name_s, dest\_name_s) \\ \mathbf{c\_none_t} \\ \mathbf{c\_anon_t} \\ \mathbf{dest\_name_t} \\ \mathbf{mk\_name_t} & \vdash c\_none_t = InL \text{ maybe} \\ & \wedge c\_anon_t = (InR \circ InL) \text{ maybe} \\ & \wedge mk\_name_t = InR \circ InR \\ & \wedge dest\_name_t = OutR \circ OutR \\ \mathbf{WHEN\_none_t} \\ \mathbf{WHEN\_anon_t} \\ \mathbf{WHEN\_name_t} & \vdash WHEN\_none_t = LiftConstant c\_none_t \\ & \wedge WHEN\_anon_t = LiftConstant c\_anon_t \\ & \wedge \$WHEN\_name_t = Lift (mk\_name_t, dest\_name_t) \\ \mathbf{dest\_upb} \\ \mathbf{mk\_upb} \\ \mathbf{dest\_constant} \\ \mathbf{mk\_constant} & \vdash mk\_upb = InL \\ & \wedge mk\_constant = InR \\ & \wedge dest\_upb = OutL \\ & \wedge dest\_constant = OutR \\ \mathbf{WHEN\_upb} \\ \mathbf{WHEN\_constant} & \vdash \$WHEN\_upb = Lift (mk\_upb, dest\_upb) \\ & \wedge \$WHEN\_constant \\ & = Lift (mk\_constant, dest\_constant) \\ \mathbf{SsqlCol} \\ \mathbf{MkSsqlCol} \\ \mathbf{sc\_name} \\ \mathbf{sc\_type\_field} \\ \mathbf{sc\_col\_exist} \\ \mathbf{sc\_col\_class} & \vdash \exists f \bullet TypeDefn (\lambda x \bullet true) f \\ & \bullet sc\_name (MkSsqlCol x1 x2 x3 x4) = x1 \\ & \wedge sc\_type\_field (MkSsqlCol x1 x2 x3 x4) = x2 \\ & \wedge sc\_col\_exist (MkSsqlCol x1 x2 x3 x4) = x3 \\ & \wedge sc\_col\_class (MkSsqlCol x1 x2 x3 x4) = x4 \\ & \wedge MkSsqlCol \end{aligned}$$


---

---


$$\begin{aligned}
& (sc\_name\ t) \\
& (sc\_type\_field\ t) \\
& (sc\_col\_exist\ t) \\
& (sc\_col\_class\ t) \\
& = t
\end{aligned}$$

**c\_anon<sub>tc</sub>**  
**dest\_name<sub>tc</sub>**  
**mk\_name<sub>tc</sub>**  
**dest\_constant<sub>tc</sub>**  
**mk\_constant<sub>tc</sub>**

$$\begin{aligned}
& \vdash c\_anon_{tc} = InL\ maybe \\
& \quad \wedge mk\_name_{tc} = InR\ o\ InL \\
& \quad \wedge mk\_constant_{tc} = InR\ o\ InR \\
& \quad \wedge dest\_name_{tc} = OutL\ o\ OutR \\
& \quad \wedge dest\_constant_{tc} = OutR\ o\ OutR
\end{aligned}$$

**WHEN\_anon<sub>tc</sub>**  
**WHEN\_name<sub>tc</sub>**  
**WHEN\_constant<sub>tc</sub>**

$$\begin{aligned}
& \vdash WHEN\_anon_{tc} = LiftConstant\ c\_anon_{tc} \\
& \quad \wedge \$WHEN\_name_{tc} \\
& \quad = Lift\ (mk\_name_{tc},\ dest\_name_{tc}) \\
& \quad \wedge \$WHEN\_constant_{tc} \\
& \quad = Lift\ (mk\_constant_{tc},\ dest\_constant_{tc})
\end{aligned}$$

**TsqlCol**  
**MkTsqlCol**  
**tc\_sterling\_name**  
**tc\_dinary\_name**  
**tc\_class\_name**

$$\begin{aligned}
& \vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f \\
& \vdash \forall t\ x1\ x2\ x3 \\
& \quad \bullet tc\_sterling\_name\ (MkTsqlCol\ x1\ x2\ x3) = x1 \\
& \quad \quad \wedge tc\_dinary\_name\ (MkTsqlCol\ x1\ x2\ x3) = x2 \\
& \quad \quad \wedge tc\_class\_name\ (MkTsqlCol\ x1\ x2\ x3) = x3 \\
& \quad \quad \wedge MkTsqlCol \\
& \quad \quad \quad (tc\_sterling\_name\ t) \\
& \quad \quad \quad (tc\_dinary\_name\ t) \\
& \quad \quad \quad (tc\_class\_name\ t) \\
& \quad = t
\end{aligned}$$

**TableInfo**  
**MkTableInfo**  
**ti\_table\_exist\_class**  
**ti\_table\_class**  
**ti\_row\_class**

$$\begin{aligned}
& \vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f \\
& \vdash \forall t\ x1\ x2\ x3 \\
& \quad \bullet ti\_table\_exist\_class\ (MkTableInfo\ x1\ x2\ x3) = x1 \\
& \quad \quad \wedge ti\_table\_class\ (MkTableInfo\ x1\ x2\ x3) = x2 \\
& \quad \quad \wedge ti\_row\_class\ (MkTableInfo\ x1\ x2\ x3) = x3 \\
& \quad \quad \wedge MkTableInfo \\
& \quad \quad \quad (ti\_table\_exist\_class\ t) \\
& \quad \quad \quad (ti\_table\_class\ t)
\end{aligned}$$


---

$$(ti\_row\_class\ t)$$

$$= t$$

**ConstraintInfo**

$$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f$$

**MkConstraintInfo****ci\_null\_allowed****ci\_lwb****ci\_unique****ci\_uniform****ci\_index**

$$\vdash \forall t\ x1\ x2\ x3\ x4\ x5$$

$$\bullet ci\_null\_allowed\ (MkConstraintInfo\ x1\ x2\ x3\ x4\ x5)$$

$$= x1$$

$$\wedge ci\_lwb\ (MkConstraintInfo\ x1\ x2\ x3\ x4\ x5) = x2$$

$$\wedge ci\_unique\ (MkConstraintInfo\ x1\ x2\ x3\ x4\ x5)$$

$$= x3$$

$$\wedge ci\_uniform\ (MkConstraintInfo\ x1\ x2\ x3\ x4\ x5)$$

$$= x4$$

$$\wedge ci\_index\ (MkConstraintInfo\ x1\ x2\ x3\ x4\ x5) = x5$$

$$\wedge MkConstraintInfo$$

$$(ci\_null\_allowed\ t)$$

$$(ci\_lwb\ t)$$

$$(ci\_unique\ t)$$

$$(ci\_uniform\ t)$$

$$(ci\_index\ t)$$

$$= t$$

**dest\_anonymous\_column****mk\_anonymous\_column****dest\_specific**

$$mk\_specific \vdash mk\_anonymous\_column = InL$$

$$\wedge mk\_specific = InR$$

$$\wedge dest\_anonymous\_column = OutL$$

$$\wedge dest\_specific = OutR$$

**WHEN\_anonymous\_column****WHEN\_specific**

$$\vdash \$WHEN\_anonymous\_column$$

$$= Lift$$

$$(mk\_anonymous\_column, dest\_anonymous\_column)$$

$$\wedge \$WHEN\_specific$$

$$= Lift\ (mk\_specific, dest\_specific)$$

**dest\_local\_identifier****mk\_local\_identifier****dest\_column****mk\_column****dest\_constant\_class****mk\_constant\_class****c\_constant\_null**

$$\vdash mk\_local\_identifier = InL$$

$$\wedge mk\_column = InR\ o\ InL$$

---


$$\begin{aligned} & \wedge mk\_constant\_class = InR \ o \ InR \ o \ InL \\ & \wedge c\_constant\_null = (InR \ o \ InR \ o \ InR) \ maybe \\ & \wedge dest\_local\_identifier = OutL \\ & \wedge dest\_column = OutL \ o \ OutR \\ & \wedge dest\_constant\_class = OutL \ o \ OutR \ o \ OutR \end{aligned}$$

**WHEN\_local\_identifier**  
**WHEN\_column**  
**WHEN\_constant\_class**  
**WHEN\_constant\_null**

$$\begin{aligned} & \vdash \$WHEN\_local\_identifier \\ & \quad = Lift \\ & \quad \quad (mk\_local\_identifier, dest\_local\_identifier) \\ & \wedge \$WHEN\_column = Lift (mk\_column, dest\_column) \\ & \wedge \$WHEN\_constant\_class \\ & \quad = Lift (mk\_constant\_class, dest\_constant\_class) \\ & \wedge WHEN\_constant\_null = LiftConstant c\_constant\_null \end{aligned}$$

**dest\_variable**  
**mk\_variable**  
**dest\_constant<sub>ec</sub>**  
**mk\_constant<sub>ec</sub>**

$$\begin{aligned} & \vdash mk\_variable = InL \\ & \quad \wedge mk\_constant_{ec} = InR \\ & \quad \wedge dest\_variable = OutL \\ & \quad \wedge dest\_constant_{ec} = OutR \end{aligned}$$

**WHEN\_variable**  
**WHEN\_constant<sub>ec</sub>**

$$\begin{aligned} & \vdash \$WHEN\_variable = Lift (mk\_variable, dest\_variable) \\ & \quad \wedge \$WHEN\_constant_{ec} \\ & \quad \quad = Lift (mk\_constant_{ec}, dest\_constant_{ec}) \end{aligned}$$

**dest\_and**  
**mk\_and**  
**dest\_or**  
**mk\_or**  
**dest\_simple**  
**mk\_simple**

$$\begin{aligned} & \vdash mk\_ands = InL \\ & \quad \wedge mk\_ors = InR \ o \ InL \\ & \quad \wedge mk\_simple = InR \ o \ InR \\ & \quad \wedge dest\_ands = OutL \\ & \quad \wedge dest\_ors = OutL \ o \ OutR \\ & \quad \wedge dest\_simple = OutR \ o \ OutR \end{aligned}$$

**WHEN\_and**  
**WHEN\_or**  
**WHEN\_simple**

$$\begin{aligned} & \vdash \$WHEN\_ands = Lift (mk\_ands, dest\_ands) \\ & \quad \wedge \$WHEN\_ors = Lift (mk\_ors, dest\_ors) \\ & \quad \wedge \$WHEN\_simple = Lift (mk\_simple, dest\_simple) \end{aligned}$$

**c\_anon<sub>tn</sub>**  
**dest\_name<sub>tn</sub>**  
**mk\_name<sub>tn</sub>**

$$\vdash c\_anon_{tn} = InL \ maybe$$


---

---


$$\wedge mk\_name_{tn} = InR$$

$$\wedge dest\_name_{tn} = OutR$$

**WHEN\_**anon<sub>tn</sub>

**WHEN\_**name<sub>tn</sub>

$$\vdash WHEN\_anon_{tn} = LiftConstant\ c\_anon_{tn}$$

$$\wedge \$WHEN\_name_{tn}$$

$$= Lift\ (mk\_name_{tn},\ dest\_name_{tn})$$

**TableDetail**

$$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f$$

**MkTableDetail**

**td\_**tableName

**td\_**corrName

**td\_**genCorr

**td\_**info

**td\_**columns

**td\_**rowClass

**td\_**implementation

**td\_**constraints

$$\vdash \forall t\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8$$

- **td\_**tableName
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x1$$
- $\wedge$  **td\_**corrName
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x2$$
- $\wedge$  **td\_**genCorr
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x3$$
- $\wedge$  **td\_**info
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x4$$
- $\wedge$  **td\_**columns
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x5$$
- $\wedge$  **td\_**rowClass
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x6$$
- $\wedge$  **td\_**implementation
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x7$$
- $\wedge$  **td\_**constraints
 
$$(MkTableDetail\ x1\ x2\ x3\ x4\ x5\ x6\ x7\ x8)$$

$$= x8$$
- $\wedge$  **MkTableDetail**

$$(td\_tableName\ t)$$

$$(td\_corrName\ t)$$

$$(td\_genCorr\ t)$$

$$(td\_info\ t)$$

$$(td\_columns\ t)$$

$$(td\_rowClass\ t)$$

---

---


$$\begin{aligned}
& (td\_implementation\ t) \\
& (td\_constraints\ t) \\
& = t \\
\mathbf{IdentDetail} & \vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f \\
\mathbf{MkIdentDetail} & \\
\mathbf{id\_identName} & \\
\mathbf{id\_info} & \\
\mathbf{id\_lub_{id}} & \\
\mathbf{id\_vName} & \\
\mathbf{id\_cName} & \vdash \forall t\ x1\ x2\ x3\ x4\ x5 \\
& \bullet id\_identName\ (MkIdentDetail\ x1\ x2\ x3\ x4\ x5) = x1 \\
& \wedge id\_info\ (MkIdentDetail\ x1\ x2\ x3\ x4\ x5) = x2 \\
& \wedge id\_lub_{id}\ (MkIdentDetail\ x1\ x2\ x3\ x4\ x5) = x3 \\
& \wedge id\_vName\ (MkIdentDetail\ x1\ x2\ x3\ x4\ x5) = x4 \\
& \wedge id\_cName\ (MkIdentDetail\ x1\ x2\ x3\ x4\ x5) = x5 \\
& \wedge MkIdentDetail \\
& \quad (id\_identName\ t) \\
& \quad (id\_info\ t) \\
& \quad (id\_lub_{id}\ t) \\
& \quad (id\_vName\ t) \\
& \quad (id\_cName\ t) \\
& = t \\
\mathbf{Scope} & \vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f \\
\mathbf{MkScope} & \\
\mathbf{s\_tables} & \\
\mathbf{s\_identifiers} & \vdash \forall t\ x1\ x2 \\
& \bullet s\_tables\ (MkScope\ x1\ x2) = x1 \\
& \wedge s\_identifiers\ (MkScope\ x1\ x2) = x2 \\
& \wedge MkScope\ (s\_tables\ t)\ (s\_identifiers\ t) = t \\
\mathbf{ParamInfo} & \vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f \\
\mathbf{MkParamInfo} & \\
\mathbf{pi\_name} & \\
\mathbf{pi\_val_p} & \\
\mathbf{pi\_clasf} & \vdash \forall t\ x1\ x2\ x3 \\
& \bullet pi\_name\ (MkParamInfo\ x1\ x2\ x3) = x1 \\
& \wedge pi\_val_p\ (MkParamInfo\ x1\ x2\ x3) = x2 \\
& \wedge pi\_clasf\ (MkParamInfo\ x1\ x2\ x3) = x3 \\
& \wedge MkParamInfo \\
& \quad (pi\_name\ t) \\
& \quad (pi\_val_p\ t) \\
& \quad (pi\_clasf\ t) \\
& = t \\
\mathbf{ST\_STACK} & \vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f \\
\mathbf{MkST\_STACK} & \\
\mathbf{symbolTable} & \\
\mathbf{parameterTable} & \vdash \forall t\ x1\ x2
\end{aligned}$$


---

- $\text{symbolTable } (\text{MkST\_STACK } x1 \ x2) = x1$   
 $\wedge \text{parameterTable } (\text{MkST\_STACK } x1 \ x2) = x2$   
 $\wedge \text{MkST\_STACK } (\text{symbolTable } t) (\text{parameterTable } t)$   
 $= t$

## 26 THE THEORY fef029

### 26.1 Parents

*fef028*

### 26.2 Children

*fef026*

### 26.3 Constants

**emptyUnionList**

*Worth*

**ambiguousName**

*Worth*

**wrongScope**

*Worth*

**noScope**

*Worth*

**noSuchParameter**

*Worth*

**notSetFunction**

*Worth*

**notTriadic**

*Worth*

**notDyadic**

*Worth*

**notMonadic**

*Worth*

**onlyInTriggers**

*Worth*

**notTrigger**

*Worth*

**wrongWorth**

*Worth*

**internalError**

*Worth*

**client\_clearance**

*TRANS\_STATE* → *Class*

**query\_constants\_class**

*TRANS\_STATE* → *Class*

**query\_class**

*TRANS\_STATE* → *Class*

**st\_stack**

*TRANS\_STATE* → *ST\_STACK*

**MkTRANS\_STATE**

*ST\_STACK* → *Class* → *Class* → *Class* → *TRANS\_STATE*

**default\_directory**

*Col*

**contextual\_data**

---

$Op \rightarrow Value \times Class$   
**unique\_name**  $TRANS\_STATE \rightarrow Op$   
**timeFormatToInterval**  
 $Op \rightarrow Op$   
**check\_time**  $Time \times Op \rightarrow Bool$   
**check\_interval**  
 $Interval \times Op \rightarrow Bool$   
**check\_floating**  
 $Float \times Fixed \times Fixed \times Fixed \rightarrow Bool$   
**check\_fixed**  $Fixed \times Fixed \times Fixed \rightarrow Bool$   
**check\_enum**  $Fixed \times Fixed \times Table\_spec \rightarrow Bool$   
**init\_trans\_state**  
 $Class \rightarrow TRANS\_STATE$   
**find<sub>column</sub>**  
 $ColumnSpecification \times TableDetail\ LIST$   
 $\rightarrow (TableDetail \times SsqlCol \times TsqlCol)\ LIST$   
**find<sub>ident</sub>**  
 $ColumnSpecification \times IdentDetail\ LIST$   
 $\rightarrow IdentDetail\ LIST$   
**lookup\_column\_info\_look**  
 $ColumnSpecification \times Scope\ LIST$   
 $\rightarrow (TableInfo \times SsqlCol)\ RESULT$   
**lookup<sub>columninfo</sub>**  
 $TRANS\_STATE$   
 $\rightarrow ColumnSpecification$   
 $\rightarrow (TableInfo \times SsqlCol)\ RESULT$   
**maxBound**  $BoundInfo \rightarrow Class$   
**innermost**  $Scope\ LIST \rightarrow Scope\ LIST$   
**lookup\_col\_spec\_class\_look**  
 $ColumnSpecification \times Scope\ LIST$   
 $\rightarrow (TsqlRepr \times Class)\ RESULT$   
**lookup<sub>colspecclass</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Bool \times ColumnSpecification$   
 $\rightarrow (TsqlRepr \times Class)\ RESULT$   
**lookup\_col\_spec\_dinary\_look**  
 $ColumnSpecification \times Scope\ LIST \rightarrow TsqlRepr\ RESULT$   
**lookup<sub>colspecdinary</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Bool \times ColumnSpecification$   
 $\rightarrow TsqlRepr\ RESULT$   
**lookup\_col\_spec\_sterling\_look**  
 $ColumnSpecification \times Scope\ LIST \rightarrow TsqlRepr\ RESULT$   
**lookup<sub>colspecsterling</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Bool \times ColumnSpecification$   
 $\rightarrow TsqlRepr\ RESULT$   
**lookup<sub>localcolimplementation</sub>**

---

---

$TRANS\_STATE \rightarrow TsqlCol\ LIST\ RESULT$   
**lookup<sub>localcolinfo</sub>**  
 $TRANS\_STATE \rightarrow SsqlCol\ LIST\ RESULT$   
**lookup<sub>localcolspecclasses</sub>**  
 $TRANS\_STATE \rightarrow (TsqlRepr \times Class)\ LIST\ RESULT$   
**lookup<sub>localcolspecsterlings</sub>**  
 $TRANS\_STATE \rightarrow TsqlRepr\ LIST\ RESULT$   
**lookup<sub>localrowclasses</sub>**  
 $TRANS\_STATE \rightarrow TsqlRepr\ LIST\ RESULT$   
**lookup<sub>column\_row\_class\_look</sub>**  
 $TRANS\_STATE$   
 $\rightarrow ColumnSpecification \times Scope\ LIST$   
 $\rightarrow TsqlRepr\ RESULT$   
**lookup<sub>columnrowclass</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Bool \times ColumnSpecification$   
 $\rightarrow TsqlRepr\ RESULT$   
**lookup<sub>table\_row\_class\_look</sub>**  
 $TableSpecification \times Scope\ LIST \rightarrow TsqlRepr\ RESULT$   
**lookup<sub>tablerowclass</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Bool \times TableSpecification$   
 $\rightarrow TsqlRepr\ RESULT$   
**lookup<sub>table\_detail\_look</sub>**  
 $TableSpecification \times Scope\ LIST \rightarrow TableDetail\ RESULT$   
**lookup<sub>tabledetail</sub>**  
 $TRANS\_STATE \rightarrow TableSpecification \rightarrow TableDetail\ RESULT$   
**update<sub>top\_scope</sub>**  
 $TRANS\_STATE \rightarrow Scope \rightarrow TRANS\_STATE$   
**enter<sub>identifier</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Op \times ExpType \times Class$   
 $\rightarrow (TRANS\_STATE \times Op \times Op)\ RESULT$   
**enter<sub>identiferconstantclass</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Op \times ExpType \times Class$   
 $\rightarrow (TRANS\_STATE \times Op)\ RESULT$   
**extract<sub>parameter</sub>**  
 $Op \times ParamInfo\ LIST \rightarrow ParamInfo\ LIST$   
**enter<sub>parameter</sub>**  
 $TRANS\_STATE \rightarrow Op \times Value \times Class \rightarrow TRANS\_STATE\ RESULT$   
**enter<sub>corrtable</sub>**  
 $TRANS\_STATE$   
 $\rightarrow Op$   
 $\times TableName$   
 $\times TableInfo$   
 $\times SsqlCol\ LIST$   
 $\times TsqlClassName$

---

---

	$\times TsqlCol$ LIST
	$\rightarrow (TRANS\_STATE \times Op)$ RESULT
<b>enter<sub>table</sub></b>	TRANS_STATE
	$\rightarrow TableName$
	$\times TableInfo$
	$\times SsqlCol$ LIST
	$\times TsqlClassName$
	$\times TsqlCol$ LIST
	$\rightarrow (TRANS\_STATE \times Op)$ RESULT
<b>enter_scope</b>	TRANS_STATE $\rightarrow$ TRANS_STATE
<b>get<sub>tableinfo</sub></b>	TableSpecification
	$\rightarrow TableInfo$
	$\times ConstraintInfo$
	$\times SsqlCol$ LIST
	$\times TsqlClassName$
	$\times TsqlCol$ LIST
<b>lookup<sub>localtableinfo</sub></b>	TRANS_STATE $\rightarrow TableInfo$ LIST RESULT
<b>lookup<sub>paramdata</sub></b>	TRANS_STATE $\rightarrow Op \rightarrow (Value \times Class)$ RESULT
<b>in_new_scope</b>	(TRANS_STATE $\rightarrow 'a \rightarrow 'b$ ) $\rightarrow$ TRANS_STATE $\rightarrow 'a \rightarrow 'b$
<b>repr_col</b>	TsqlRepr $\rightarrow Col\_spec$ RESULT
<b>all_data_columns<sub>local</sub></b>	TRANS_STATE $\rightarrow Col\_spec$ LIST RESULT
<b>binop_type</b>	Op $\times ColType \rightarrow SwordType$ RESULT
<b>monop_type</b>	Op $\times SwordType \rightarrow SwordType$ RESULT
<b>triop_type</b>	Op $\times SwordType \times ColType \rightarrow SwordType$ RESULT
<b>set_func_type</b>	Op $\times SwordType \rightarrow SwordType$ RESULT
<b>check_boolean</b>	ExpType $\rightarrow ExpType$ RESULT
<b>check_type_conversion</b>	ColType $\rightarrow Maybe$ RESULT
<b>convert<sub>colspec</sub></b>	Col_spec $\rightarrow ColumnSpecification$
<b>class_column</b>	TRANS_STATE $\rightarrow Col\_spec \rightarrow (Col\_spec + Class)$ RESULT
<b>denote_name</b>	TsqlRepr $\rightarrow Value$
<b>column_data_test</b>	TRANS_STATE $\rightarrow Col\_spec \rightarrow Value$ LIST RESULT
<b>col_exp</b>	ExpType $\rightarrow ColType$ RESULT
<b>col_target</b>	SsqlCol $\rightarrow TsqlCol$ RESULT
<b>convert<sub>tablespec</sub></b>	Table_spec $\rightarrow TableSpecification$
<b>constant_value<sub>type</sub></b>	Value $\rightarrow SwordType$
<b>convert<sub>ssqltype</sub></b>	

---

---

	$Type \rightarrow SwordType$
<b>convert_tableSpecification_backup</b>	$Col \times Worth \rightarrow Col RESULT$
<b>convert_tableSpecification</b>	$TableSpecification \rightarrow Table\_spec RESULT$
<b>table_name</b>	$TableSpecification \rightarrow Op$
<b>convert_swordtype</b>	$SwordType \rightarrow Type$
<b>convert_type</b>	$Type \rightarrow Type$
<b>denote_classexp</b>	$ExpClass \rightarrow Value$
<b>lub_boundinfo</b>	$BoundInfo \times BoundInfo \rightarrow BoundInfo$
<b>lub_expclass</b>	$ExpClass \times ExpClass \rightarrow ExpClass$
<b>lub_type</b>	$ColType \rightarrow SwordType$
<b>lub_coltype</b>	$ColType \times ColType \rightarrow ColType$
<b>lub_worth</b>	$Worth \times Worth \rightarrow Worth$
<b>lub_exp</b>	$ExpType \times ExpType \rightarrow ExpType$
<b>lub_ssqlname</b>	$SsqlName \times SsqlName \rightarrow SsqlName$
<b>lub_ssqlcol</b>	$SsqlCol \times SsqlCol \rightarrow SsqlCol$
<b>lub_tableinfo</b>	$TableInfo \times TableInfo \rightarrow TableInfo$
<b>lub_tsqlclassname</b>	$TsqlClassName \times TsqlClassName \rightarrow TsqlClassName$
<b>lub_tsqlname</b>	$TsqlName \times TsqlName \rightarrow TsqlName$
<b>lub_tsqlcol</b>	$TsqlCol \times TsqlCol \rightarrow TsqlCol$
<b>make_sv</b>	$TsqlCol \times TsqlCol \rightarrow Value LIST RESULT$
<b>remove_constants</b>	$(Col\_spec + Class) LIST \rightarrow Col\_spec LIST$
<b>remove_nulls</b>	$TsqlRepr LIST \rightarrow TsqlRepr LIST$
<b>upper</b>	$ExpClass \rightarrow Class$
<b>make_case</b>	$Value \times ExpClass \rightarrow Value$
<b>simplify_and</b>	$Value LIST \times ExpClass LIST \rightarrow Value \times ExpClass$
<b>simplify_or</b>	$Value LIST \times ExpClass LIST \rightarrow Value \times ExpClass$
<b>constant_value_data</b>	$Value \rightarrow Value$
<b>dinary_columns</b>	$TRANS\_STATE \rightarrow Col\_spec LIST \rightarrow Col\_spec LIST RESULT$

---

**sterling\_columns***TRANS\_STATE* → *Col\_spec LIST* → *Col\_spec LIST RESULT***tuple\_list<sub>maxrowclass</sub>***TRANS\_STATE* → *Tuple\_list* → *ExpClass***upb\_row\_class***TableInfo* → *Class***value<sub>data</sub>***TRANS\_STATE* → *Value* → *Value***value<sub>type</sub>***TRANS\_STATE* → *Value* → *ExpType***value<sub>class</sub>***(TRANS\_STATE* → *Value* → *InternalExpClass RESULT*)  
→ *TRANS\_STATE*  
→ *Value*  
→ *ExpClass RESULT***tuple\_list<sub>data</sub>***TRANS\_STATE* → *Tuple\_list* → *Tuple\_list RESULT***tuple\_list<sub>type</sub>***TRANS\_STATE* → *Tuple\_list* → *Tuple\_list RESULT***from\_spec<sub>center</sub>***TRANS\_STATE* → *From\_spec* → *From\_spec RESULT***select\_list<sub>type</sub>***TRANS\_STATE* → *Select\_list* → *ExpType RESULT***select\_value<sub>type</sub>***TRANS\_STATE* → *Value* → *ExpType RESULT***tuple\_list<sub>info</sub>***TRANS\_STATE*  
→ *Tuple\_list*  
→ *(TableName × TableInfo × SsqlCol LIST) RESULT***tuple\_list<sub>make</sub>***TRANS\_STATE*  
→ *Tuple\_list × TsqlClassName × TsqlCol LIST*  
→ *Tuple\_list RESULT***from\_spec<sub>info</sub>***TRANS\_STATE*  
→ *From\_spec*  
→ *(TableInfo*  
× *SsqlCol LIST*  
× *TsqlClassName*  
× *TsqlCol LIST) RESULT***select\_list<sub>info</sub>***TRANS\_STATE* → *Select\_list* → *SsqlCol LIST RESULT***select\_value<sub>info</sub>***TRANS\_STATE* → *Value* → *SsqlCol RESULT***value<sub>info</sub>***TRANS\_STATE* → *Value* → *SsqlCol RESULT***internal\_value<sub>class</sub>***TRANS\_STATE* → *Value* → *InternalExpClass RESULT*

---

**tuple\_list<sub>class</sub>**      *TRANS\_STATE* → *Tuple\_list* → *Tuple\_list*  
**select\_list<sub>class</sub>**      *TRANS\_STATE* → *Select\_list* → *Select\_list RESULT*  
**select\_value<sub>class</sub>**      *TRANS\_STATE* → *Value* → *Value RESULT*  
**select\_list<sub>make</sub>**      *TRANS\_STATE*  
                             → *Select\_list* × *TsqlClassName* × *TsqlCol LIST*  
                             → *Value LIST RESULT*  
**select\_value<sub>make</sub>**      *TRANS\_STATE* → *Value* × *TsqlCol LIST* → *Value LIST RESULT*  
**make\_col**                *TRANS\_STATE* → *Value* × *TsqlCol* → *Value LIST RESULT*  
**make<sub>dinary</sub>**            *TRANS\_STATE* → *Value* × *ExpType* → *Value RESULT*  
**make<sub>sterling</sub>**         *TRANS\_STATE* → *Value* × *ExpType* → *Value RESULT*  
**select\_list<sub>data</sub>**        *TRANS\_STATE* → *Select\_list* → *Select\_list RESULT*  
**select\_value<sub>data</sub>**      *TRANS\_STATE* → *Value* → *Value RESULT*  
**tuple\_list<sub>makeouter</sub>**    *TRANS\_STATE*  
                             → *Tuple\_list* × *Bool* × *TsqlClassName* × *TsqlCol LIST*  
                             → (*Tuple\_list* × *Query LIST*) *RESULT*  
**tuple\_list<sub>outerinfo</sub>**    *TRANS\_STATE* → *Tuple\_list* → *Bool RESULT*  
**transform<sub>selectquery</sub>**    *TRANS\_STATE*  
                             → *Query*  
                             → (*SsqlCol LIST*  
                                × *Query*  
                                × *Bool*  
                                × *TsqlClassName*  
                                × *TsqlCol LIST*  
                                × *Query LIST*) *RESULT*  
**query<sub>selectquery</sub>**      *TRANS\_STATE* → *Query* → *Query RESULT*  
**STP**                    (*Query*, *FILTER\_PARS*) *STP\_TYPE*

## 26.4 Types

### TRANS\_STATE

## 26.5 Definitions

### internalError

### wrongWorth

**notTrigger**  
**onlyInTriggers**  
**notMonadic**  
**notDyadic**  
**notTriadic**  
**notSetFunction**  
**noSuchParameter**  
**noScope**  
**wrongScope**  
**ambiguousName**  
**emptyUnionList**

$\vdash true$

**TRANS\_STATE**  $\vdash \exists f \bullet TypeDefn (\lambda x \bullet true) f$

**MkTRANS\_STATE**

**st\_stack**

**query\_class**

**query\_constants\_class**

**client\_clearance**

$\vdash \forall t \ x1 \ x2 \ x3 \ x4$

- $st\_stack (MkTRANS\_STATE \ x1 \ x2 \ x3 \ x4) = x1$
- $\wedge query\_class (MkTRANS\_STATE \ x1 \ x2 \ x3 \ x4) = x2$
- $\wedge query\_constants\_class$   
     $(MkTRANS\_STATE \ x1 \ x2 \ x3 \ x4)$   
     $= x3$
- $\wedge client\_clearance (MkTRANS\_STATE \ x1 \ x2 \ x3 \ x4)$   
     $= x4$
- $\wedge MkTRANS\_STATE$   
     $(st\_stack \ t)$   
     $(query\_class \ t)$   
     $(query\_constants\_class \ t)$   
     $(client\_clearance \ t)$   
     $= t$

**check\_enum**

**check\_fixed**

**check\_floating**

**check\_interval**

**check\_time**

**timeFormatToInterval**

**unique\_name**

**contextual\_data**

**default\_directory**

$\vdash true$

**init\_trans\_state**

$\vdash \forall c$

- $init\_trans\_state \ c$   
     $= MkTRANS\_STATE (MkST\_STACK \ [] \ []) \ c \ c \ c$

**find<sub>column</sub>**

$\vdash \forall cs \ tdl$

---

```

• findcolumn (cs, tdl)
  = (let look (n, td, scl, tcl)
      = (let sctcl = splice scl tcl
          in let sctcl'
              = sctcl
                ↑ {(sc, tc)
                  |sc_name sc = mk_name_s n}
              in Map
                (λ (sc, tc) • (td, sc, tc))
                sctcl')
          in let do1td td
              = CASE
                cs
                [WHEN_anonymous_column col
                  • look
                    (col, td, td_columns td,
                     td_implementation td);
                  WHEN_specific (ts, col)
                  • CASE
                    (td_tableName td)
                    [WHEN_nametn ts
                      • look
                        (col, td,
                         td_columns td,
                         td_implementation
                          td);
                      OTHERS []]
                    in Flat (Map do1td tdl))

```

**find<sub>ident</sub>**

```

⊢ ∀ cs idl
• findident (cs, idl)
  = CASE
    cs
    [WHEN_specific (t, c) • [];
     WHEN_anonymous_column col
     • idl ↑ {id | col = id_identName id}]

```

**lookup\_column\_info\_look**

```

⊢ ConstSpec
(λ lookup_column_info_look'
  • ∀ cs ti outer
    • lookup_column_info_look' (cs, [])
      = Exception [noSuchColumn]
    ∧ lookup_column_info_look'
      (cs, outer @ [ti])
    = (let t = s_tables ti
        in let tdsetcl
            = findcolumn (cs, t)
            in if tdsetcl = []

```

---

```

      then
        lookup_column_info_look'
          (cs, outer)
      else if Tail tdsctcl = []
      then
        let (td, sc, tc) = Head tdsctcl
        in Ok (td_info td, sc)
      else Exception [ambiguousName]))
lookup_column_info_look

lookup_column_info
  ⊢ ∀ st cs
  • lookup_column_info st cs
    = lookup_column_info_look
      (cs, symbolTable (st_stack st))

maxBound
  ⊢ ∀ bi
  • maxBound bi
    = CASE bi [WHEN_upb c • c; WHEN_constant c • c]

innermost
  ⊢ ConstSpec
  (λ innermost'
  • ∀ outer inner
  • innermost' [] = []
    ∧ innermost' (outer @ [inner])
    = (let tds = s_tables inner
    in if tds = []
    then innermost' outer @ [inner]
    else [inner]))

innermost
lookup_col_spec_class_look
  ⊢ ConstSpec
  (λ lookup_col_spec_class_look'
  • ∀ cs outer ti
  • lookup_col_spec_class_look' (cs, [])
    = Exception [noSuchColumn]
    ∧ lookup_col_spec_class_look'
      (cs, outer @ [ti])
    = (let (t, i)
    = (s_tables ti, s_identifiers ti)
    in let cds = find_column (cs, t)
    and ids = find_ident (cs, i)
    in if ids = []
    then
      if cds = []
      then
        lookup_col_spec_class_look'
          (cs, outer)
      else if Tail cds = []
      then
        let (td, sc, tc) = Head cds

```

---

---

```

    in let u
      = maxBound
        (sc_col_class sc)
    in CASE
      (tc_class_name tc)
      [WHEN_anontc
        (Exception
          [internalError]);
        WHEN_nametc s
          • Ok
            (mk_column
              (td_genCorr td,
s), u);
          WHEN_constanttc c
            • Ok
              (mk_constant_class
                c, u)]
        else Exception [ambiguousName]
      else if cds = [] ^ Tail ids = []
      then
        let id = Head ids
          in let cl = id_lubid id
            in CASE
              (id_cName id)
              [WHEN_nonet
                (Ok
                  (mk_constant_class
cl, cl)); WHEN_anont (Exception [internalError]);
                WHEN_namet s
                  • Ok
                    (mk_local_identifier
s, cl)]
              else Exception [ambiguousName]])
        lookup_col_spec_class_look

```

**lookup<sub>colspecclass</sub>**

```

  ⊢ ∀ st flg cs
    • lookupcolspecclass st (flg, cs)
      = (if flg
        then
          lookup_col_spec_class_look
            (cs, innermost (symbolTable (st_stack st)))
        else
          lookup_col_spec_class_look
            (cs, symbolTable (st_stack st)))

```

**lookup\_col\_spec\_dinary\_look**

```

  ⊢ ConstSpec
    (λ lookup_col_spec_dinary_look'
      • ∀ cs outer ti

```

---

```

• lookup_col_spec_dinary_look' (cs, [])
  = Exception [noSuchColumn]
  ∧ lookup_col_spec_dinary_look'
    (cs, outer @ [ti])
  = (let (t, i)
     = (s_tables ti, s_identifiers ti)
     in let cds = find_column (cs, t)
        and ids = find_ident (cs, i)
        in if ids = []
           then
             if cds = []
               then
                 lookup_col_spec_dinary_look'
                   (cs, outer)
             else if Tail cds = []
               then
                 let (td, sc, tc) = Head cds
                     in CASE
                       (tc_dinary_name tc)
                       [WHEN_none_t
                        (Exception
                         [internalError]);
                        WHEN_anon_t
                        (Exception
                         [internalError]);
                        WHEN_name_t s
                        • Ok
                          (mk_column
                           (td_genCorr td,
s))]
                       else Exception [ambiguousName]
             else if cds = [] ∧ Tail ids = []
               then
                 let id = Head ids
                     in let (st, w) = id_info id
                        in if w = dinary
                           then
                             Ok
                               (mk_local_identifier
                                (id_vName id))
                           else Ok c_constant_null
             else Exception [ambiguousName]))
  lookup_col_spec_dinary_look

```

**lookupcolspecdinary**

```

⊢ ∀ st flg cs
  • lookupcolspecdinary st (flg, cs)
    = (if flg
       then

```

---

```

      lookup_col_spec_dinary_look
      (cs, innermost (symbolTable (st_stack st)))
    else
      lookup_col_spec_dinary_look
      (cs, symbolTable (st_stack st))
lookup_col_spec_sterling_look
  ⊢ ConstSpec
  (λ lookup_col_spec_sterling_look'
    • ∀ cs outer ti
    • lookup_col_spec_sterling_look' (cs, [])
      = Exception [noSuchColumn]
    ∧ lookup_col_spec_sterling_look'
      (cs, outer @ [ti])
    = (let (t, i)
      = (s_tables ti, s_identifiers ti)
      in let cds = find_column (cs, t)
        and ids = find_ident (cs, i)
        in if ids = []
        then
          if cds = []
          then
            lookup_col_spec_sterling_look'
              (cs, outer)
          else if Tail cds = []
          then
            let (td, sc, tc) = Head cds
            in CASE
              (tc_sterling_name tc)
              [WHEN_none_t
                (Exception
                  [internalError]);
                WHEN_anon_t
                (Exception
                  [internalError]);
                WHEN_name_t s
                • Ok
                  (mk_column
                    (td_genCorr td,
s))]
              else Exception [ambiguousName]
            else if cds = [] ∧ Tail ids = []
            then
              let id = Head ids
              in let (st, w) = id_info id
                in if w = sterling
                then
                  Ok
                    (mk_local_identifier

```

---

---

```

                                (id_vName id))
                                else Ok c_constant_null
                                else Exception [ambiguousName]))
lookup_col_spec_sterling_look

lookup_col_spec_sterling
  ⊢ ∀ st flg cs
  • lookup_col_spec_sterling st (flg, cs)
    = (if flg
       then
         lookup_col_spec_sterling_look
         (cs, innermost (symbolTable (st_stack st)))
       else
         lookup_col_spec_sterling_look
         (cs, symbolTable (st_stack st)))

lookup_local_col_implementation
  ⊢ ∀ st
  • lookup_local_col_implementation
    st
    = (let extract_implementation sc
        = Fold
          $@
          (Map td_implementation (s_tables sc))
          []
        in let trs
            = Fold
              $@
              (Map
                extract_implementation
                (innermost
                  (symbolTable (st_stack st))))
              []
            in if trs = []
               then Exception [noScope]
               else Ok trs)

lookup_local_col_info
  ⊢ ∀ st
  • lookup_local_col_info st
    = (let extract_columns sc
        = Fold $@ (Map td_columns (s_tables sc)) []
        in let trs
            = Fold
              $@
              (Map
                extract_columns
                (innermost
                  (symbolTable (st_stack st))))
              []
            in if trs = []

```

---

---

```

    then Exception [noScope]
    else Ok trs)

lookuplocalcolspecclasses
  ⊢ ∀ st
  • lookuplocalcolspecclasses st
    = (let look2 (corr, sc, tc)
      = (let u = maxBound (sc_col_class sc)
        in CASE
          (tc_class_name tc)
          [WHEN_ anontc
            (Exception [internalError]);
           WHEN_ nametc s
            • Ok (mk_column (corr, s), u);
           WHEN_ constanttc c
            • Ok (mk_constant_class c, u)])
      in let look1 td
        = at3
          (Map look2)
          (seq
            (# (td_columns td))
            (td_genCorr td), td_columns td,
            td_implementation td)
        in let look sc
          = Fold $@ (Map look1 (s_tables sc)) []
        in let trs
          = Fold
            $@
            (Map
              look
              (innermost
                (symbolTable
                  (st_stack st))))
            []
        in if trs = []
          then Exception [noScope]
          else ListOk trs)

lookuplocalcolspecsterlings
  ⊢ ∀ st
  • lookuplocalcolspecsterlings st
    = (let look2 (corr, tc)
      = CASE
        (tc_sterling_name tc)
        [WHEN_ nonet (Ok c_constant_null);
         WHEN_ anont
          (Exception [internalError]);
         WHEN_ namet s
          • Ok (mk_column (corr, s))]
      in let look1 td

```

```

= at2
  (Map look2)
  (seq
    (# (td_columns td))
    (td_genCorr td),
    td_implementation td)
in let look sc
  = Fold $@ (Map look1 (s_tables sc)) []
in let trs
  = Fold
    $@
    (Map
      look
      (innermost
        (symbolTable
          (st_stack st))))
    []
in if trs = []
  then Exception [noScope]
  else ListOk trs

```

**lookup<sub>localrowclasses</sub>**

```

⊢ ∀ st
• lookuplocalrowclasses st
  = (let look1 td
    = CASE
      (td_rowClass td)
      [WHEN_ anontc
        (Exception [internalError]);
      WHEN_ nametc s
        • Ok
          (mk_column (td_genCorr td, s));
      WHEN_ constanttc c
        • Ok (mk_constant_class c)]
in let look sc = Map look1 (s_tables sc)
in let trs
  = Fold
    $@
    (Map
      look
      (innermost
        (symbolTable (st_stack st))))
    []
in if trs = []
  then Exception [noScope]
  else ListOk trs)

```

**lookup<sub>column\_row\_class\_look</sub>**

```

⊢ ConstSpec
  (λ lookupcolumn_row_class_look'

```

---

- $\forall st\ cs\ outer\ ti$ 
  - $lookup\_column\_row\_class\_look'\ st\ (cs, [])$   
 $=\ Exception\ [noSuchColumn]$   
 $\wedge\ lookup\_column\_row\_class\_look'$   
 $st$   
 $(cs, outer\ @\ [ti])$   
 $=\ (let\ (t, i)$   
 $=\ (s\_tables\ ti, s\_identifiers\ ti)$   
 $in\ let\ cds = find\_column\ (cs, t)$   
 $and\ ids = find\_ident\ (cs, i)$   
 $in\ if\ ids = []$   
 $then$   
 $if\ cds = []$   
 $then$   
 $lookup\_column\_row\_class\_look'$   
 $st$   
 $(cs, outer)$   
 $else\ if\ Tail\ cds = []$   
 $then$   
 $let\ (td, sc, tc) = Head\ cds$   
 $in\ CASE$   
 $(td\_rowClass\ td)$   
 $[WHEN\_anon_{tc}$   
 $(Exception$   
 $[internalError]);$   
 $WHEN\_name_{tc}\ s$   
    - $Ok$   
 $(mk\_column$   
 $(td\_genCorr\ td,$   
 $s));\ WHEN\_constant_{tc}\ c$ 
      - $Ok\ (mk\_constant\_class\ c)$   
 $else\ Exception\ [ambiguousName]$ $else\ if\ cds = [] \wedge\ Tail\ ids = []$   
 $then$   
 $let\ id = Head\ ids$   
 $in\ Ok$   
 $(mk\_constant\_class$   
 $(query\_class\ st))$   
 $else\ Exception\ [ambiguousName]))$   
 $lookup\_column\_row\_class\_look$

**lookup<sub>columnrowclass</sub>**

- $\vdash\ \forall\ st\ flg\ cs$ 
  - $lookup_{columnrowclass}\ st\ (flg, cs)$   
 $=\ (if\ flg$   
 $then$   
 $lookup\_column\_row\_class\_look$   
 $st$   
 $(cs, innermost\ (symbolTable\ (st\_stack\ st)))$   
 $else$

---

```

lookup_column_row_class_look
  st
  (cs, symbolTable (st_stack st))
lookup_table_row_class_look
  ⊢ ConstSpec
  (λ lookup_table_row_class_look'
    • ∀ ts outer ti
    • lookup_table_row_class_look' (ts, [])
      = Exception [noSuchTable]
    ∧ lookup_table_row_class_look'
      (ts, outer @ [ti])
      = (let look1 (ts, td)
        = CASE
          (td_tableName td)
          [WHEN_anontn [];
           WHEN_nametn tn
            • if ts = tn
              then
                CASE
                  (td_rowClass td)
                  [WHEN_anontc
                   [Exception
                    [internalError]];
                   WHEN_nametc s
                    • [Ok
                     (mk_column
                      (td_genCorr td, s))]
                   WHEN_constanttc
                     c
                    • [Ok
                     (mk_constant_class
                      c))]
                  else []
                ]
          ]
        in let (tds, ids)
          = (s_tables ti, s_identifiers ti)
          in let trs
            = Fold
              $@
              (at2
               (Map look1)
               (seq (# tds) ts, tds))
              []
          in if trs = []
            then
              lookup_table_row_class_look'
                (ts, outer)
            else if Tail trs = []
              then Head trs

```

---

```

                else Exception [ambiguousName]))
lookup_table_row_class_look
lookuptablerowclass
  ⊢ ∀ st flg ts
    • lookuptablerowclass st (flg, ts)
      = (if flg
        then
          lookup_table_row_class_look
            (ts, innermost (symbolTable (st_stack st)))
        else
          lookup_table_row_class_look
            (ts, symbolTable (st_stack st)))
lookup_table_detail_look
  ⊢ ConstSpec
    (λ lookup_table_detail_look'
      • ∀ ts outer ti
        • lookup_table_detail_look' (ts, [])
          = Exception [noSuchTable]
        ∧ lookup_table_detail_look'
          (ts, outer @ [ti])
          = (let look1 (ts, td)
            = CASE
              (td_tableName td)
              [WHEN_anontn [];
               WHEN_nametn tn
                • if ts = tn
                  then [td]
                  else []]
            in let (tds, ids)
              = (s_tables ti, s_identifiers ti)
            in let tis
              = Fold
                $@
                (at2
                 (Map look1)
                 (seq (# tds) ts, tds))
                []
            in if tis = []
              then
                lookup_table_detail_look'
                  (ts, outer)
              else if Tail tis = []
                then Ok (Head tis)
                else Exception [ambiguousName]))
lookup_table_detail_look
lookuptabledetail
  ⊢ ∀ st ts
    • lookuptabledetail st ts

```

---

```

= lookup_table_detail_look
  (ts, symbolTable (st_stack st))

```

**update\_top\_scope**

```

⊢ ∀ st sc
• update_top_scope st sc
  = (let stk = st_stack st
     in let symt = symbolTable stk
        in let outer = Rev (Tail (Rev symt))
           in let symt' = outer @ [sc]
              in MkTRANS_STATE
                (MkST_STACK symt' (parameterTable stk))
                (query_class st)
                (query_constants_class st)
                (client_clearance st))

```

**enter<sub>identifier</sub>**

```

⊢ ∀ st name et up
• enteridentifier st (name, et, up)
  = (let fynd (n, id)
     = (if n = id_identName id
        then [id]
        else []))
  in let sl = symbolTable (st_stack st)
     in if sl = []
        then Exception [noScope]
     else
       (let outer = Rev (Tail (Rev sl))
          in let ti = Head (Rev sl)
             in let (tds, ids)
                = (s_tables ti, s_identifiers ti)
                 in if tds = []
                    then
                      let unv = unique_name st
                         in let unc = unique_name st
                            in let id
                               = MkIdentDetail
                                 name
                                 et
                                 up
                                 unv
                                 (mk_namet unc)
                               in if
                                  ¬ at2
                                  (Map fynd)
                                  (seq (# ids) name,
                                   ids)
                                  = []
                               then
                                 Exception [ambiguousName]

```

---

```

else
  (let sc'
    = MkScope
      []
      (ids @ [id])
    in Ok
      (update_top_scope
        st
        sc', unv, unc))
else Exception [wrongScope]))

enteridentiferconstantclass
  ⊢ ∀ st name et clasf
  • enteridentiferconstantclass
    st
    (name, et, clasf)
  = (let fynd (n, id)
    = (if n = id_identName id
      then [id]
      else []))
  in let sl = symbolTable (st_stack st)
  in if sl = []
  then Exception [noScope]
  else
    (let outer = Rev (Tail (Rev sl))
    in let ti = Head (Rev sl)
    in let (tds, ids)
      = (s_tables ti, s_identifiers ti)
    in if tds = []
    then
      let unv = unique_name st
      in let id
        = MkIdentDetail
          name
          et
          clasf
          unv
          c_anont
      in if
        ¬ at2
          (Map fynd)
          (seq (# ids) name, ids)
        = []
      then Exception [ambiguousName]
    else
      (let sc'
        = MkScope
          []
          (ids @ [id])

```

---

---

```

                                in Ok
                                (update_top_scope st sc',
                                 unv))
                                else Exception [wrongScope]))
extractparameter
  ⊢ ∀ name l
  • extractparameter (name, l)
    = l ⊢ {pi|pi_name pi = name}
enterparameter
  ⊢ ∀ st name v clasf
  • enterparameter st (name, v, clasf)
    = (let sl = symbolTable (st_stack st)
       in let pt = parameterTable (st_stack st)
          in if sl = []
             then Exception [noScope]
             else if
                  extractparameter (name, pt) = []
                then
                  Ok
                  (MkTRANS_STATE
                   (MkST_STACK
                    sl
                    (pt
                     @ [MkParamInfo
                        name
                        v
                        clasf]))
                   (query_class st)
                   (query_constants_class st)
                   (client_clearance st))
                else Exception [ambiguousName])
entercorrtable
  ⊢ ∀ st cn ts ti scs rcn tcs
  • entercorrtable
    st
    (cn, ts, ti, scs, rcn, tcs)
    = (let sl = symbolTable (st_stack st)
       in if sl = []
          then Exception [noScope]
          else
            (let outer = Rev (Tail (Rev sl))
               in let t = Head (Rev sl)
                  in let (tds, ids)
                       = (s_tables t, s_identifiers t)
                     in if ids = []
                        then
                          let gc = unique_name st
                             in let td

```

---

```

      = MkTableDetail
        ts
        (mk_name_s cn)
        gc
        ti
        scs
        rcn
        tcs
        (MkConstraintInfo
         []
         []
         []
         []
         []))
    in let sc'
      = MkScope (tds @ [td]) []
    in Ok
      (update_top_scope st sc', gc)
    else Exception [wrongScope]))

```

**enter<sub>table</sub>**

```

⊢ ∀ st ts ti scs rcn tcs
• entertable st (ts, ti, scs, rcn, tcs)
  = (let sl = symbolTable (st_stack st)
    in if sl = []
    then Exception [noScope]
    else
      (let outer = Rev (Tail (Rev sl))
        in let t = Head (Rev sl)
          in let (tds, ids)
            = (s_tables t, s_identifiers t)
          in if ids = []
          then
            let gc = unique_name st
              in let td
                = MkTableDetail
                  ts
                  c_anon_s
                  gc
                  ti
                  scs
                  rcn
                  tcs
                  (MkConstraintInfo
                   []
                   []
                   []
                   []
                   []))

```

---

```

                                in let sc'
                                  = MkScope (tds @ [td]) []
                                in Ok
                                  (update_top_scope st sc', gc)
                                else Exception [wrongScope]))
enter_scope    ⊢ ∀ st
  • enter_scope st
    = (let stk = st_stack st
        in let symt = symbolTable stk
            in let outer = Rev (Tail (Rev symt))
                in let symt' = outer @ []
                    in MkTRANS_STATE
                      (MkST_STACK symt' (parameterTable stk))
                      (query_class st)
                      (query_constants_class st)
                      (client_clearance st))

get_tableinfo    ⊢ true

lookup_localtableinfo
  ⊢ ∀ st
  • lookup_localtableinfo st
    = (let look_sc = Map td_info (s_tables sc)
        in let trs
            = Fold
              $@
              (Map
                look
                (innermost
                  (symbolTable (st_stack st))))
              []
            in if trs = []
                then Exception [noScope]
                else Ok trs)

lookup_paramdata
  ⊢ ∀ st name
  • lookup_paramdata st name
    = (let infos
        = extract_parameter
          (name, parameterTable (st_stack st))
        in if infos = []
            then Exception [noSuchParameter]
            else if Tail infos = []
                then
                  let info = Head infos
                      in Ok (pi_val_p info, pi_clasf info)
                else Exception [internalError])

in_new_scope  ⊢ ∀ what
  • in_new_scope what

```

---

---

```

      = (λ st a • what (enter_scope st) a)
repr_col    ⊢ ∀ tr
  • repr_col tr
    = CASE
      tr
      [WHEN_local_identifier name
        • Ok (denote_col_spec [name]);
        WHEN_column (corr, col)
          • Ok (denote_col_spec [corr; col]);
        WHEN_constant_class c
          • Exception [internalError];
        WHEN_constant_null
          (Exception [internalError])]
all_data_columns_local
  ⊢ ∀ st
  • all_data_columns_local st
    = Try
      (ListOk o Map repr_col)
      (lookup_local_colspecsterlings
        st)
check_type_conversion
check_boolean
set_func_type
triop_type
monop_type
binop_type    ⊢ true
convert_colspec
  ⊢ ConstSpec
    (λ convert_colspec'
      • ∀ il
        • convert_colspec' (denote_col_spec il)
          = (let col = Head (Rev il)
              in let dir = Head (Tail (Rev il))
                  in let tab = Tail (Tail (Rev il))
                      in mk_specific
                        (mk_absolute (tab, dir), col)))
convert_colspec
class_column  ⊢ ∀ st cs
  • class_column st cs
    = (let csc
        = lookup_colspecclass
          st
          (true, convert_colspec cs)
        in if isVal csc
          then
            let (tr, lub_cl) = destVal csc
              in CASE
                tr

```

---

---

```

      [WHEN_local_identifier name
        • Ok (InL (denote_col_spec [name]));
        WHEN_column (gen_corr, gen_col)
          • Ok
            (InL
              (denote_col_spec
                [gen_corr; gen_col]));
        WHEN_constant_class cl • Ok (InR cl);
        WHEN_constant_null
          (Exception [internalError])]
    else giveError (destError csc)
denote_name ⊢ ∀ tr
  • denote_name tr
    = CASE
      tr
      [WHEN_local_identifier s
        • contents (denote_col_spec [s]);
        WHEN_column (cn, col)
          • contents (denote_col_spec [cn; col]);
        WHEN_constant_class c • denote_class c;
        WHEN_constant_null denote_null]
column_data_test
  ⊢ ∀ st cs
  • column_data_test st cs
    = (let csc
      = lookup_col_spec_class
        st
        (true, convert_col_spec cs)
      in if isVal csc
      then
        let (tr, u) = destVal csc
        in if client_clearance st dom u
        then Ok []
        else
          (let cc
            = denote_class
              (client_clearance st)
          in Ok
            [binop "dom" (cc, denote_name tr)])
      else giveError (destError csc))
col_exp
  ⊢ ∀ t w
  • col_exp (t, w)
    = (if w = dinary
      then Ok (c_nullType, t)
      else if w = sterling
      then Ok (t, c_nullType)
      else if w = worthless
      then Ok (t, c_nullType)

```

---

---

```

      else Exception [wrongType])
col_target  ⊢ ∀ sc
  • col_target sc
    = (let bound bi
      = CASE
        bi
        [WHEN_upb c • c_anontc;
         WHEN_constant c • mk_constanttc c]
      in let target ((s, d), c)
        = (if s = c_nullType ∧ d = c_nullType
          then Exception [internalError]
          else if d = c_nullType
            then Ok (MkTsqlCol c_anont c_nonet c)
            else if s = c_nullType
              then Ok (MkTsqlCol c_nonet c_anont c)
              else
                Ok (MkTsqlCol c_anont c_anont c))
          in target
            (sc_type_field sc,
             bound (sc_col_class sc)))
converttableSpec
  ⊢ ConstSpec
    (λ converttableSpec'
     • ∀ il
       • converttableSpec'
         (denote_table_spec il)
         = (let tab = Head (Rev il)
           in let dir = Tail (Rev il)
             in mk_absolute (dir, tab)))
     converttableSpec
convertssqltype
constant_valuetype
  ⊢ true
convert_tableSpecification_backup
  ⊢ ∀ d dirs n
    • convert_tableSpecification_backup ([], 0) = Ok []
      ∧ convert_tableSpecification_backup
        (Cons d dirs, 0)
      = Ok (Cons d dirs)
      ∧ convert_tableSpecification_backup ([], n + 1)
      = Exception [noSuchDirectory]
      ∧ convert_tableSpecification_backup
        (Cons d dirs, n + 1)
      = convert_tableSpecification_backup (dirs, n)
converttableSpecification
  ⊢ ∀ ts
    • converttableSpecification ts
      = CASE

```

---

---

```

    ts
  [WHEN_absolute (directory, table)
    • Ok
      (denote_table_spec
        (directory @ [table]));
    WHEN_default (up, directory, table)
    • (let dir
        = convert_tableSpecification_backup
          (default_directory, up)
        in if isError dir
          then giveError (destError dir)
          else
            Ok
              (denote_table_spec
                (destVal dir
                  @ directory
                  @ [table])))])

table_name    ⊢ ∀ ts
  • table_name ts
    = (let dot s = s @ "."
        in CASE
          ts
          [WHEN_absolute (dir, tab)
            • if dir = []
              then tab
              else Fold $@ (Map dot dir) [] @ tab;
            WHEN_default (up, dir, tab)
            • if dir = []
              then Flat (seq up "-") @ tab
              else
                Flat (seq up "-")
                  @ Fold $@ (Map dot dir) []
                  @ tab])

convert_swordtype
  ⊢ true

convert_type
  ⊢ ∀ t
    • convert_type t
      = convert_swordtype
        (convert_sqltype t)

denote_classexp
  ⊢ ∀ ec
    • denote_classexp ec
      = CASE
        ec
        [WHEN_variable (v, c) • v;
         WHEN_constantec c • denote_class c]

lub_boundinfo

```

---

---

```

⊢ ∀ bi1 bi2
  • lubboundinfo (bi1, bi2)
    = CASE
      bi1
      [WHEN_upb c1
        • CASE
          bi2
          [WHEN_upb c2 • mk_upb (c1 lub c2);
            WHEN_constant c2
              • mk_upb (c1 lub c2)];
          WHEN_constant c1
            • CASE
              bi2
              [WHEN_upb c2 • mk_upb (c1 lub c2);
                WHEN_constant c2
                  • if c1 = c2
                    then bi1
                    else mk_upb (c1 lub c2)]]

```

**lub<sub>expclass</sub>**

```

⊢ ∀ ec1 ec2
  • lubexpclass (ec1, ec2)
    = CASE
      ec1
      [WHEN_variable (v1, c1)
        • CASE
          ec2
          [WHEN_variable (v2, c2)
            • mk_variable
              (binop "lub" (v1, v2),
                c1 lub c2);
            WHEN_constantec c2
              • mk_variable
                (binop
                  "lub"
                  (v1,
                    denote_class c2),
                  c1 lub c2)];
          WHEN_constantec c1
            • CASE
              ec2
              [WHEN_variable (v2, c2)
                • mk_variable
                  (binop
                    "lub"
                    (v2,
                      denote_class c1),
                    c1 lub c2);
                WHEN_constantec c2

```

---

	• $mk\_constant_{ec} (c1 \text{ lub } c2)]]$
<b>lub<sub>type</sub></b>	$\vdash true$
<b>lub<sub>coltype</sub></b>	$\vdash \forall s1 \ d1 \ s2 \ d2$ <ul style="list-style-type: none"> <li>• <math>lub_{coltype} ((s1, d1), s2, d2)</math></li> <li><math>= (lub_{type} (s1, s2), lub_{type} (d1, d2))</math></li> </ul>
<b>lub<sub>worth</sub></b>	$\vdash \forall w1 \ w2$ <ul style="list-style-type: none"> <li>• <math>lub_{worth} (w1, w2)</math></li> <li><math>= (if \ w1 = worthless</math></li> <li style="padding-left: 2em;"><math>then \ w1</math></li> <li style="padding-left: 2em;"><math>else \ if \ w2 = worthless</math></li> <li style="padding-left: 2em;"><math>then \ w1</math></li> <li style="padding-left: 2em;"><math>else \ if \ w1 = worthless</math></li> <li style="padding-left: 2em;"><math>then \ w2</math></li> <li style="padding-left: 2em;"><math>else \ priceless)</math></li> </ul>
<b>lub<sub>exp</sub></b>	$\vdash \forall t1 \ w1 \ t2 \ w2$ <ul style="list-style-type: none"> <li>• <math>lub_{exp} ((t1, w1), t2, w2)</math></li> <li><math>= (lub_{type} (t1, t2), lub_{worth} (w1, w2))</math></li> </ul>
<b>lub<sub>sqlname</sub></b>	$\vdash \forall sn1 \ sn2$ <ul style="list-style-type: none"> <li>• <math>lub_{sqlname} (sn1, sn2)</math></li> <li><math>= CASE</math></li> <li style="padding-left: 2em;"><math>sn1</math></li> <li style="padding-left: 2em;">[<math>WHEN\_name_s \ s1</math></li> <li style="padding-left: 4em;">• <math>CASE</math></li> <li style="padding-left: 6em;"><math>sn2</math></li> <li style="padding-left: 6em;">[<math>WHEN\_name_s \ s2</math></li> <li style="padding-left: 8em;">• <math>if \ s1 = s2</math></li> <li style="padding-left: 8em;"><math>then \ sn1</math></li> <li style="padding-left: 8em;"><math>else \ c\_anon_s; OTHERS \ c\_anon_s];</math></li> <li style="padding-left: 4em;"><math>OTHERS \ c\_anon_s]</math></li> </ul>
<b>lub<sub>sqlcol</sub></b>	$\vdash ConstSpec$ <ul style="list-style-type: none"> <li>(<math>\lambda \ lub_{sqlcol}'</math></li> <li>• <math>\forall n1 \ t1 \ ce1 \ cc1 \ n2 \ t2 \ ce2 \ cc2</math></li> <li>• <math>lub_{sqlcol}'</math></li> <li style="padding-left: 2em;"><math>(MkSqlCol \ n1 \ t1 \ ce1 \ cc1,</math></li> <li style="padding-left: 2em;"><math>MkSqlCol \ n2 \ t2 \ ce2 \ cc2)</math></li> <li><math>= MkSqlCol</math></li> <li style="padding-left: 2em;"><math>(lub_{sqlname} (n1, n2))</math></li> <li style="padding-left: 2em;"><math>(lub_{coltype} (t1, t2))</math></li> <li style="padding-left: 2em;"><math>(ce1 \ lub \ ce2)</math></li> <li style="padding-left: 2em;"><math>(lub_{boundinfo} (cc1, cc2)))</math></li> <li><math>lub_{sqlcol}</math></li> </ul>
<b>lub<sub>tableinfo</sub></b>	$\vdash ConstSpec$ <ul style="list-style-type: none"> <li>(<math>\lambda \ lub_{tableinfo}'</math></li> </ul>

---

---

- $\forall tec1\ tc1\ rc1\ tec2\ tc2\ rc2$ 
  - $lub_{tableinfo}'$   
 $(MkTableInfo\ tec1\ tc1\ rc1,$   
 $\quad MkTableInfo\ tec2\ tc2\ rc2)$   
 $= MkTableInfo$   
 $(tec1\ lub\ tec2)$   
 $(tc1\ lub\ tc2)$   
 $(lub_{boundinfo}\ (rc1,\ rc2)))$

**lub<sub>tsqlclassname</sub>**

$\vdash \forall tcn1\ tcn2$

- $lub_{tsqlclassname}\ (tcn1,\ tcn2)$   
 $= CASE$   
 $\quad tcn1$   
 $\quad [WHEN\_name_{tc}\ s1$   
  - $CASE$   
 $\quad tcn2$   
 $\quad [WHEN\_name_{tc}\ s2$   
    - $if\ s1 = s2$   
 $\quad then\ tcn1$   
 $\quad else\ c\_anon_{tc};$   
 $\quad OTHERS\ c\_anon_{tc}];$ $\quad OTHERS\ c\_anon_{tc}]$

**lub<sub>tsqlname</sub>**

$\vdash \forall tn1\ tn2$

- $lub_{tsqlname}\ (tn1,\ tn2)$   
 $= CASE$   
 $\quad tn1$   
 $\quad [WHEN\_name_t\ s1$   
  - $CASE$   
 $\quad tn2$   
 $\quad [WHEN\_name_t\ s2$   
    - $if\ s1 = s2$   
 $\quad then\ tn1$   
 $\quad else\ c\_anon_t; OTHERS\ c\_anon_t];$ $\quad WHEN\_none_t$   
 $\quad (CASE$   
 $\quad \quad tn2$   
 $\quad \quad [WHEN\_none_t\ c\_none_t;$   
 $\quad \quad \quad OTHERS\ c\_anon_t]);$   
 $\quad OTHERS\ c\_anon_t]$

**lub<sub>tsqlcol</sub>**

$\vdash ConstSpec$

$(\lambda\ lub_{tsqlcol}'$   

- $\forall s1\ d1\ c1\ s2\ d2\ c2$
- $lub_{tsqlcol}'$   
 $(MkTsqlCol\ s1\ d1\ c1,\ MkTsqlCol\ s2\ d2\ c2)$   
 $= MkTsqlCol$

---

---

```

      (lubtsqlname (s1, s2))
      (lubtsqlname (d1, d2))
      (lubtsqlclassname (c1, c2)))
lubtsqlcol
makesv ⊢ ∀ f t
  • makesv (f, t)
    = (let data_col (ftn, ttn)
      = CASE
        ftn
        [WHENnone_t
          (CASE
            ttn
            [WHENnone_t (Ok []);
              WHENanon_t
                (Ok [denote_null]);
              WHENname_t ts
                • Exception
                  [internalError]);
            WHENanon_t
              (CASE
                ttn
                [WHENnone_t
                  (Ok [denote_null]);
                  OTHERS
                    (Exception
                      [internalError])]);
            WHENname_t fs
              • CASE
                ttn
                [WHENanon_t
                  (Ok
                    [contents
                      (denote_col_spec
                        [fs])]);
                  WHENname_t ts
                    • Ok
                      [contents
                        (denote_col_spec
                          [fs])];
                  WHENnone_t (Exception [internalError])])
            in let class_col (fcn, tcn)
              = CASE
                fcn
                [WHENconstanttc fc
                  • CASE
                    tcn
                    [WHENconstanttc tc
                      • if fc = tc
                        then Ok []

```

---

```

        else
          Exception
            [internalError];
        WHEN_nametc tn
        • Ok [denote_class fc];
        WHEN_anontc
          (Exception
            [internalError]);
        WHEN_nametc f
        • CASE
          tcn
          [WHEN_constanttc tc
            • Exception
              [internalError];
            WHEN_nametc tn
            • Ok
              [contents
                (denote_col_spec
[f])]);
          WHEN_anontc
            (Ok
              [contents
(denote_col_spec [f])]);
          WHEN_anontc
            (Exception [internalError])]
    in let ssv
      = data_col
        (tc_sterling_name f,
         tc_sterling_name t)
    in let dsv
      = data_col
        (tc_dinary_name f,
         tc_dinary_name t)
    in let csv
      = class_col
        (tc_class_name f,
         tc_class_name t)
    in if
      isError ssv
      ∨ isError dsv
      ∨ isError csv
    then Exception [internalError]
    else
      Ok
        (destVal ssv
         @ destVal dsv
         @ destVal csv))

```

**remove\_constants**

---

$\vdash \forall x s$   
 •  $remove\_constants [] = []$   
 $\wedge remove\_constants (Cons x s)$   
 $= (if\ IsL\ x$   
 $\quad then\ Cons\ (OutL\ x)\ (remove\_constants\ s)$   
 $\quad else\ remove\_constants\ s)$

**remove\_nulls**  $\vdash \forall x trs$   
 •  $remove\_nulls [] = []$   
 $\wedge remove\_nulls (Cons x trs)$   
 $= CASE$   
 $\quad x$   
 $\quad [WHEN\_constant\_null\ (remove\_nulls\ trs);$   
 $\quad \quad OTHERS\ (Cons\ x\ (remove\_nulls\ trs))]$

**upper**  $\vdash \forall ec$   
 •  $upper\ ec$   
 $= CASE$   
 $\quad ec$   
 $\quad [WHEN\_variable\ (c,\ u)\bullet\ u;$   
 $\quad \quad WHEN\_constant_{ec}\ u\bullet\ u]$

**make\_case**  $\vdash \forall data\ ec$   
 •  $make\_case\ (data,\ ec)$   
 $= CASE$   
 $\quad ec$   
 $\quad [WHEN\_variable\ (c,\ u)$   
 $\quad \quad \bullet\ case\ [data]\ [denote\_class\ lattice\_top]\ c;$   
 $\quad \quad WHEN\_constant_{ec}\ c$   
 $\quad \quad \bullet\ case$   
 $\quad \quad \quad [data]$   
 $\quad \quad \quad [denote\_class\ lattice\_top]$   
 $\quad \quad \quad (denote\_class\ c)]$

**simplify\_and**  $\vdash \forall vs\ cs$   
 •  $simplify\_ands\ (vs,\ cs)$   
 $= (let\ v = fold\ (binop\ And)\ vs$   
 $\quad in\ let\ c$   
 $\quad = case$   
 $\quad \quad [v]$   
 $\quad \quad [fold$   
 $\quad \quad \quad (binop\ "lub")$   
 $\quad \quad \quad (Map\ denote\_class\_exp\ cs)]$   
 $\quad \quad (fold$   
 $\quad \quad \quad (binop\ "glb")$   
 $\quad \quad \quad (at2\ (Map\ make\_case)\ (vs,\ cs)))$   
 $\quad in\ let\ u = fold\ (Uncurry\ $lub)\ (Map\ upper\ cs)$   
 $\quad in\ (v,\ mk\_variable\ (c,\ u))$

**simplify\_or**  $\vdash \forall vs\ cs$   
 •  $simplify\_ors\ (vs,\ cs)$

---

---

```

= (let v = fold (binop Or) vs
   in let c
       = case
         [v]
         [fold
          (binop "glb")
          (Map denoteclassexp cs)]
         (fold
          (binop "lub")
          (at2 (Map make_case) (vs, cs)))
   in let u = fold (Uncurry $lub) (Map upper cs)
       in (v, mk_variable (c, u)))

constant_valuedata
  ⊢ true

dinary_columns
  ⊢ ∀ st css
    • dinary_columns st css
      = (let look cs
          = lookupcolspecdinary
            st
            (false, convertcolspec cs)
          in Try
            (ListOk o Map repr_col o remove_nulls)
            (ListOk (Map look css)))

sterling_columns
  ⊢ ∀ st css
    • sterling_columns st css
      = (let look cs
          = lookupcolspecsterling
            st
            (false, convertcolspec cs)
          in Try
            (ListOk o Map repr_col o remove_nulls)
            (ListOk (Map look css)))

tuple_listmaxrowclass
  ⊢ ∀ st t
    • tuple_listmaxrowclass st t
      = mk_constantec (client_clearance st)

upb_row_class
  ⊢ ConstSpec
    (λ upb_row_class'
     • ∀ tec tc rowc
       • upb_row_class' (MkTableInfo tec tc rowc)
         = CASE
           rowc
           [WHEN_upb rc • rc; WHEN_constant rc • rc])
    upb_row_class

valuetype

```

---

**value<sub>data</sub>** $\vdash true$ **value<sub>class</sub>** $\vdash \forall st\ ivc\ v$ 

- $value_{class}\ ivc\ st\ v$

- =  $(let\ x = ivc\ st\ v$

- in  $if\ isError\ x$

- then  $giveError\ (destError\ x)$

- else

- Ok

- (CASE

- ( $destVal\ x$ )

- [WHEN\_ands ( $datas, classes$ )

- $(let\ (v, c)$

- =  $simplify_{ands}$

- ( $datas, classes$ ) in  $c$ );

- WHEN\_ors ( $datas, classes$ )

- $(let\ (v, c)$

- =  $simplify_{ors}$

- ( $datas, classes$ ) in  $c$ );

- WHEN\_simple  $ec$

- CASE

- $ec$

- [WHEN\_variable ( $exp, up$ )

- $mk\_variable\ (exp, up)$ ;

- WHEN\_constant<sub>ec</sub>  $c$

- $mk\_constant_{ec}\ c$ ]]))

**query<sub>selectquery</sub>****transform<sub>selectquery</sub>****tuple\_list<sub>outerinfo</sub>****tuple\_list<sub>makeouter</sub>****select\_value<sub>data</sub>****select\_list<sub>data</sub>****make<sub>sterling</sub>****make<sub>dinary</sub>****make\_col****select\_value<sub>make</sub>****select\_list<sub>make</sub>****select\_value<sub>class</sub>****select\_list<sub>class</sub>****tuple\_list<sub>class</sub>****internal\_value<sub>class</sub>****value<sub>info</sub>****select\_value<sub>info</sub>****select\_list<sub>info</sub>****from\_spec<sub>info</sub>****tuple\_list<sub>make</sub>****tuple\_list<sub>info</sub>**

**select\_value**<sub>type</sub>  
**select\_list**<sub>type</sub>  
**from\_spec**<sub>enter</sub>  
**tuple\_list**<sub>type</sub>  
**tuple\_list**<sub>data</sub>

$\vdash true$   
**STP**  $\vdash \forall q c$ 

- *STP* ( $q, c$ )
  - = (let  $st = init\_trans\_state\ c$
  - in let  $res$ 
    - =  $transform_{selectquery}\ st\ q$
    - in if  $isError\ res$
    - then  $giveError\ (destError\ res)$
    - else
    - Ok*
      - (let ( $scs, tq, scw, rc, tcs, chks$ )
      - =  $destVal\ res$
      - in let  $dynamic\ tc$
      - $\Leftrightarrow CASE$
      - $tc$
      - [ $WHEN\_constant_{tc}\ c \bullet false;$
      - $OTHERS\ true$ ]
      - in let  $cc$
      - =  $Map$
      - ( $dynamic\ o\ tc\_class\_name$ )
      - $tcs$
      - in let  $cr$
      - = (if  $dynamic\ rc$
      - then  $InL\ c$
      - else  $InR\ maybe$ )
      - in let  $cq$
      - = (if  $chks = []$
      - then  $InR\ maybe$
      - else  $InL\ (Head\ chks)$ )
      - in ( $tq, cq, scw, cr, cc$ ))

## 27 THE THEORY fef031

### 27.1 Parents

*fef034 fef026*

### 27.2 Children

*fef033*

### 27.3 Constants

**View<sub>t</sub>\_secureE***Bool***outputFilter\_secureE***Bool*

### 27.4 Definitions

**View<sub>t</sub>\_secureE**

$$\begin{aligned} &\vdash \text{View}_t\text{-secureE} \\ &\Leftrightarrow (\forall c\ s1\ s2) \\ &\quad \bullet \text{hide}(c, s1) = \text{hide}(c, s2) \\ &\quad \Rightarrow \text{Map}(\text{HideDerTable } c) (\text{View}_t(\text{reprState } s1)) \\ &\quad = \text{Map} \\ &\quad \quad (\text{HideDerTable } c) \\ &\quad \quad (\text{View}_t(\text{reprState } s2)) \end{aligned}$$
**outputFilter\_secureE**

$$\begin{aligned} &\vdash \text{outputFilter\_secureE} \\ &\Leftrightarrow (\forall q\ c\ t1\ t2\ dq\ ocq\ fps) \\ &\quad \bullet \neg \text{isError}(\text{STP}(q, c)) \\ &\quad \quad \wedge \text{destVal}(\text{STP}(q, c)) = (dq, ocq, fps) \\ &\quad \quad \wedge \text{HideDerTable } c\ t1 = \text{HideDerTable } c\ t2 \\ &\quad \Rightarrow \text{outputFilter}(c, (\text{GiveData } t1, []), fps) \\ &\quad = \text{outputFilter}(c, (\text{GiveData } t2, []), fps) \end{aligned}$$

### 27.5 Theorems

**map\_o\_lemma**  $\vdash \forall f\ g\ list \bullet \text{Map } f (\text{Map } g\ list) = \text{Map} (f\ o\ g)\ list$ **lub\_lemma**  $\vdash \forall c \bullet c\ \text{lub } c = c \wedge \text{lubl } [c] = c$ **dominates\_lub\_lemma**

$$\begin{aligned} &\vdash \forall c1\ c2\ c3 \\ &\quad \bullet c1\ \text{dominates } c2\ \text{lub } c3 \\ &\quad \Leftrightarrow c1\ \text{dominates } c2 \wedge c1\ \text{dominates } c3 \end{aligned}$$
**MkDerTable\_lemma**

$$\begin{aligned} &\vdash \forall x1\ x2\ y1\ y2 \\ &\quad \bullet \text{MkDerTable } x1\ x2 = \text{MkDerTable } y1\ y2 \\ &\quad \Leftrightarrow x1 = y1 \wedge x2 = y2 \end{aligned}$$
**MkDerTableRow\_lemma**

$$\begin{aligned} &\vdash \forall x1\ x2\ x3\ y1\ y2\ y3 \\ &\quad \bullet \text{MkDerTableRow } x1\ x2\ x3 = \text{MkDerTableRow } y1\ y2\ y3 \\ &\quad \Leftrightarrow x1 = y1 \wedge x2 = y2 \wedge x3 = y3 \end{aligned}$$
**Act<sub>t</sub>\_lemma**

$$\begin{aligned} &\vdash \forall \text{upd } qdes \\ &\quad \bullet \text{Act}_t\ \text{upd } qdes \\ &\quad = (\text{let } (query, (dt, errs), st) = qdes \\ &\quad \quad \text{in if } \neg errs = [] \\ &\quad \quad \text{then } (st, [], errs) \\ &\quad \quad \text{else if } \text{is\_select } query \end{aligned}$$

then (*st*, *GiveData dt*, *errs*)  
else (*upd (query, (dt, errs), st)*, [], [])

**giveVal\_consistent**  
**giveError\_consistent**  
**destVal\_consistent**  
**destError\_consistent**  
**isVal\_consistent**  
**isError\_consistent**

⊢ *Consistent*  
(λ  
  (*giveVal'*, *giveError'*, *destVal'*, *destError'*,  
   *isVal'*, *isError'*)  
  • ∀ *v e ve*  
    • *giveVal' v = InL v*  
      ∧ *giveError' e = InR e*  
      ∧ *destVal' (giveVal' v) = v*  
      ∧ *destError' (giveError' e) = e*  
      ∧ (*isVal' ve* ⇔ (∃ *v<sub>1</sub>* • *ve = giveVal' v<sub>1</sub>*))  
      ∧ (*isError' ve*  
        ⇔ (∃ *e<sub>1</sub>* • *ve = giveError' e<sub>1</sub>*)))

**EM\_SecureE\_Lemma1**

∀ *compile upd*  
  • *STP ∈ STP\_secure\_E compile*  
  ⇒ (*EM<sub>1</sub> compile upd, STP, outputFilter*)  
  ∈ *subsys\_secureE reprState*  
  ⊢ *Correct\_Compile\_STP\_secure\_E* ⇒ *Subsys\_SecureE*

**EM\_SecureE\_Lemma2**

*View<sub>t</sub>\_secureE*,  
*outputFilter\_secureE*,  
  ∀ *t* • *GiveData t = []* ⇔ *DT\_rows t = []*  
  ⊢ ∀ *compile upd*  
    • *STP ∈ STP\_secure\_E compile*  
    ⇒ (*EM<sub>1</sub> compile upd, STP, outputFilter*)  
    ∈ *subsys\_secureE reprState*

**EM\_SecureE\_Lemma3**

⊢ ∀ *t* • *GiveData t = []* ⇔ *DT\_rows t = []*

**EM\_SecureE\_thm**

*View<sub>t</sub>\_secureE, outputFilter\_secureE* ⊢ *EM\_SecureE*

## 28 THE THEORY fef032

### 28.1 Parents

*fef026*

### 28.2 Children

*fef033 fef034*

---

### 28.3 Constants

**DenoteConstant**

	$Class \times ValuedItem \ OPT \rightarrow VALUE\_COMP$
<b>MonOp</b>	$(ValuedItem \ OPT \rightarrow ValuedItem \ OPT)$ $\rightarrow VALUE\_COMP$ $\rightarrow VALUE\_COMP$
<b>ItemBool</b>	$ValuedItem \ OPT \rightarrow Bool$
<b>BoolItem</b>	$Bool \rightarrow ValuedItem \ OPT$
<b>ListAnd</b>	$Bool \ LIST \rightarrow Bool$
<b>ListOr</b>	$Bool \ LIST \rightarrow Bool$
<b>ComputeAnd</b>	$Class$ $\rightarrow (Class \times ValuedItem \ OPT) \ LIST$ $\rightarrow Class \times ValuedItem \ OPT$
<b>BinOpAnd</b>	$Class \rightarrow VALUE\_COMP \ LIST \rightarrow VALUE\_COMP$
<b>ComputeOr</b>	$Class$ $\rightarrow (Class \times ValuedItem \ OPT) \ LIST$ $\rightarrow Class \times ValuedItem \ OPT$
<b>BinOpOr</b>	$Class \rightarrow VALUE\_COMP \ LIST \rightarrow VALUE\_COMP$
<b>BinOp</b>	$(ValuedItem \ OPT \rightarrow ValuedItem \ OPT \rightarrow ValuedItem \ OPT)$ $\rightarrow VALUE\_COMP$ $\rightarrow VALUE\_COMP$ $\rightarrow VALUE\_COMP$
<b>TriOp</b>	$(ValuedItem \ OPT$ $\rightarrow ValuedItem \ OPT$ $\rightarrow ValuedItem \ OPT$ $\rightarrow ValuedItem \ OPT)$ $\rightarrow VALUE\_COMP$ $\rightarrow VALUE\_COMP$ $\rightarrow VALUE\_COMP$ $\rightarrow VALUE\_COMP$
<b>CheckList</b>	$Class$ $\rightarrow ValuedItem \ OPT$ $\rightarrow ((Class \times ValuedItem \ OPT) \times Class \times ValuedItem \ OPT)$ $LIST$ $\rightarrow Class$ $\rightarrow Class$
<b>CheckTest</b>	$Class$ $\rightarrow Class \times ValuedItem \ OPT$ $\rightarrow ((Class \times ValuedItem \ OPT) \times Class \times ValuedItem \ OPT)$ $LIST$ $\rightarrow Class$ $\rightarrow Class$
<b>CaseValValue</b>	$ValuedItem \ OPT$ $\rightarrow ((Class \times ValuedItem \ OPT) \times Class \times ValuedItem \ OPT)$ $LIST$ $\rightarrow ValuedItem \ OPT$ $\rightarrow ValuedItem \ OPT$

---

<b>CaseVal</b>	<i>Class</i> → <i>VALUE_COMP</i> → ( <i>VALUE_COMP</i> × <i>VALUE_COMP</i> ) <i>LIST</i> → <i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>CaseC</b>	<i>Class</i> → (( <i>Class</i> × <i>ValuedItem OPT</i> ) × <i>Class</i> × <i>ValuedItem OPT</i> ) <i>LIST</i> → <i>Class</i> → <i>Class</i>
<b>CaseValue</b>	(( <i>Class</i> × <i>ValuedItem OPT</i> ) × <i>Class</i> × <i>ValuedItem OPT</i> ) <i>LIST</i> → <i>ValuedItem OPT</i> → <i>ValuedItem OPT</i>
<b>Case</b>	<i>Class</i> → ( <i>VALUE_COMP</i> × <i>VALUE_COMP</i> ) <i>LIST</i> → <i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>SetFuncAllAnd</b>	<i>Class</i> → <i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>SetFuncAllOr</b>	<i>Class</i> → <i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>SetFuncAll</b>	( <i>ValuedItem OPT LIST</i> → <i>ValuedItem OPT</i> ) → <i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>SetFuncDistinct</b>	( <i>ValuedItem OPT</i> $\mathbb{P}$ → <i>ValuedItem OPT</i> ) → <i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>NatItem</b>	<i>Worth</i> → <i>ValuedItem OPT</i>
<b>CountNonNull</b>	<i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>CountDistinct</b>	<i>VALUE_COMP</i> → <i>VALUE_COMP</i>
<b>CountAll</b>	<i>VALUE_COMP</i>
<b>ExistsTuples</b>	<i>Class</i> → <i>TABLE_COMP</i> → <i>VALUE_COMP</i>
<b>SingleValue</b>	<i>Class</i> → <i>TABLE_COMP</i> → <i>VALUE_COMP</i>
<b>Contents</b>	<i>Worth</i> → <i>VALUE_COMP</i>
<b>ClassItem</b>	<i>Class</i> → <i>ValuedItem OPT</i>
<b>Classification</b>	<i>Worth</i> → <i>VALUE_COMP</i>
<b>JoinedRowExistence</b>	<i>Class</i> → <i>VALUE_COMP</i>
<b>JoinSpecs</b>	<i>DerTableSpec LIST</i> → <i>DerTableSpec</i>
<b>JoinRows</b>	<i>DerTableRow</i> → <i>DerTableRow LIST</i> → <i>DerTableRow LIST</i>
<b>JoinData</b>	<i>DerTableRow LIST LIST</i> → <i>DerTableRow LIST</i>
<b>Join</b>	<i>DerTable LIST</i> → <i>DerTableSpec</i> × <i>DerTableRow LIST</i>
<b>ProjectSpec</b>	<i>DerColSpec LIST</i> → <i>DerTableSpec</i> → <i>DerTableSpec</i>
<b>ProjectData</b>	<i>DerTable LIST</i> → <i>VALUE_COMP LIST</i>

---

---

	→ <i>DerTableRow LIST LIST</i>
	→ <i>DerTableRow LIST</i>
<b>Project</b>	<i>DerTableSpec</i>
	→ <i>DerTable LIST</i>
	→ <i>(VALUE_COMP × DerColSpec) LIST</i>
	→ <i>DerTableRow LIST LIST</i>
	→ <i>DerTable</i>
<b>EvalProjectData</b>	<i>DerTable LIST</i>
	→ <i>VALUE_COMP LIST</i>
	→ <i>DerTableRow LIST LIST</i>
	→ <i>DerTableRow LIST</i>
<b>EvalProject</b>	<i>DerTableSpec</i>
	→ <i>DerTable LIST</i>
	→ <i>(VALUE_COMP × DerColSpec) LIST</i>
	→ <i>DerTableRow LIST LIST</i>
	→ <i>DerTable</i>
<b>Where</b>	<i>Class</i>
	→ <i>DerTable LIST</i>
	→ <i>DerTableRow LIST</i>
	→ <i>VALUE_COMP</i>
	→ <i>DerTableRow LIST</i>
<b>PutInGroup</b>	<i>('a → 'b) → 'a → 'a LIST LIST → 'a LIST LIST</i>
<b>MakeGroups</b>	<i>('a → 'b) → 'a LIST → 'a LIST LIST</i>
<b>ListNth</b>	<i>Errors → 'a LIST → 'a LIST</i>
<b>CommonValue</b>	<i>VALUE_COMP → VALUE_COMP</i>
<b>Group</b>	<i>Class</i>
	→ <i>DerTable LIST</i>
	→ <i>DerTableRow LIST</i>
	→ <i>Errors</i>
	→ <i>Errors</i>
	→ <i>VALUE_COMP</i>
	→ <i>Class × DerTableRow LIST LIST</i>
<b>TableContents</b>	<i>Worth → TABLE_COMP</i>
<b>AllTuples</b>	<i>Class</i>
	→ <i>(VALUE_COMP × DerColSpec) LIST</i>
	→ <i>TABLE_COMP LIST</i>
	→ <i>VALUE_COMP</i>
	→ <i>Errors</i>
	→ <i>Errors</i>
	→ <i>VALUE_COMP</i>
	→ <i>TABLE_COMP</i>
<b>RemoveDuplicates</b>	<i>'a LIST → 'a LIST</i>
<b>DistinctTuples</b>	<i>Class</i>
	→ <i>(VALUE_COMP × DerColSpec) LIST</i>

---

---

	→ <i>TABLE_COMP LIST</i>
	→ <i>VALUE_COMP</i>
	→ <i>Errors</i>
	→ <i>Errors</i>
	→ <i>VALUE_COMP</i>
	→ <i>TABLE_COMP</i>
<b>Evaluate</b>	<i>Class</i>
	→ ( <i>VALUE_COMP</i> × <i>DerColSpec</i> ) <i>LIST</i>
	→ <i>TABLE_COMP LIST</i>
	→ <i>VALUE_COMP</i>
	→ <i>Errors</i>
	→ <i>Errors</i>
	→ <i>VALUE_COMP</i>
	→ <i>TABLE_COMP</i>
$\cap_2$	'a $\mathbb{P}$ ↔ 'b $\mathbb{P}$ → 'a $\mathbb{P}$ × 'b $\mathbb{P}$
<b>ValueComputations</b>	<i>Class</i> → <i>VALUE_COMP</i> $\mathbb{P}$
<b>TableComputations</b>	<i>Class</i> → <i>TABLE_COMP</i> $\mathbb{P}$
<b>OkTableComputation</b>	<i>Class</i> → <i>TABLE_COMP</i> $\mathbb{P}$
<b>OkSTP</b>	( <i>Query</i> → <i>DerTable LIST</i> → <i>DerTable</i> × <i>Errors</i> ) → ( <i>Query</i> , 'PARS) <i>STP_TYPE</i> $\mathbb{P}$
<b>OK_TC<sub>d</sub></b>	<i>Class</i> → <i>TABLE_COMP</i> $\mathbb{P}$
<b>OK_VC<sub>d</sub></b>	<i>Class</i> → <i>VALUE_COMP</i> $\mathbb{P}$
<b>OK_VC<sub>c</sub></b>	<i>Class</i> → <i>VALUE_COMP</i> $\mathbb{P}$
<b>OK_TC<sub>c</sub></b>	<i>Class</i> → <i>TABLE_COMP</i> $\mathbb{P}$

## 28.4 Type Abbreviations

<b>TABLE_COMP</b>	<i>TABLE_COMP</i>
<b>VALUE_COMP</b>	<i>VALUE_COMP</i>

## 28.5 Definitions

### DenoteConstant

⊢ ∀ *ci* • *DenoteConstant ci* = (λ *tl rl r* • *ci*)

### MonOp

⊢ ∀ *f e*  
 • *MonOp f e*  
 = (λ *tl rl r*  
 • (let (*c, v*) = *e tl rl r* in (*c, f v*)))

### ItemBool

⊢ ∀ *v*  
 • *ItemBool v*  
 ⇔ *v*  
 = *ValuedItemItem*  
 (*MkValuedItem sterling (BoolVal true)*)

### BoolItem

⊢ ∀ *v*  
 • *BoolItem v*

---

	$= \text{ValuedItemItem}$ $(\text{MkValuedItem } \text{sterling } (\text{BoolVal } v))$
<b>ListAnd</b>	$\vdash \forall b \text{ bs}$ <ul style="list-style-type: none"> <li>• <math>(\text{ListAnd } [] \Leftrightarrow \text{true})</math></li> <li><math>\wedge (\text{ListAnd } (\text{Cons } b \text{ bs}) \Leftrightarrow b \wedge \text{ListAnd } \text{bs})</math></li> </ul>
<b>ListOr</b>	$\vdash \forall b \text{ bs}$ <ul style="list-style-type: none"> <li>• <math>(\text{ListOr } [] \Leftrightarrow \text{false})</math></li> <li><math>\wedge (\text{ListOr } (\text{Cons } b \text{ bs}) \Leftrightarrow b \vee \text{ListOr } \text{bs})</math></li> </ul>
<b>ComputeAnd</b>	$\vdash \forall cc \text{ cil}$ <ul style="list-style-type: none"> <li>• <math>\text{ComputeAnd } cc \text{ cil}</math></li> <li><math>= (\text{let } hcil</math></li> <li><math>\quad = cil</math></li> <li><math>\quad \uparrow \{(c, i)   cc \text{ dominates } c \wedge \neg \text{ItemBool } i\}</math></li> <li><math>\text{in let } v \Leftrightarrow \text{ListAnd } (\text{Map } (\text{ItemBool } o \text{ Snd}) \text{ cil})</math></li> <li><math>\text{in let } \text{makecase } (c, u)</math></li> <li><math>\quad = (\text{if } \text{ItemBool } u</math></li> <li><math>\quad \quad \text{then } \text{lattice\_bottom}</math></li> <li><math>\quad \quad \text{else } c)</math></li> <li><math>\text{in if } hcil = []</math></li> <li><math>\text{then } (\text{lubl } (\text{Map } \text{Fst } cil), \text{BoolItem } v)</math></li> <li><math>\text{else}</math></li> <li><math>\quad (\text{lubl } (\text{Map } \text{makecase } hcil),</math></li> <li><math>\quad \quad \text{BoolItem } \text{false}))</math></li> </ul>
<b>BinOpAnd</b>	$\vdash \forall cc \text{ el}$ <ul style="list-style-type: none"> <li>• <math>\text{BinOpAnd } cc \text{ el}</math></li> <li><math>= (\lambda \text{ tl } rl \text{ r}</math></li> <li><math>\quad \bullet \text{ComputeAnd } cc (\text{Map } (\lambda e \bullet e \text{ tl } rl \text{ r}) \text{ el}))</math></li> </ul>
<b>ComputeOr</b>	$\vdash \forall cc \text{ cil}$ <ul style="list-style-type: none"> <li>• <math>\text{ComputeOr } cc \text{ cil}</math></li> <li><math>= (\text{let } hcil</math></li> <li><math>\quad = cil</math></li> <li><math>\quad \uparrow \{(c, i)   cc \text{ dominates } c \wedge \text{ItemBool } i\}</math></li> <li><math>\text{in let } v \Leftrightarrow \text{ListOr } (\text{Map } (\text{ItemBool } o \text{ Snd}) \text{ cil})</math></li> <li><math>\text{in let } \text{makecase } (c, u)</math></li> <li><math>\quad = (\text{if } \neg \text{ItemBool } u</math></li> <li><math>\quad \quad \text{then } \text{lattice\_bottom}</math></li> <li><math>\quad \quad \text{else } c)</math></li> <li><math>\text{in if } hcil = []</math></li> <li><math>\text{then } (\text{lubl } (\text{Map } \text{Fst } cil), \text{BoolItem } v)</math></li> <li><math>\text{else}</math></li> <li><math>\quad (\text{lubl } (\text{Map } \text{makecase } hcil),</math></li> <li><math>\quad \quad \text{BoolItem } \text{true}))</math></li> </ul>
<b>BinOpOr</b>	$\vdash \forall cc \text{ el}$ <ul style="list-style-type: none"> <li>• <math>\text{BinOpOr } cc \text{ el}</math></li> <li><math>= (\lambda \text{ tl } rl \text{ r}</math></li> <li><math>\quad \bullet \text{ComputeOr } cc (\text{Map } (\lambda e \bullet e \text{ tl } rl \text{ r}) \text{ el}))</math></li> </ul>
<b>BinOp</b>	$\vdash \forall f \text{ e1 } \text{ e2}$ <ul style="list-style-type: none"> <li>• <math>\text{BinOp } f \text{ e1 } \text{ e2}</math></li> </ul>

---

---

	$= (\lambda tl rl r$ <ul style="list-style-type: none"> <li>• <math>(let (c1, v1) = e1 tl rl r</math> <ul style="list-style-type: none"> <li>in <math>let (c2, v2) = e2 tl rl r</math> <ul style="list-style-type: none"> <li>in <math>(c1 lub c2, f v1 v2)))</math></li> </ul> </li> </ul> </li> </ul>
<b>TriOp</b>	$\vdash \forall f e1 e2 e3$ <ul style="list-style-type: none"> <li>• <math>TriOp f e1 e2 e3</math> <ul style="list-style-type: none"> <li><math>= (\lambda tl rl r</math> <ul style="list-style-type: none"> <li>• <math>(let (c1, v1) = e1 tl rl r</math> <ul style="list-style-type: none"> <li>in <math>let (c2, v2) = e2 tl rl r</math> <ul style="list-style-type: none"> <li>in <math>let (c3, v3) = e3 tl rl r</math> <ul style="list-style-type: none"> <li>in <math>(c1 lub c2 lub c3, f v1 v2 v3)))</math></li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul> </li></ul>
<b>CheckList</b>	$\vdash \forall cc ti cv cvs elsec$ <ul style="list-style-type: none"> <li>• <math>CheckList cc ti [] elsec = elsec</math> <ul style="list-style-type: none"> <li><math>\wedge CheckList cc ti (Cons cv cvs) elsec</math> <ul style="list-style-type: none"> <li><math>= (let ((cec, cei), vec, vei) = cv</math> <ul style="list-style-type: none"> <li>in <math>if \neg cc \text{ dominates } cec</math> <ul style="list-style-type: none"> <li>then <math>cec</math></li> <li>else <math>if ti = cei</math> <ul style="list-style-type: none"> <li>then <math>vec</math></li> <li>else <math>CheckList cc ti cvs elsec)</math></li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul> </li></ul>
<b>CheckTest</b>	$\vdash \forall cc ti tc cvs elsec$ <ul style="list-style-type: none"> <li>• <math>CheckTest cc (tc, ti) cvs elsec</math> <ul style="list-style-type: none"> <li><math>= (if cc \text{ dominates } tc</math> <ul style="list-style-type: none"> <li>then <math>CheckList cc ti cvs elsec</math></li> <li>else <math>tc)</math></li> </ul> </li> </ul> </li></ul>
<b>CaseValValue</b>	$\vdash \forall ti cv cvs elsev$ <ul style="list-style-type: none"> <li>• <math>CaseValValue ti [] elsev = elsev</math> <ul style="list-style-type: none"> <li><math>\wedge CaseValValue ti (Cons cv cvs) elsev</math> <ul style="list-style-type: none"> <li><math>= (let ((cec, cei), vec, vei) = cv</math> <ul style="list-style-type: none"> <li>in <math>if ti = cei</math> <ul style="list-style-type: none"> <li>then <math>vei</math></li> <li>else <math>CaseValValue ti cvs elsev)</math></li> </ul> </li> </ul> </li> </ul> </li> </ul> </li></ul>
<b>CaseVal</b>	$\vdash \forall cc tst casevals elseval$ <ul style="list-style-type: none"> <li>• <math>CaseVal cc tst casevals elseval</math> <ul style="list-style-type: none"> <li><math>= (\lambda tl rl r</math> <ul style="list-style-type: none"> <li>• <math>(let (tc, ti) = tst tl rl r</math> <ul style="list-style-type: none"> <li>in <math>let cvs</math> <ul style="list-style-type: none"> <li><math>= Map</math> <ul style="list-style-type: none"> <li><math>(\lambda (c, v) \bullet (c tl rl r, v tl rl r))</math></li> <li><math>casevals</math></li> </ul> </li> <li>in <math>let (ec, ei) = elseval tl rl r</math> <ul style="list-style-type: none"> <li>in <math>let c = CheckTest cc (tc, ti) cvs ec</math> <ul style="list-style-type: none"> <li>in <math>let v = CaseValValue ti cvs ei</math> <ul style="list-style-type: none"> <li>in <math>(c, v)))</math></li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul> </li></ul></li></ul>
<b>CaseC</b>	$\vdash \forall cc cv cvs elsec$ <ul style="list-style-type: none"> <li>• <math>CaseC cc [] elsec = elsec</math> <ul style="list-style-type: none"> <li><math>\wedge CaseC cc (Cons cv cvs) elsec</math> <ul style="list-style-type: none"> <li><math>= (let ((cec, cei), vec, vei) = cv</math></li> </ul> </li> </ul> </li> </ul>

---

---

	<i>in if</i> $\neg$ <i>cc dominates cec</i> <i>then cec</i> <i>else if ItemBool cei</i> <i>then vec</i> <i>else CaseC cc cvs elsec)</i>
<b>CaseValue</b>	$\vdash \forall cv\ cvs\ elsev$ <ul style="list-style-type: none"> <li>• <i>CaseValue [] elsev = elsev</i>  <math>\wedge</math> <i>CaseValue (Cons cv cvs) elsev</i>  <math>=</math> (<i>let</i> ((<i>cec</i>, <i>cei</i>), <i>vec</i>, <i>vei</i>) <math>=</math> <i>cv</i>  <i>in if ItemBool cei</i>  <i>then vei</i>  <i>else CaseValue cvs elsev</i>)</li> </ul>
<b>Case</b>	$\vdash \forall cc\ casevals\ elseval$ <ul style="list-style-type: none"> <li>• <i>Case cc casevals elseval</i>  <math>=</math> (<math>\lambda</math> <i>tl rl r</i>  <ul style="list-style-type: none"> <li>• (<i>let cvs</i>  <math>=</math> <i>Map</i>  <math>(\lambda</math> (<i>c</i>, <i>v</i>) • (<i>c tl rl r</i>, <i>v tl rl r</i>))  <i>casevals</i>  <i>in let</i> (<i>ec</i>, <i>ei</i>) <math>=</math> <i>elseval tl rl r</i>  <i>in let</i> <i>c</i> <math>=</math> <i>CaseC cc cvs ec</i>  <i>in let</i> <i>v</i> <math>=</math> <i>CaseValue cvs ei in (c, v)</i>))</li> </ul> </li> </ul>
<b>SetFuncAllAnd</b>	$\vdash \forall cc\ e$ <ul style="list-style-type: none"> <li>• <i>SetFuncAllAnd cc e</i>  <math>=</math> (<math>\lambda</math> <i>tl rl r</i> • <i>ComputeAnd cc (Map (e tl rl) rl)</i>)</li> </ul>
<b>SetFuncAllOr</b>	$\vdash \forall cc\ e$ <ul style="list-style-type: none"> <li>• <i>SetFuncAllOr cc e</i>  <math>=</math> (<math>\lambda</math> <i>tl rl r</i> • <i>ComputeOr cc (Map (e tl rl) rl)</i>)</li> </ul>
<b>SetFuncAll</b>	$\vdash \forall f\ e$ <ul style="list-style-type: none"> <li>• <i>SetFuncAll f e</i>  <math>=</math> (<math>\lambda</math> <i>tl rl r</i>  <ul style="list-style-type: none"> <li>• (<i>let</i> (<i>cl</i>, <i>il</i>) <math>=</math> <i>Split (Map (e tl rl) rl)</i>  <i>in (lubl cl, f il)</i>))</li> </ul> </li> </ul>
<b>SetFuncDistinct</b>	$\vdash \forall f\ e$ <ul style="list-style-type: none"> <li>• <i>SetFuncDistinct f e</i>  <math>=</math> (<math>\lambda</math> <i>tl rl r</i>  <ul style="list-style-type: none"> <li>• (<i>let</i> (<i>cl</i>, <i>il</i>) <math>=</math> <i>Split (Map (e tl rl) rl)</i>  <i>in (lubl cl, f (Elems il))</i>))</li> </ul> </li> </ul>
<b>NatItem</b>	$\vdash true$
<b>CountNonNull</b>	$\vdash \forall e$ <ul style="list-style-type: none"> <li>• <i>CountNonNull e</i>  <math>=</math> (<i>let counter il</i>  <math>=</math> <i>NatItem (# (il <math>\uparrow</math> {i isValuedItem i}))</i>  <i>in SetFuncAll counter e</i>)</li> </ul>
<b>CountDistinct</b>	$\vdash \forall e$

---

---

	<ul style="list-style-type: none"> <li>• <i>CountDistinct e</i> = (let counter is = NatItem (# (is ∩ {i isValuedItem i})) in SetFuncDistinct counter e)</li> </ul>
<b>CountAll</b>	<ul style="list-style-type: none"> <li>⊢ <i>CountAll</i> = (λ tl rl r • (let cl = Map DTR_row rl in (lubl cl, NatItem (# rl))))</li> </ul>
<b>ExistsTuples</b>	<ul style="list-style-type: none"> <li>⊢ ∀ cc te           <ul style="list-style-type: none"> <li>• <i>ExistsTuples cc te</i> = (λ tl rl r • (let (c, t) = te tl in let trl = DT_rows t ↑ {r   cc dominates DTR_row r ∧ cc dominates DTR_where r} in if cc dominates c then (lubl (Map DTR_row trl), BoolItem (¬ trl = [])) else (c, Arbitrary))))</li> </ul> </li> </ul>
<b>SingleValue</b>	<ul style="list-style-type: none"> <li>⊢ ∀ cc te           <ul style="list-style-type: none"> <li>• <i>SingleValue cc te</i> = (λ tl rl r • (let (c, t) = te tl in let trl = DT_rows t ↑ {r   cc dominates DTR_row r ∧ cc dominates DTR_where r} in let cil = DTR_cols (Head trl) in let (ic, ii) = Head cil in if cc dominates c then if # trl = 1 ∧ # cil = 1 then (ic, ii) else (c, Arbitrary) else (c, Arbitrary))))</li> </ul> </li> </ul>
<b>Contents</b>	<ul style="list-style-type: none"> <li>⊢ ∀ i           <ul style="list-style-type: none"> <li>• <i>Contents i</i> = (λ tl rl r • if 1 ≤ i ∧ i ≤ # (DTR_cols r) then Nth (DTR_cols r) i else Arbitrary)</li> </ul> </li> </ul>
<b>ClassItem</b>	<ul style="list-style-type: none"> <li>⊢ ∀ v           <ul style="list-style-type: none"> <li>• <i>ClassItem v</i> = ValuedItemItem</li> </ul> </li> </ul>

---

---

(MkValuedItem sterling (ClassVal v))

**Classification**

⊢ ∀ i

- Classification i

= (λ tl rl r

- if 1 ≤ i ∧ i ≤ # (DTR\_cols r)

then

(DTR\_row r,

ClassItem (Fst (Nth (DTR\_cols r) i)))

else Arbitrary)

**JoinedRowExistence**

⊢ ∀ cc

- JoinedRowExistence cc

= (λ tl rl r • (cc, ClassItem (DTR\_row r)))

**JoinSpecs**

⊢ ∀ sl

- JoinSpecs sl

= (let n = []

and mr = lubl (Map DTS\_maxRow sl)

and csl = Flat (Map DTS\_colSpecs sl)

in MkDerTableSpec n mr csl)

**JoinRows**

⊢ ∀ r rs

- JoinRows r rs

= (let join2 rr

= MkDerTableRow

(DTR\_where r lub DTR\_where rr)

(DTR\_row r lub DTR\_row rr)

(DTR\_cols r @ DTR\_cols rr)

in Map join2 rs)

**JoinData**

⊢ JoinData [] = []

∧ (∀ tab rest

- JoinData (Cons tab rest)

= (if rest = []

then tab

else

(let jrest = JoinData rest

in let join\_blk r = JoinRows r jrest

in Flat (Map join\_blk tab))))

**Join**

⊢ ∀ tabl

- Join tabl

= (JoinSpecs (Map DT\_spec tabl),

JoinData (Map DT\_rows tabl))

**ProjectSpec**

⊢ ∀ sl s

- ProjectSpec sl s

= MkDerTableSpec [] (DTS\_maxRow s) sl

**ProjectData**

⊢ ∀ tl el gps

- ProjectData tl el gps

= (let h gp r

= MkDerTableRow

---

$(DTR\_where\ r)$   
 $(DTR\_row\ r)$   
 $(Map\ (\lambda\ e\bullet\ e\ tl\ gp\ r)\ el)$   
 $in\ let\ k\ gp = Map\ (h\ gp)\ gp$   
 $in\ Flat\ (Map\ k\ gps)$

**Project**      $\vdash\ \forall\ ts\ tl\ sellist\ gps$

- $Project\ ts\ tl\ sellist\ gps$   
 $=\ MkDerTable$   
 $(ProjectSpec\ (Map\ Snd\ sellist)\ ts)$   
 $(ProjectData\ tl\ (Map\ Fst\ sellist)\ gps)$

**EvalProjectData**

$\vdash\ \forall\ tl\ el\ gps$

- $EvalProjectData\ tl\ el\ gps$   
 $=\ (let\ h\ gp\ r$   
 $=\ MkDerTableRow$   
 $(DTR\_where\ r)$   
 $(DTR\_row\ r)$   
 $(Map\ (\lambda\ e\bullet\ e\ tl\ gp\ r)\ el)$   
 $in\ let\ k\ gp$   
 $=\ (let\ results = Map\ (h\ gp)\ gp$   
 $in\ if\ \#(Elems\ results) = 1$   
 $then\ Head\ results$   
 $else\ Arbitrary)\ in\ Map\ k\ gps)$

**EvalProject**      $\vdash\ \forall\ ts\ tl\ sellist\ gps$

- $EvalProject\ ts\ tl\ sellist\ gps$   
 $=\ MkDerTable$   
 $(ProjectSpec\ (Map\ Snd\ sellist)\ ts)$   
 $(EvalProjectData\ tl\ (Map\ Fst\ sellist)\ gps)$

**Where**

$\vdash\ \forall\ c\ tl\ rl\ e$

- $Where\ c\ tl\ rl\ e$   
 $=\ (let\ hrl = rl\ \upharpoonright\ \{r\mid c\ dominates\ DTR\_row\ r\}$   
 $in\ let\ w\ r = DTR\_where\ r\ lub\ Fst\ (e\ tl\ hrl\ r)$   
 $in\ let\ h\ r$   
 $=\ ((ItemBool\ (Snd\ (e\ tl\ hrl\ r))$   
 $\vee\ \neg\ c\ dominates\ w\ r),$   
 $MkDerTableRow$   
 $(w\ r)$   
 $(DTR\_row\ r)$   
 $(DTR\_cols\ r))$   
 $in\ Map\ Snd\ (Map\ h\ hrl\ \upharpoonright\ \{(t, r)\mid t\}))$

**PutInGroup**      $\vdash\ \forall\ gpby\ x\ gp\ gps$

- $PutInGroup\ gpby\ x\ [] = [[x]]$   
 $\wedge\ PutInGroup\ gpby\ x\ (Cons\ gp\ gps)$   
 $=\ (if\ gpby\ x = gpby\ (Head\ gp)$   
 $then\ Cons\ (Cons\ x\ gp)\ gps$   
 $else\ Cons\ gp\ (PutInGroup\ gpby\ x\ gps))$

**MakeGroups**      $\vdash\ \forall\ gpby\ x\ xs$

- $MakeGroups\ gpby\ [] = []$

---

---

$\wedge$  *MakeGroups* *gpsy* (*Cons* *x xs*)  
 $=$  *PutInGroup* *gpsy* *x* (*MakeGroups* *gpsy* *xs*)

**ListNth**  $\vdash \forall n\ nl\ list$

- *ListNth* [] *list* = []
- $\wedge$  *ListNth* (*Cons* *n nl*) *list*  
 $=$  *Cons*  
*(if*  $1 \leq n \wedge n \leq \# list$   
*then* *Nth* *list* *n*  
*else* *Arbitrary*)  
*(ListNth* *nl list*)

**CommonValue**  $\vdash \forall e$

- *CommonValue* *e*  
 $=$  (*let* *pick* *il*  
 $=$  (*if*  $\# (Elems\ il) = 1$   
*then* *Head* *il*  
*else* *Arbitrary*) *in* *SetFuncAll* *pick* *e*)

**Group**  $\vdash \forall cc\ tl\ rl\ gbsterling\ gbclass\ having$

- *Group* *cc* *tl* *rl* *gbsterling* *gbclass* *having*  
 $=$  (*let* *gpsy* *row*  
 $=$  (*ListNth*  
*gbsterling*  
*(Map* *Snd* (*DTR\_cols* *row*)),  
*ListNth* *gbclass* (*Map* *Fst* (*DTR\_cols* *row*)))  
*in* *let* *gbc* *row*  
 $=$  *lubl*  
*(ListNth*  
*gbsterling*  
*(Map* *Fst* (*DTR\_cols* *row*)))  
*in* *let* *gps* = *MakeGroups* *gpsy* *rl*  
*in* *let* *has\_test* *gp*  
 $=$  *CommonValue* *having* *tl* *gp* *Arbitrary*  
*in* *let* *cl*  
 $=$  (*if*  
*cc* *dominates* *lubl* (*Map* *gbc* *rl*)  
*then*  
*lubl* (*Map* (*Fst* *o* *has\_test*) *gps*)  
*else* *lubl* (*Map* *gbc* *rl*)  
*in* *let* *wanted\_gps*  
 $=$  *gps*  
 $\uparrow$  {*gp*  
| *ItemBool* (*Snd* (*has\_test* *gp*))}  
*in* (*cl*, *wanted\_gps*))

**TableContents**  $\vdash \forall i$

- *TableContents* *i*  
 $=$  ( $\lambda$  *tl*  
  - *if*  $1 \leq i \wedge i \leq \# tl$   
*then* (*lattice\_bottom*, *Nth* *tl* *i*))

---

**AllTuples**  $\vdash \forall cc \text{ sellist fromspec where gbsterling gbclass having}$

- *AllTuples*
  - cc*
  - sellist*
  - fromspec*
  - where*
  - gbsterling*
  - gbclass*
  - having*
- $= (\lambda tl$ 
  - *(let (c1, tabs)*
    - $= \text{Split (Map } (\lambda te \bullet te tl) \text{ fromspec)}$
    - in let (ts, tab1) = Join tabs*
    - in let tab2 = Where cc tl tab1 where*
    - in let (cl1, gps)*
      - $= \text{Group}$
      - cc*
      - tl*
      - tab2*
      - gbsterling*
      - gbclass*
      - having*
    - in let cl2*
      - $= (\text{if } cc \text{ dominates lubl } cl1$
      - $\text{then } cl1$
      - $\text{else lubl } cl1)$

$\text{in } (cl2, \text{Project } ts \text{ tl } \text{sellist } \text{gps}))$

**RemoveDuplicates**

$\vdash \forall x xs$

- *RemoveDuplicates [] = []*
- $\wedge \text{RemoveDuplicates (Cons } x \text{ xs)}$
- $= \text{Cons } x \text{ (RemoveDuplicates } xs \uparrow \{y | \neg y = x\})$

**DistinctTuples**

$\vdash \forall cc \text{ sellist fromspec where gbsterling gbclass having}$

- *DistinctTuples*
  - cc*
  - sellist*
  - fromspec*
  - where*
  - gbsterling*
  - gbclass*
  - having*
- $= (\lambda tl$ 
  - *(let (c1, tabs)*
    - $= \text{Split (Map } (\lambda te \bullet te tl) \text{ fromspec)}$
    - in let (ts, tab1) = Join tabs*
    - in let tab2 = Where cc tl tab1 where*

---

*in let (cl, gps)*  
 = *Group*  
*cc*  
*tl*  
*tab2*  
*gbsterling*  
*gbclass*  
*having*  
*in let rem\_dups tab*  
 = *MkDerTable*  
 (*DT\_spec tab*)  
 (*RemoveDuplicates*  
 (*DT\_rows tab*))  
*in (cl lub lubl cll,*  
*rem\_dups*  
 (*Project ts tl sellist gps*)))

**Evaluate**  $\vdash \forall cc$  *sellist fromspec where gbsterling gbclass having*

- *Evaluate*  
*cc*  
*sellist*  
*fromspec*  
*where*  
*gbsterling*  
*gbclass*  
*having*  
 = ( $\lambda$  *tl*
  - (*let (cll, tabs)*  
 = *Split (Map ( $\lambda$  te • te tl) fromspec)*  
*in let (ts, tab1) = Join tabs*  
*in let tab2 = Where cc tl tab1 where*  
*in let (cl, gps)*  
 = *Group*  
*cc*  
*tl*  
*tab2*  
*gbsterling*  
*gbclass*  
*having*  
*in (cl lub lubl cll,*  
*EvalProject ts tl sellist gps*)))

$\bigcap_2$   $\vdash \forall u$

- $\bigcap_2 u$   
 = ( $\bigcap \{a \mid \exists b \bullet (a, b) \in u\}, \bigcap \{b \mid \exists a \bullet (a, b) \in u\}$ )

**TableComputations**  
**ValueComputations**

$\vdash$  *ConstSpec*  
 ( $\lambda$  (*TableComputations'*, *ValueComputations'*)  
 •  $\forall cc$ )

---

- (*TableComputations'* cc,  
*ValueComputations'* cc)
- $$= \bigcap_2$$
- {(tes, es)
  - | ( $\forall ci \bullet \text{DenoteConstant } ci \in es$ )
  - $\wedge (\forall i \bullet \text{Contents } i \in es)$
  - $\wedge (\forall i \bullet \text{Classification } i \in es)$
  - $\wedge \text{CountAll} \in es$
  - $\wedge (\forall f e \bullet e \in es \Rightarrow \text{MonOp } f e \in es)$
  - $\wedge (\forall f e1 e2$ 
    - $e1 \in es \wedge e2 \in es$
    - $\Rightarrow \text{BinOp } f e1 e2 \in es)$
  - $\wedge (\forall f e1 e2 e3$ 
    - $e1 \in es \wedge e2 \in es \wedge e3 \in es$
    - $\Rightarrow \text{TriOp } f e1 e2 e3 \in es)$
  - $\wedge (\forall el$ 
    - $\text{Elems } el \subseteq es$
    - $\Rightarrow \text{BinOpAnd } cc el \in es)$
  - $\wedge (\forall el$ 
    - $\text{Elems } el \subseteq es$
    - $\Rightarrow \text{BinOpOr } cc el \in es)$
  - $\wedge (\forall te cel ee$ 
    - $te \in es$
    - $\wedge \text{Elems } (\text{Map } \text{Fst } cel) \subseteq es$
    - $\wedge \text{Elems } (\text{Map } \text{Snd } cel) \subseteq es$
    - $\wedge ee \in es$
    - $\Rightarrow \text{CaseVal } cc te cel ee \in es)$
  - $\wedge (\forall cel ee$ 
    - $\text{Elems } (\text{Map } \text{Fst } cel) \subseteq es$
    - $\wedge \text{Elems } (\text{Map } \text{Snd } cel) \subseteq es$
    - $\wedge ee \in es$
    - $\Rightarrow \text{Case } cc cel ee \in es)$
  - $\wedge (\forall e$ 
    - $e \in es \Rightarrow \text{SetFuncAllAnd } cc e \in es)$
  - $\wedge (\forall e$ 
    - $e \in es \Rightarrow \text{SetFuncAllOr } cc e \in es)$
  - $\wedge (\forall e$ 
    - $e \in es \Rightarrow \text{CountNonNull } e \in es)$
  - $\wedge (\forall e$ 
    - $e \in es \Rightarrow \text{CountDistinct } e \in es)$
  - $\wedge (\forall e$ 
    - $e \in es \Rightarrow \text{CommonValue } e \in es)$
  - $\wedge (\forall f e$ 
    - $e \in es \Rightarrow \text{SetFuncAll } f e \in es)$
  - $\wedge (\forall f e$ 
    - $e \in es$
    - $\Rightarrow \text{SetFuncDistinct } f e \in es)$
  - $\wedge (\forall te$

- $te \in tes$   
 $\Rightarrow$  *ExistsTuples*  $cc\ te \in es$ )
- $\wedge (\forall te$
- $te \in tes$   
 $\Rightarrow$  *SingleValue*  $cc\ te \in es$ )
- $\wedge$  *JoinedRowExistence*  $cc \in es$
- $\wedge (\forall i \bullet$  *TableContents*  $i \in tes$ )
- $\wedge (\forall esl\ tel\ e1\ ml\ nl\ e2$
- *Elms* (*Map Fst esl*)  $\subseteq es$   
 $\wedge$  *Elms tel*  $\subseteq tes$   
 $\wedge e1 \in es$   
 $\wedge e2 \in es$   
 $\Rightarrow$  *AllTuples*  
 $cc$   
 $esl$   
 $tel$   
 $e1$   
 $ml$   
 $nl$   
 $e2$   
 $\in tes$ ))

(*TableComputations*, *ValueComputations*)

### OkTableComputation

- $\vdash \forall cc\ te$
- $te \in OkTableComputation\ cc$   
 $\Leftrightarrow$  (*let*  $c\ tl = Fst\ (te\ tl)$   
*in let*  $f\ tl = (Snd\ (te\ tl), \[])$   
*in RiskInputs*  $cc\ f$   
 $\subseteq \{tl \mid \neg cc\ dominates\ c\ tl\}$ )

### OkSTP

- $\vdash \forall compile\ stp$
- $stp \in OkSTP\ compile$   
 $\Leftrightarrow (\forall q\ c$
- *isError* (*stp* ( $q, c$ ))  
 $\vee$  (*let* ( $dq, ocq, pars$ ) = *destVal* (*stp* ( $q, c$ ))  
*in*  $\exists\ dte$
- $dte \in TableComputations\ c$   
 $\wedge$  *compile*  $dq$   
 $= (\lambda\ tl \bullet (Snd\ (dte\ tl), \[]))$   
 $\wedge (\forall tl$
- $\neg c\ dominates\ Fst\ (dte\ tl)$   
 $\Rightarrow$  *IsL*  $ocq$   
 $\wedge$  *is\_select* (*OutL*  $ocq$ )  
 $\wedge \neg DT\_rows$   
 $(Fst$   
 $(compile$   
 $(OutL\ ocq)$   
 $tl))$   
 $= \[]))$

- 
- OK\_TC<sub>d</sub>**      $\vdash \forall c \ tc$
- $tc \in OK\_TC_d \ c$
  - $\Leftrightarrow (\forall tl_0 \ tl_1$
  - $Map \ (HideDerTable \ c) \ tl_0$
  - $= Map \ (HideDerTable \ c) \ tl_1$
  - $\wedge \neg \ HideDerTable \ c \ (Snd \ (tc \ tl_0))$
  - $= HideDerTable \ c \ (Snd \ (tc \ tl_1))$
  - $\Rightarrow \neg \ c \ dominates \ Fst \ (tc \ tl_0))$
- OK\_VC<sub>d</sub>**      $\vdash \forall c \ vc$
- $vc \in OK\_VC_d \ c$
  - $\Leftrightarrow (\forall tl_0 \ tl_1 \ rl_0 \ rl_1 \ r_0 \ r_1$
  - $Map \ (HideDerTable \ c) \ tl_0$
  - $= Map \ (HideDerTable \ c) \ tl_1$
  - $\wedge Map \ (HideDerTableRow \ c) \ rl_0$
  - $= Map \ (HideDerTableRow \ c) \ rl_1$
  - $\wedge HideDerTableRow \ c \ r_0$
  - $= HideDerTableRow \ c \ r_1$
  - $\wedge \neg \ Snd \ (vc \ tl_0 \ rl_0 \ r_0)$
  - $= Snd \ (vc \ tl_1 \ rl_1 \ r_1)$
  - $\Rightarrow \neg \ c \ dominates \ Fst \ (vc \ tl_0 \ rl_0 \ r_0))$
- OK\_VC<sub>c</sub>**      $\vdash \forall c \ vc$
- $vc \in OK\_VC_c \ c$
  - $\Leftrightarrow (\forall tl_0 \ tl_1 \ rl_0 \ rl_1 \ r_0 \ r_1$
  - $Map \ (HideDerTable \ c) \ tl_0$
  - $= Map \ (HideDerTable \ c) \ tl_1$
  - $\wedge Map \ (HideDerTableRow \ c) \ rl_0$
  - $= Map \ (HideDerTableRow \ c) \ rl_1$
  - $\wedge HideDerTableRow \ c \ r_0$
  - $= HideDerTableRow \ c \ r_1$
  - $\Rightarrow Fst \ (vc \ tl_0 \ rl_0 \ r_0)$
  - $= Fst \ (vc \ tl_1 \ rl_1 \ r_1))$
- OK\_TC<sub>c</sub>**      $\vdash \forall c \ tc$
- $tc \in OK\_TC_c \ c$
  - $\Leftrightarrow (\forall tl_0 \ tl_1$
  - $Map \ (HideDerTable \ c) \ tl_0$
  - $= Map \ (HideDerTable \ c) \ tl_1$
  - $\Rightarrow Fst \ (tc \ tl_0) = Fst \ (tc \ tl_1))$

## 29 THE THEORY fef033

### 29.1 Parents

*fef031 fef032*

### 29.2 Children

*fef035*

---

### 29.3 Theorems

**TableComputations\_consistent**  
**ValueComputations\_consistent**

⊢ *Consistent*

(λ (TableComputations' ValueComputations')

• ∀ cc

• (TableComputations' cc,  
ValueComputations' cc)

= ∩<sub>2</sub>

{(tes, es)

| (∀ ci • DenoteConstant ci ∈ es)

∧ (∀ i • Contents i ∈ es)

∧ (∀ i • Classification i ∈ es)

∧ CountAll ∈ es

∧ (∀ f e • e ∈ es ⇒ MonOp f e ∈ es)

∧ (∀ f e1 e2

• e1 ∈ es ∧ e2 ∈ es

⇒ BinOp f e1 e2 ∈ es)

∧ (∀ f e1 e2 e3

• e1 ∈ es ∧ e2 ∈ es ∧ e3 ∈ es

⇒ TriOp f e1 e2 e3 ∈ es)

∧ (∀ el

• Elems el ⊆ es

⇒ BinOpAnd cc el ∈ es)

∧ (∀ el

• Elems el ⊆ es

⇒ BinOpOr cc el ∈ es)

∧ (∀ te cel ee

• te ∈ es

∧ Elems (Map Fst cel) ⊆ es

∧ Elems (Map Snd cel) ⊆ es

∧ ee ∈ es

⇒ CaseVal cc te cel ee ∈ es)

∧ (∀ cel ee

• Elems (Map Fst cel) ⊆ es

∧ Elems (Map Snd cel) ⊆ es

∧ ee ∈ es

⇒ Case cc cel ee ∈ es)

∧ (∀ e

• e ∈ es ⇒ SetFuncAllAnd cc e ∈ es)

∧ (∀ e

• e ∈ es ⇒ SetFuncAllOr cc e ∈ es)

∧ (∀ e

• e ∈ es ⇒ CountNonNull e ∈ es)

∧ (∀ e

• e ∈ es ⇒ CountDistinct e ∈ es)

∧ (∀ e

---

- $e \in es \Rightarrow \text{CommonValue } e \in es$
- $\wedge (\forall f e$
- $e \in es \Rightarrow \text{SetFuncAll } f e \in es$
- $\wedge (\forall f e$
- $e \in es$
- $\Rightarrow \text{SetFuncDistinct } f e \in es$ )
- $\wedge (\forall te$
- $te \in tes$
- $\Rightarrow \text{ExistsTuples } cc te \in es$ )
- $\wedge (\forall te$
- $te \in tes$
- $\Rightarrow \text{SingleValue } cc te \in es$ )
- $\wedge \text{JoinedRowExistence } cc \in es$
- $\wedge (\forall i \bullet \text{TableContents } i \in tes)$
- $\wedge (\forall esl tel e1 ml nl e2$
- $\text{Elems } (\text{Map } \text{Fst } esl) \subseteq es$
- $\wedge \text{Elems } tel \subseteq tes$
- $\wedge e1 \in es$
- $\wedge e2 \in es$
- $\Rightarrow \text{AllTuples}$
- $cc$
- $esl$
- $tel$
- $e1$
- $ml$
- $nl$
- $e2$
- $\in tes\})$ )

**BoolItem\_OneOne\_lemma**

$$\vdash \forall i1 i2 \bullet \text{BoolItem } i1 = \text{BoolItem } i2 \Leftrightarrow i1 \Leftrightarrow i2$$
**HideDerTableRow\_Length\_lemma**

$$\vdash \forall c r1 r2$$

- $\text{HideDerTableRow } c r1 = \text{HideDerTableRow } c r2$
- $\Rightarrow \# (\text{DTR\_cols } r1) = \# (\text{DTR\_cols } r2)$

**Nth\_HideDerTableRow\_lemma**

$$\vdash \forall c r i$$

- $1 \leq i \wedge i \leq \# (\text{DTR\_cols } r)$
- $\Rightarrow \text{Nth } (\text{DTR\_cols } (\text{HideDerTableRow } c r)) i$
- $= (\text{Fst } (\text{Nth } (\text{DTR\_cols } r) i),$
- $(\text{if } c \text{ dominates } \text{Fst } (\text{Nth } (\text{DTR\_cols } r) i)$
- $\text{then } \text{Snd } (\text{Nth } (\text{DTR\_cols } r) i)$
- $\text{else}$

$$\text{ValuedItemItem}$$

$$(\text{MkValuedItem } \text{sterling } \text{dummyVal}))$$
**DTR\_row\_o\_HideDerTableRow\_lemma**

$$\vdash \forall c \bullet \text{DTR\_row } o \text{ HideDerTableRow } c = \text{DTR\_row}$$
**fun\_if\_thm**

$$\vdash \forall f a b c$$

- $f (\text{if } a \text{ then } b \text{ else } c) = (\text{if } a \text{ then } f b \text{ else } f c)$

**└\_null\_map\_hide\_lemma**

$$\begin{aligned} &\vdash \forall c \text{ } rl \\ &\bullet \text{ } rl \\ &\quad \uparrow \{r \\ &\quad |c \text{ dominates } DTR\_row \ r \\ &\quad \wedge \ c \text{ dominates } DTR\_where \ r\} \\ &= [] \\ &\Leftrightarrow \text{Map} \\ &\quad (\text{HideDerTableRow } c) \\ &\quad (rl \uparrow \{r|c \text{ dominates } DTR\_row \ r\}) \\ &\quad \uparrow \{r|c \text{ dominates } DTR\_where \ r\} \\ &= [] \end{aligned}$$
**map\_hide\_map\_hide\_└\_lemma**

$$\begin{aligned} &\vdash \forall c \text{ } rl1 \text{ } rl2 \\ &\bullet \text{Map } (\text{HideDerTableRow } c) \text{ } rl1 \\ &\quad = \text{Map } (\text{HideDerTableRow } c) \text{ } rl2 \\ &\Rightarrow \text{Map} \\ &\quad (\text{HideDerTableRow } c) \\ &\quad (rl1 \uparrow \{r|c \text{ dominates } DTR\_where \ r\}) \\ &= \text{Map} \\ &\quad (\text{HideDerTableRow } c) \\ &\quad (rl2 \uparrow \{r|c \text{ dominates } DTR\_where \ r\}) \end{aligned}$$
**└\_┐\_lemma**

$$\vdash \forall l \ a \ b \bullet \ l \uparrow (a \cap b) = l \uparrow a \uparrow b$$
**split\_thm**

$$\begin{aligned} &\vdash \forall xy \text{ } list \\ &\bullet \text{Split } [] = ([], []) \\ &\quad \wedge \text{Split } (\text{Cons } xy \text{ } list) \\ &\quad = (\text{Cons } (\text{Fst } xy) (\text{Fst } (\text{Split } list)), \\ &\quad \text{Cons } (\text{Snd } xy) (\text{Snd } (\text{Split } list))) \end{aligned}$$

$$\text{length\_0\_thm} \quad \vdash \forall list \bullet \# \text{ } list = 0 \Leftrightarrow list = []$$

$$\text{length\_1\_thm} \quad \vdash \forall list \bullet \# \text{ } list = 1 \Leftrightarrow (\exists x \bullet list = [x])$$
**fst\_snd\_split\_thm**

$$\begin{aligned} &\vdash \forall list \\ &\bullet \text{Fst } (\text{Split } list) = \text{Map } \text{Fst } list \\ &\quad \wedge \text{Snd } (\text{Split } list) = \text{Map } \text{Snd } list \end{aligned}$$
**lubl\_lemma**

$$\begin{aligned} &\vdash \forall c \text{ } cl \\ &\bullet \text{lubl } [] = \text{lattice\_bottom} \\ &\quad \wedge \text{lubl } (\text{Cons } c \text{ } cl) = c \text{ lub } \text{lubl } cl \end{aligned}$$
**lub\_lattice\_bottom\_thm**

$$\vdash \forall c \bullet c \text{ lub } \text{lattice\_bottom} = c$$
**dominates\_lubl\_map\_hide\_lemma**

$$\begin{aligned} &\vdash \forall cc \text{ } cil \\ &\bullet cc \text{ dominates } \text{lubl } (\text{Map } \text{Fst } cil) \\ &\Rightarrow \text{Map} \\ &\quad (\lambda (c, i) \\ &\quad \bullet (c, \\ &\quad \quad (\text{if } cc \text{ dominates } c \\ &\quad \quad \text{then } i \\ &\quad \quad \text{else } \text{Arbitrary}))) \end{aligned}$$

$$cil$$

$$= cil$$
**CaseVal\_lemma**

$$\vdash \forall cc \text{ } \text{tst } cev \text{ } cevs \text{ } ev$$

- $CaseVal \text{ } cc \text{ } \text{tst } [] \text{ } ev$ 

$$= (\lambda \text{ } tl \text{ } rl \text{ } r$$
  - $(let \text{ } (tc, ti) = \text{tst } tl \text{ } rl \text{ } r$ 

$$\text{ in } let \text{ } (ec, ei) = ev \text{ } tl \text{ } rl \text{ } r$$

$$\text{ in if } cc \text{ dominates } tc$$

$$\text{ then } (ec, ei)$$

$$\text{ else } (tc, ei)))$$
- $\wedge CaseVal \text{ } cc \text{ } \text{tst } (Cons \text{ } cev \text{ } cevs) \text{ } ev$ 

$$= (\lambda \text{ } tl \text{ } rl \text{ } r$$
  - $(let \text{ } (cr, ir)$ 

$$= CaseVal \text{ } cc \text{ } \text{tst } cevs \text{ } ev \text{ } tl \text{ } rl \text{ } r$$

$$\text{ in } let \text{ } (tc, ti) = \text{tst } tl \text{ } rl \text{ } r$$

$$\text{ in } let \text{ } (ce, cv) = cev$$

$$\text{ in } let \text{ } (cec, cei) = ce \text{ } tl \text{ } rl \text{ } r$$

$$\text{ in } let \text{ } (cvc, cvi) = cv \text{ } tl \text{ } rl \text{ } r$$

$$\text{ in if } ti = cei$$

$$\text{ then}$$

$$\text{ if}$$
    - $cc \text{ dominates } tc$ 

$$\wedge cc \text{ dominates } cec$$

$$\text{ then } (cvc, cvi)$$
    - $\text{ else if } \neg cc \text{ dominates } tc$ 

$$\text{ then } (tc, cvi)$$
    - $\text{ else } (cec, cvi)$
  - $\text{ else if}$ 
    - $cc \text{ dominates } tc$ 

$$\wedge cc \text{ dominates } cec$$

$$\text{ then } (cr, ir)$$
    - $\text{ else if } \neg cc \text{ dominates } tc$ 

$$\text{ then } (tc, ir)$$
    - $\text{ else } (cec, ir)))$

**OK\_TC<sub>d</sub>-lemma**

$$\vdash \forall c \bullet OK\_TC_d \text{ } c \subseteq OkTableComputation \text{ } c$$
**DenoteConstant\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \text{ } ci \bullet DenoteConstant \text{ } ci \in OK\_VC_d \text{ } c$$
**Contents\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \text{ } i \bullet Contents \text{ } i \in OK\_VC_d \text{ } c$$
**Classification\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \text{ } i \bullet Classification \text{ } i \in OK\_VC_d \text{ } c$$
**CountAll\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \bullet CountAll \in OK\_VC_d \text{ } c$$
**MonOp\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \text{ } f \text{ } vc \bullet vc \in OK\_VC_d \text{ } c \Rightarrow MonOp \text{ } f \text{ } vc \in OK\_VC_d \text{ } c$$
**ComputeAnd\_lemma**

$$\begin{aligned}
& \vdash \forall cc \ cil1 \ cil2 \\
& \bullet \text{Map} \\
& \quad (\lambda (c, i) \\
& \quad \bullet (c, \\
& \quad \quad (\text{if } cc \text{ dominates } c \\
& \quad \quad \text{then } i \\
& \quad \quad \text{else Arbitrary}))) \\
& \quad cil1 \\
& = \text{Map} \\
& \quad (\lambda (c, i) \\
& \quad \bullet (c, \\
& \quad \quad (\text{if } cc \text{ dominates } c \\
& \quad \quad \text{then } i \\
& \quad \quad \text{else Arbitrary}))) \\
& \quad cil2 \\
& \Rightarrow \text{Fst } (\text{ComputeAnd } cc \ cil1) \\
& \quad = \text{Fst } (\text{ComputeAnd } cc \ cil2) \\
& \quad \wedge (cc \text{ dominates } \text{Fst } (\text{ComputeAnd } cc \ cil1)) \\
& \quad \Rightarrow \text{Snd } (\text{ComputeAnd } cc \ cil1) \\
& \quad = \text{Snd } (\text{ComputeAnd } cc \ cil2))
\end{aligned}$$
**BinOpAnd\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
& \vdash \forall c \ vcl \\
& \bullet \text{Elms } vcl \subseteq \text{OK\_VC}_d \ c \wedge \text{Elms } vcl \subseteq \text{OK\_VC}_c \ c \\
& \Rightarrow \text{BinOpAnd } c \ vcl \in \text{OK\_VC}_d \ c
\end{aligned}$$
**ComputeOr\_lemma**

$$\begin{aligned}
& \vdash \forall cc \ cil1 \ cil2 \\
& \bullet \text{Map} \\
& \quad (\lambda (c, i) \\
& \quad \bullet (c, \\
& \quad \quad (\text{if } cc \text{ dominates } c \\
& \quad \quad \text{then } i \\
& \quad \quad \text{else Arbitrary}))) \\
& \quad cil1 \\
& = \text{Map} \\
& \quad (\lambda (c, i) \\
& \quad \bullet (c, \\
& \quad \quad (\text{if } cc \text{ dominates } c \\
& \quad \quad \text{then } i \\
& \quad \quad \text{else Arbitrary}))) \\
& \quad cil2 \\
& \Rightarrow \text{Fst } (\text{ComputeOr } cc \ cil1) \\
& \quad = \text{Fst } (\text{ComputeOr } cc \ cil2) \\
& \quad \wedge (cc \text{ dominates } \text{Fst } (\text{ComputeOr } cc \ cil1)) \\
& \quad \Rightarrow \text{Snd } (\text{ComputeOr } cc \ cil1) \\
& \quad = \text{Snd } (\text{ComputeOr } cc \ cil2))
\end{aligned}$$
**BinOpOr\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
& \vdash \forall c \ vcl \\
& \bullet \text{Elms } vcl \subseteq \text{OK\_VC}_d \ c \wedge \text{Elms } vcl \subseteq \text{OK\_VC}_c \ c
\end{aligned}$$

$$\Rightarrow \text{BinOpOr } c \text{ } vcl \in \text{OK\_VC}_d \text{ } c$$
**BinOp\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ f \ vc1 \ vc2$$

- $vc1 \in \text{OK\_VC}_d \text{ } c \wedge vc2 \in \text{OK\_VC}_d \text{ } c$

$$\Rightarrow \text{BinOp } f \ vc1 \ vc2 \in \text{OK\_VC}_d \text{ } c$$
**TriOp\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ f \ vc1 \ vc2 \ vc3$$

- $vc1 \in \text{OK\_VC}_d \text{ } c \wedge vc2 \in \text{OK\_VC}_d \text{ } c \wedge vc3 \in \text{OK\_VC}_d \text{ } c$

$$\Rightarrow \text{TriOp } f \ vc1 \ vc2 \ vc3 \in \text{OK\_VC}_d \text{ } c$$
**CaseVal\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ te \ cel \ ee$$

- $te \in \text{OK\_VC}_d \text{ } c$

$$\wedge \text{Elems } (\text{Map } \text{Fst } cel) \subseteq \text{OK\_VC}_d \text{ } c$$

$$\wedge \text{Elems } (\text{Map } \text{Snd } cel) \subseteq \text{OK\_VC}_d \text{ } c$$

$$\wedge ee \in \text{OK\_VC}_d \text{ } c$$

$$\Rightarrow \text{CaseVal } c \ te \ cel \ ee \in \text{OK\_VC}_d \text{ } c$$
**Case\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ cel \ ee$$

- $\text{Elems } (\text{Map } \text{Fst } cel) \subseteq \text{OK\_VC}_d \text{ } c$

$$\wedge \text{Elems } (\text{Map } \text{Snd } cel) \subseteq \text{OK\_VC}_d \text{ } c$$

$$\wedge ee \in \text{OK\_VC}_d \text{ } c$$

$$\Rightarrow \text{Case } c \ cel \ ee \in \text{OK\_VC}_d \text{ } c$$
**SetFuncAll\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ f \ vc$$

- $vc \in \text{OK\_VC}_d \text{ } c \Rightarrow \text{SetFuncAll } f \ vc \in \text{OK\_VC}_d \text{ } c$

**SetFuncDistinct\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ f \ vc$$

- $vc \in \text{OK\_VC}_d \text{ } c \Rightarrow \text{SetFuncDistinct } f \ vc \in \text{OK\_VC}_d \text{ } c$

**SetFuncAllAnd\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ vc$$

- $vc \in \text{OK\_VC}_d \text{ } c \wedge vc \in \text{OK\_VC}_c \text{ } c$

$$\Rightarrow \text{SetFuncAllAnd } c \ vc \in \text{OK\_VC}_d \text{ } c$$
**SetFuncAllOr\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ vc$$

- $vc \in \text{OK\_VC}_d \text{ } c \wedge vc \in \text{OK\_VC}_c \text{ } c$

$$\Rightarrow \text{SetFuncAllOr } c \ vc \in \text{OK\_VC}_d \text{ } c$$
**CountNonNull\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ vc \bullet vc \in \text{OK\_VC}_d \text{ } c \Rightarrow \text{CountNonNull } vc \in \text{OK\_VC}_d \text{ } c$$
**CountDistinct\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ vc \bullet vc \in \text{OK\_VC}_d \text{ } c \Rightarrow \text{CountDistinct } vc \in \text{OK\_VC}_d \text{ } c$$
**CommonValue\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ vc \bullet vc \in \text{OK\_VC}_d \text{ } c \Rightarrow \text{CommonValue } vc \in \text{OK\_VC}_d \text{ } c$$
**ExistsTuples\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ tc$$

- $tc \in \text{OK\_TC}_d \text{ } c \wedge tc \in \text{OK\_TC}_c \text{ } c$

$$\Rightarrow \text{ExistsTuples } c \ tc \in \text{OK\_VC}_d \text{ } c$$
**SingleValue\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ tc$$

- $tc \in OK\_TC_d\ c \wedge tc \in OK\_TC_c\ c$   
 $\Rightarrow SingleValue\ c\ tc \in OK\_VC_d\ c$

**JoinedRowExistence\_OK<sub>d</sub>-lemma**

- $\vdash \forall c\ i \bullet JoinedRowExistence\ i \in OK\_VC_d\ c$

**DenoteConstant\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ ci \bullet DenoteConstant\ ci \in OK\_VC_c\ c$

**Contents\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ i \bullet Contents\ i \in OK\_VC_c\ c$

**Classification\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ i \bullet Classification\ i \in OK\_VC_c\ c$

**CountAll\_OK<sub>c</sub>-lemma**

- $\vdash \forall c \bullet CountAll \in OK\_VC_c\ c$

**MonOp\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ f\ vc \bullet vc \in OK\_VC_c\ c \Rightarrow MonOp\ f\ vc \in OK\_VC_c\ c$

**BinOpAnd\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ vcl$ 
  - $Elms\ vcl \subseteq OK\_VC_d\ c \wedge Elms\ vcl \subseteq OK\_VC_c\ c$   
 $\Rightarrow BinOpAnd\ c\ vcl \in OK\_VC_c\ c$

**BinOpOr\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ vcl$ 
  - $Elms\ vcl \subseteq OK\_VC_d\ c \wedge Elms\ vcl \subseteq OK\_VC_c\ c$   
 $\Rightarrow BinOpOr\ c\ vcl \in OK\_VC_c\ c$

**BinOp\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ f\ vc1\ vc2$ 
  - $vc1 \in OK\_VC_c\ c \wedge vc2 \in OK\_VC_c\ c$   
 $\Rightarrow BinOp\ f\ vc1\ vc2 \in OK\_VC_c\ c$

**TriOp\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ f\ vc1\ vc2\ vc3$ 
  - $vc1 \in OK\_VC_c\ c \wedge vc2 \in OK\_VC_c\ c \wedge vc3 \in OK\_VC_c\ c$   
 $\Rightarrow TriOp\ f\ vc1\ vc2\ vc3 \in OK\_VC_c\ c$

**CaseVal\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ te\ cel\ ee$ 
  - $te \in OK\_VC_d\ c$   
 $\wedge te \in OK\_VC_c\ c$   
 $\wedge Elms\ (Map\ Fst\ cel) \subseteq OK\_VC_d\ c$   
 $\wedge Elms\ (Map\ Fst\ cel) \subseteq OK\_VC_c\ c$   
 $\wedge Elms\ (Map\ Snd\ cel) \subseteq OK\_VC_c\ c$   
 $\wedge ee \in OK\_VC_c\ c$   
 $\Rightarrow CaseVal\ c\ te\ cel\ ee \in OK\_VC_c\ c$

**Case\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ cel\ ee$ 
  - $Elms\ (Map\ Fst\ cel) \subseteq OK\_VC_d\ c$   
 $\wedge Elms\ (Map\ Fst\ cel) \subseteq OK\_VC_c\ c$   
 $\wedge Elms\ (Map\ Snd\ cel) \subseteq OK\_VC_c\ c$   
 $\wedge ee \in OK\_VC_c\ c$   
 $\Rightarrow Case\ c\ cel\ ee \in OK\_VC_c\ c$

**SetFuncAll\_OK<sub>c</sub>-lemma**

- $\vdash \forall c\ f\ vc$

•  $vc \in OK\_VC_c c \Rightarrow SetFuncAll f vc \in OK\_VC_c c$

**SetFuncDistinct\_OK<sub>c</sub>-lemma**

$\vdash \forall c f vc$

•  $vc \in OK\_VC_c c \Rightarrow SetFuncDistinct f vc \in OK\_VC_c c$

**SetFuncAllAnd\_OK<sub>c</sub>-lemma**

$\vdash \forall c vc$

•  $vc \in OK\_VC_d c \wedge vc \in OK\_VC_c c$

$\Rightarrow SetFuncAllAnd c vc \in OK\_VC_c c$

**SetFuncAllOr\_OK<sub>c</sub>-lemma**

$\vdash \forall c vc$

•  $vc \in OK\_VC_d c \wedge vc \in OK\_VC_c c$

$\Rightarrow SetFuncAllOr c vc \in OK\_VC_c c$

**CountNonNull\_OK<sub>c</sub>-lemma**

$\vdash \forall c vc \bullet vc \in OK\_VC_c c \Rightarrow CountNonNull vc \in OK\_VC_c c$

**CountDistinct\_OK<sub>c</sub>-lemma**

$\vdash \forall c vc \bullet vc \in OK\_VC_c c \Rightarrow CountDistinct vc \in OK\_VC_c c$

**CommonValue\_OK<sub>c</sub>-lemma**

$\vdash \forall c vc \bullet vc \in OK\_VC_c c \Rightarrow CommonValue vc \in OK\_VC_c c$

**ExistsTuples\_OK<sub>c</sub>-lemma**

$\vdash \forall c tc$

•  $tc \in OK\_TC_d c \wedge tc \in OK\_TC_c c$

$\Rightarrow ExistsTuples c tc \in OK\_VC_c c$

**SingleValue\_OK<sub>c</sub>-lemma**

$\vdash \forall c tc$

•  $tc \in OK\_TC_d c \wedge tc \in OK\_TC_c c$

$\Rightarrow SingleValue c tc \in OK\_VC_c c$

**JoinedRowExistence\_OK<sub>c</sub>-lemma**

$\vdash \forall c i \bullet JoinedRowExistence i \in OK\_VC_c c$

## 30 THE THEORY fef034

### 30.1 Parents

*fef032*

### 30.2 Children

*fef031*

### 30.3 Constants

**FE\_SWORD\_SYSTEM**

*(Query, ANSWER) BEHAVIOURS*

**FE\_SWORD\_SYSTEM\_secure**

*Bool*

**Architecture\_Secure**

*Bool*

**Subsys\_SecureE***Bool***Subsys\_SecureD***Bool***Subsys\_SecureC***Bool***Subsys\_SecureB***Bool***Subsys\_SecureA***Bool***Correct\_Compile\_STP\_secure\_E***Bool***EM\_SecureE** *Bool***OkSTP\_Secure\_E\_Lemma***Bool***TableComputationsSecure***Bool***Correct\_Compile\_OkSTP***Bool*

### 30.4 Definitions

**FE\_SWORD\_SYSTEM** $\vdash FE\_SWORD\_SYSTEM$  $= FE\_SWORD \text{ reprState } TSQltf \text{ STP } outputFilter$ **FE\_SWORD\_SYSTEM\_secure** $\vdash FE\_SWORD\_SYSTEM\_secure \Leftrightarrow FE\_SWORD\_SYSTEM \in secure$ **Architecture\_Secure** $\vdash Architecture\_Secure$  $\Leftrightarrow (TSQltf, STP, outputFilter)$  $\in subsys\_secure \text{ reprState}$  $\Rightarrow FE\_SWORD\_SYSTEM \in secure$ **Subsys\_SecureA****Subsys\_SecureB****Subsys\_SecureC****Subsys\_SecureD****Subsys\_SecureE** $\vdash (Subsys\_SecureA$  $\Leftrightarrow (TSQltf, STP) \in subsys\_secureA \text{ reprState})$  $\wedge (Subsys\_SecureB$  $\Leftrightarrow (TSQltf, STP) \in subsys\_secureB \text{ reprState})$  $\wedge (Subsys\_SecureC$  $\Leftrightarrow (TSQltf, STP) \in subsys\_secureC \text{ reprState})$  $\wedge (Subsys\_SecureD$  $\Leftrightarrow (TSQltf, STP) \in subsys\_secureD \text{ reprState})$  $\wedge (Subsys\_SecureE$  $\Leftrightarrow (TSQltf, STP, outputFilter)$  $\in subsys\_secureE \text{ reprState})$

**Correct\_Compile\_STP\_secure\_E**

$$\begin{aligned} &\vdash \text{Correct\_Compile\_STP\_secure\_E} \\ &\Leftrightarrow (\exists \text{ compile upd} \\ &\quad \bullet (\text{compile}, \text{upd}) \in \text{Correct\_Compile} \\ &\quad \wedge \text{STP} \in \text{STP\_secure\_E compile}) \end{aligned}$$
**EM\_SecureE**

$$\begin{aligned} &\vdash \text{EM\_SecureE} \\ &\Leftrightarrow \text{Correct\_Compile\_STP\_secure\_E} \Rightarrow \text{Subsys\_SecureE} \end{aligned}$$
**OkSTP\_Secure\_E\_Lemma**

$$\begin{aligned} &\vdash \text{OkSTP\_Secure\_E\_Lemma} \\ &\Leftrightarrow (\forall \text{ compile} \bullet \text{OkSTP compile} \subseteq \text{STP\_secure\_E compile}) \end{aligned}$$
**TableComputationsSecure**

$$\begin{aligned} &\vdash \text{TableComputationsSecure} \\ &\Leftrightarrow (\forall \text{ cc} \\ &\quad \bullet \text{TableComputations cc} \subseteq \text{OkTableComputation cc}) \end{aligned}$$
**Correct\_Compile\_OkSTP**

$$\begin{aligned} &\vdash \text{Correct\_Compile\_OkSTP} \\ &\Leftrightarrow (\exists \text{ compile upd} \\ &\quad \bullet (\text{compile}, \text{upd}) \in \text{Correct\_Compile} \\ &\quad \wedge \text{STP} \in \text{OkSTP compile}) \end{aligned}$$
**30.5 Theorems****Theorem1**

$$\begin{aligned} &\text{Architecture\_Secure,} \\ &\text{Subsys\_SecureA,} \\ &\text{Subsys\_SecureB,} \\ &\text{Subsys\_SecureC,} \\ &\text{Subsys\_SecureD,} \\ &\text{Subsys\_SecureE} \\ &\vdash \text{FE\_SWORD\_SYSTEM\_secure} \end{aligned}$$
**Theorem2**

$$\begin{aligned} &\text{Architecture\_Secure,} \\ &\text{Subsys\_SecureA,} \\ &\text{Subsys\_SecureB,} \\ &\text{Subsys\_SecureC,} \\ &\text{Subsys\_SecureD,} \\ &\text{EM\_SecureE,} \\ &\text{Correct\_Compile\_STP\_secure\_E} \\ &\vdash \text{FE\_SWORD\_SYSTEM\_secure} \end{aligned}$$
**Theorem3**

$$\vdash \text{TableComputationsSecure} \Rightarrow \text{OkSTP\_Secure\_E\_Lemma}$$
**Theorem4**

$$\begin{aligned} &\text{Architecture\_Secure,} \\ &\text{Subsys\_SecureA,} \\ &\text{Subsys\_SecureB,} \\ &\text{Subsys\_SecureC,} \\ &\text{Subsys\_SecureD,} \\ &\text{EM\_SecureE,} \\ &\text{Correct\_Compile\_OkSTP,} \\ &\text{TableComputationsSecure} \\ &\vdash \text{FE\_SWORD\_SYSTEM\_secure} \end{aligned}$$

## 31 THE THEORY fef035

### 31.1 Parents

*fef033*

### 31.2 Children

*fef036*

### 31.3 Constants

**GroupA**      *DerTableRow LIST*  
                   → *Errors*  
                   → *Errors*  
                   → *Class × DerTableRow LIST LIST*

**GroupB**      *DerTable LIST*  
                   → *DerTableRow LIST LIST*  
                   → *VALUE\_COMP*  
                   → *Class × DerTableRow LIST LIST*

**W**             *Class*  
                   → *DerTable LIST*  
                   → *DerTableRow LIST*  
                   → *DerTableRow LIST*  
                   → *VALUE\_COMP*  
                   → *DerTableRow LIST*

**H**             *Class → DerTableRow LIST → DerTableRow LIST*

### 31.4 Definitions

**GroupA**       $\vdash \forall rl \text{ gbsterling gbclass}$   
                   • *GroupA rl gbsterling gbclass*  
                   = (let *gpsy row*  
                   = (*ListNth*  
                   *gbsterling*  
                   (*Map Snd (DTR\_cols row)*),  
                   *ListNth gbclass (Map Fst (DTR\_cols row))*))  
                   in let *gbc row*  
                   = *lubl*  
                   (*ListNth*  
                   *gbsterling*  
                   (*Map Fst (DTR\_cols row)*))  
                   in (*lubl (Map gbc rl), MakeGroups gpsy rl*))

**GroupB**       $\vdash \forall tl \text{ gps having}$   
                   • *GroupB tl gps having*  
                   = (let *has\_test gp*  
                   = *CommonValue having tl gp Arbitrary*

---

*in let wanted\_gps*  
 = *gps*  $\uparrow$  {*gp*|*ItemBool* (*Snd* (*has\_test* *gp*))}  
*in* (*lubl* (*Map* (*Fst* *o* *has\_test*) *gps*),  
*wanted\_gps*)

**W**  $\vdash \forall c\ tl\ rl_1\ r\ rl_2\ e$

- $W\ c\ tl\ rl_1\ []\ e = []$
- $\wedge W\ c\ tl\ rl_1\ (Cons\ r\ rl_2)\ e$
- = (*let* *w* = *DTR\_where* *r* *lub* *Fst* (*e* *tl* *rl\_1* *r*)  
*in let* *h*  
 $\Leftrightarrow$  (*ItemBool* (*Snd* (*e* *tl* *rl\_1* *r*))  
 $\vee \neg c$  *dominates* *w*)  
 $\wedge c$  *dominates* *DTR\_row* *r*  
*in if* *h*  
*then*  
*Cons*  
*(MkDerTableRow*  
*w*  
*(DTR\_row* *r)*  
*(DTR\_cols* *r)*  
*(W* *c* *tl* *rl\_1* *rl\_2* *e)*  
*else* *W* *c* *tl* *rl\_1* *rl\_2* *e)*

**H**  $\vdash \forall c\ rl \bullet H\ c\ rl = rl \uparrow \{r | c\ \text{dominates}\ DTR\_row\ r\}$

### 31.5 Theorems

#### length\_map\_lemma

 $\vdash \forall f\ l \bullet \# (Map\ f\ l) = \# l$ 

#### fun\_nth\_map\_lemma

 $\vdash \forall l\ i\ f$ 

- $i \leq \# l \wedge 1 \leq i \Rightarrow f (Nth\ l\ i) = Nth (Map\ f\ l)\ i$

#### fun\_nth\_map\_lemma1

 $\vdash \forall i\ f\ l_1\ l_2$ 

- $i \leq \# l_1 \wedge 1 \leq i \wedge Map\ f\ l_1 = Map\ f\ l_2$   
 $\Rightarrow f (Nth\ l_1\ i) = f (Nth\ l_2\ i)$

#### ProjectData\_lemma

 $\vdash \forall tl\ el\ gp\ gps$ 

- *ProjectData* *tl* *el* [] = []  
 $\wedge ProjectData\ tl\ el\ (Cons\ gp\ gps)$   
 = *Map*  
 ( $\lambda r$   
  - *MkDerTableRow*  
*(DTR\_where* *r)*  
*(DTR\_row* *r)*  
*(Map* ( $\lambda e \bullet e\ tl\ gp\ r) *el*)$*gp*  
 @ *ProjectData* *tl* *el* *gps*

#### DT\_spec\_HideDerTable\_lemma

 $\vdash \forall c\ t \bullet DT\_spec (HideDerTable\ c\ t) = DT\_spec\ t$

**map\_HideDerTable\_map\_DT\_spec\_lemma**

$\vdash \forall c \ ts1 \ ts2$   
 •  $Map \ (HideDerTable \ c) \ ts1 = Map \ (HideDerTable \ c) \ ts2$   
 $\Rightarrow Map \ DT\_spec \ ts1 = Map \ DT\_spec \ ts2$

**HideDerTableData\_lemma**

$\vdash \forall c \ r \ rs$   
 •  $HideDerTableData \ c \ [] = []$   
 $\wedge HideDerTableData \ c \ (Cons \ r \ rs)$   
 $= (if \ c \ dominates \ DTR\_row \ r$   
 then  
 $Cons$   
 $(HideDerTableRow \ c \ r)$   
 $(HideDerTableData \ c \ rs)$   
 else  $HideDerTableData \ c \ rs)$

**JoinRows\_lemma**

$\vdash \forall c \ r1 \ r2 \ rs$   
 •  $JoinRows \ r1 \ [] = []$   
 $\wedge JoinRows \ r1 \ (Cons \ r2 \ rs)$   
 $= Cons$   
 $(MkDerTableRow$   
 $(DTR\_where \ r1 \ lub \ DTR\_where \ r2)$   
 $(DTR\_row \ r1 \ lub \ DTR\_row \ r2)$   
 $(DTR\_cols \ r1 \ @ \ DTR\_cols \ r2))$   
 $(JoinRows \ r1 \ rs)$

**HideDerTable\_flat\_lemma**

$\vdash \forall c \ blks$   
 •  $HideDerTableData \ c \ (Flat \ blks)$   
 $= Flat \ (Map \ (HideDerTableData \ c) \ blks)$

**Join\_lemma1**

$\forall c \ ts1 \ ts2$   
 •  $Map \ (HideDerTable \ c) \ ts1 = Map \ (HideDerTable \ c) \ ts2$   
 $\Rightarrow HideDerTableData \ c \ (JoinData \ (Map \ DT\_rows \ ts1))$   
 $= HideDerTableData$   
 $c$   
 $(JoinData \ (Map \ DT\_rows \ ts2))$

$\vdash \forall tl_0 \ tl_1 \ c$

•  $Map \ (HideDerTable \ c) \ tl_0$   
 $= Map \ (HideDerTable \ c) \ tl_1$   
 $\Rightarrow HideDerTable$   
 $c$   
 $(MkDerTable$   
 $(Fst \ (Join \ tl_0))$   
 $(Snd \ (Join \ tl_0)))$   
 $= HideDerTable$   
 $c$   
 $(MkDerTable$   
 $(Fst \ (Join \ tl_1))$   
 $(Snd \ (Join \ tl_1)))$

**Join\_lemma2**

$\forall c \ ts1 \ ts2$

- 
- $Map (HideDerTable\ c)\ ts1 = Map (HideDerTable\ c)\ ts2$   
 $\Rightarrow Map (HideDerTableData\ c)\ (Map\ DT\_rows\ ts1)$   
 $= Map (HideDerTableData\ c)\ (Map\ DT\_rows\ ts2),$   
 $\forall\ c\ ds1\ ds2$ 
    - $Map (HideDerTableData\ c)\ ds1$   
 $= Map (HideDerTableData\ c)\ ds2$   
 $\Rightarrow HideDerTableData\ c\ (JoinData\ ds1)$   
 $= HideDerTableData\ c\ (JoinData\ ds2)$
  - $\vdash \forall\ c\ ts1\ ts2$ 
    - $Map (HideDerTable\ c)\ ts1$   
 $= Map (HideDerTable\ c)\ ts2$   
 $\Rightarrow HideDerTableData$   
 $\quad c$   
 $\quad (JoinData\ (Map\ DT\_rows\ ts1))$   
 $= HideDerTableData$   
 $\quad c$   
 $\quad (JoinData\ (Map\ DT\_rows\ ts2))$
  - Join\_lemma3**  $\vdash \forall\ c\ ts1\ ts2$ 
    - $Map (HideDerTable\ c)\ ts1 = Map (HideDerTable\ c)\ ts2$   
 $\Rightarrow Map (HideDerTableData\ c)\ (Map\ DT\_rows\ ts1)$   
 $= Map (HideDerTableData\ c)\ (Map\ DT\_rows\ ts2)$
  - Join\_lemma4**  $\forall\ c\ rs1\ rs2$ 
    - *Flat*  
 $(Map$   
 $\quad (\lambda\ r\bullet\ HideDerTableData\ c\ (JoinRows\ r\ rs2))$   
 $\quad rs1)$   
 $= Flat$   
 $(Map$   
 $\quad (\lambda\ r\bullet\ JoinRows\ r\ (HideDerTableData\ c\ rs2))$   
 $\quad (HideDerTableData\ c\ rs1))$
  - $\vdash \forall\ c\ ds1\ ds2$ 
    - $Map (HideDerTableData\ c)\ ds1$   
 $= Map (HideDerTableData\ c)\ ds2$   
 $\Rightarrow HideDerTableData\ c\ (JoinData\ ds1)$   
 $= HideDerTableData\ c\ (JoinData\ ds2)$
  - Join\_lemma5**  $\forall\ c\ r\ rs$ 
    - $HideDerTableData\ c\ (JoinRows\ r\ rs)$   
 $= (if\ c\ dominates\ DTR\_row\ r$   
 $\quad then$   
 $\quad JoinRows$   
 $\quad (HideDerTableRow\ c\ r)$   
 $\quad (HideDerTableData\ c\ rs)$   
 $\quad else\ [])$
  - $\vdash \forall\ c\ rs1\ rs2$ 
    - *Flat*  
 $(Map$   
 $\quad (\lambda\ r$   
 $\quad \bullet\ HideDerTableData\ c\ (JoinRows\ r\ rs2))$
-

$$\begin{aligned}
& rs1) \\
& = Flat \\
& \quad (Map \\
& \quad \quad (\lambda r \\
& \quad \quad \quad \bullet JoinRows r (HideDerTableData c rs2)) \\
& \quad \quad (HideDerTableData c rs1))
\end{aligned}$$

**Join\_lemma6**  $\vdash \forall c r rs$

- $HideDerTableData c (JoinRows r rs)$

$$\begin{aligned}
& = (if c \text{ dominates } DTR\_row r \\
& \quad then \\
& \quad \quad JoinRows \\
& \quad \quad \quad (HideDerTableRow c r) \\
& \quad \quad \quad (HideDerTableData c rs) \\
& \quad else [])
\end{aligned}$$

**Join\_OK<sub>d</sub>\_lemma**

$$\begin{aligned}
& \vdash \forall tl_0 tl_1 c \\
& \quad \bullet Map (HideDerTable c) tl_0 \\
& \quad \quad = Map (HideDerTable c) tl_1 \\
& \quad \Rightarrow HideDerTable \\
& \quad \quad c \\
& \quad \quad (MkDerTable \\
& \quad \quad \quad (Fst (Join tl_0)) \\
& \quad \quad \quad (Snd (Join tl_0))) \\
& \quad = HideDerTable \\
& \quad \quad c \\
& \quad \quad (MkDerTable \\
& \quad \quad \quad (Fst (Join tl_1)) \\
& \quad \quad \quad (Snd (Join tl_1)))
\end{aligned}$$

**AllTuples\_lemma1**

$$\begin{aligned}
& \vdash \forall c tl_0 tl_1 tel \\
& \quad \bullet Elems tel \subseteq OK\_TC_d c \\
& \quad \quad \wedge c \\
& \quad \quad \quad dominates lubl \\
& \quad \quad \quad (Fst (Split (Map (\lambda te \bullet te tl_0) tel))) \\
& \quad \quad \wedge Map (HideDerTable c) tl_0 \\
& \quad \quad \quad = Map (HideDerTable c) tl_1 \\
& \quad \Rightarrow Map \\
& \quad \quad (HideDerTable c) \\
& \quad \quad (Snd (Split (Map (\lambda te \bullet te tl_0) tel))) \\
& \quad = Map \\
& \quad \quad (HideDerTable c) \\
& \quad \quad (Snd (Split (Map (\lambda te \bullet te tl_1) tel)))
\end{aligned}$$

**ProjectData\_lemma1**

$$\begin{aligned}
& \vdash \forall c tl_0 tl_1 rl_0 rl_1 r_0 r_1 sl \\
& \quad \bullet Elems sl \subseteq OK\_VC_d c \cap OK\_VC_c c \\
& \quad \quad \wedge Map (HideDerTable c) tl_0 \\
& \quad \quad \quad = Map (HideDerTable c) tl_1 \\
& \quad \quad \wedge Map (HideDerTableRow c) rl_0
\end{aligned}$$

$$\begin{aligned}
&= \text{Map } (\text{HideDerTableRow } c) \text{ } rl_1 \\
&\wedge \text{HideDerTableRow } c \text{ } rl_0 = \text{HideDerTableRow } c \text{ } rl_1 \\
&\Rightarrow \text{HideDerTableRow} \\
&\quad c \\
&\quad (\text{MkDerTableRow} \\
&\quad\quad (\text{DTR\_where } r_0) \\
&\quad\quad (\text{DTR\_row } r_0) \\
&\quad\quad (\text{Map } (\lambda e \bullet e \text{ } tl_0 \text{ } rl_0 \text{ } r_0) \text{ } sl)) \\
&= \text{HideDerTableRow} \\
&\quad c \\
&\quad (\text{MkDerTableRow} \\
&\quad\quad (\text{DTR\_where } r_1) \\
&\quad\quad (\text{DTR\_row } r_1) \\
&\quad\quad (\text{Map } (\lambda e \bullet e \text{ } tl_1 \text{ } rl_1 \text{ } r_1) \text{ } sl))
\end{aligned}$$

**HideDerTableData\_lemma1**

$$\begin{aligned}
&\vdash \forall c \text{ } rl_0 \text{ } rl_1 \\
&\quad \bullet \text{Map } (\text{HideDerTableRow } c) \text{ } rl_0 \\
&\quad\quad = \text{Map } (\text{HideDerTableRow } c) \text{ } rl_1 \\
&\quad\quad \Rightarrow \text{HideDerTableData } c \text{ } rl_0 \\
&\quad\quad = \text{HideDerTableData } c \text{ } rl_1
\end{aligned}$$

**Group\_lemma1**  $\vdash \forall cc \text{ } tl \text{ } rl \text{ } gbsterling \text{ } gbclass \text{ } having$ 

$$\begin{aligned}
&\bullet \text{Group } cc \text{ } tl \text{ } rl \text{ } gbsterling \text{ } gbclass \text{ } having \\
&\quad = (\text{let } (c1, \text{gps}) = \text{GroupA } rl \text{ } gbsterling \text{ } gbclass \\
&\quad\quad \text{in let } (c2, \text{wanted\_gps}) = \text{GroupB } tl \text{ } \text{gps} \text{ } having \\
&\quad\quad \text{in } ((\text{if } cc \text{ } \text{dominates } c1 \text{ then } c2 \text{ else } c1), \\
&\quad\quad \text{wanted\_gps}))
\end{aligned}$$

**Group\_lemma2**  $\forall rl_0 \text{ } rl_1 \text{ } c \text{ } gbsterling \text{ } gbclass$ 

$$\begin{aligned}
&\bullet c \text{ } \text{dominates } \text{lubl } (\text{Map } \text{DTR\_row } rl_0) \\
&\quad \wedge c \text{ } \text{dominates } \text{lubl } (\text{Map } \text{DTR\_row } rl_1) \\
&\quad \wedge \text{Map } (\text{HideDerTableRow } c) \text{ } rl_0 \\
&\quad\quad = \text{Map } (\text{HideDerTableRow } c) \text{ } rl_1 \\
&\quad \wedge \neg \text{Map} \\
&\quad\quad (\text{Map } (\text{HideDerTableRow } c)) \\
&\quad\quad (\text{Snd } (\text{GroupA } rl_0 \text{ } gbsterling \text{ } gbclass)) \\
&\quad = \text{Map} \\
&\quad\quad (\text{Map } (\text{HideDerTableRow } c)) \\
&\quad\quad (\text{Snd } (\text{GroupA } rl_1 \text{ } gbsterling \text{ } gbclass)) \\
&\Rightarrow \neg c \\
&\quad \text{dominates } \text{Fst} \\
&\quad\quad (\text{GroupA } rl_0 \text{ } gbsterling \text{ } gbclass), \\
&\forall tl_0 \text{ } tl_1 \text{ } gps_0 \text{ } gps_1 \text{ } c \text{ } having \\
&\quad \bullet \text{having} \in \text{OK\_VC}_d \text{ } c \cap \text{OK\_VC}_c \text{ } c \\
&\quad\quad \wedge \text{Map } (\text{HideDerTable } c) \text{ } tl_0 \\
&\quad\quad\quad = \text{Map } (\text{HideDerTable } c) \text{ } tl_1 \\
&\quad\quad \wedge \text{Map } (\text{Map } (\text{HideDerTableRow } c)) \text{ } gps_0 \\
&\quad\quad\quad = \text{Map } (\text{Map } (\text{HideDerTableRow } c)) \text{ } gps_1 \\
&\quad \wedge \neg \text{Map} \\
&\quad\quad (\text{Map } (\text{HideDerTableRow } c))
\end{aligned}$$

$$\begin{aligned}
& (Snd (GroupB tl_0 gps_0 having)) \\
& = Map \\
& \quad (Map (HideDerTableRow c)) \\
& \quad (Snd (GroupB tl_1 gps_1 having)) \\
\Rightarrow & \neg c \\
& \quad dominates Fst (GroupB tl_0 gps_0 having) \\
\vdash & \forall tl_0 tl_1 rl_0 rl_1 c e ml nl \\
& \bullet e \in OK\_VC_d c \cap OK\_VC_c c \\
& \quad \wedge c dominates lubl (Map DTR\_row rl_0) \\
& \quad \wedge c dominates lubl (Map DTR\_row rl_1) \\
& \quad \wedge Map (HideDerTable c) tl_0 \\
& \quad = Map (HideDerTable c) tl_1 \\
& \quad \wedge Map (HideDerTableRow c) rl_0 \\
& \quad = Map (HideDerTableRow c) rl_1 \\
& \quad \wedge \neg Map \\
& \quad \quad (Map (HideDerTableRow c)) \\
& \quad \quad (Snd (Group c tl_0 rl_0 ml nl e)) \\
& \quad = Map \\
& \quad \quad (Map (HideDerTableRow c)) \\
& \quad \quad (Snd (Group c tl_1 rl_1 ml nl e)) \\
\Rightarrow & \neg c dominates Fst (Group c tl_0 rl_0 ml nl e)
\end{aligned}$$

**Where\_W\_lemma**

$$\begin{aligned}
\vdash & \forall c tl rl e \\
& \bullet Where c tl rl e = W c tl (H c rl) (H c rl) e
\end{aligned}$$

**Where\_dominates\_lemma**

$$\begin{aligned}
\vdash & \forall tl rl c e \\
& \bullet c dominates lubl (Map DTR\_row (Where c tl rl e))
\end{aligned}$$

**Where\_lemma**  $\vdash \forall tl_0 tl_1 rl_0 rl_1 r_0 r_1 c e$ 

$$\begin{aligned}
& \bullet e \in OK\_VC_d c \cap OK\_VC_c c \\
& \quad \wedge Map (HideDerTable c) tl_0 \\
& \quad = Map (HideDerTable c) tl_1 \\
& \quad \wedge Map (HideDerTableRow c) rl_0 \\
& \quad = Map (HideDerTableRow c) rl_1 \\
& \quad \wedge HideDerTableRow c r_0 = HideDerTableRow c r_1 \\
\Rightarrow & ((ItemBool (Snd (e tl_0 rl_0 r_0))) \\
& \quad \vee \neg c \\
& \quad \quad dominates DTR\_where r_0 \\
& \quad \quad \quad lub Fst (e tl_0 rl_0 r_0)) \\
& \quad \wedge c dominates DTR\_row r_0 \\
& \quad \Leftrightarrow (ItemBool (Snd (e tl_1 rl_1 r_1))) \\
& \quad \vee \neg c \\
& \quad \quad dominates DTR\_where r_1 \\
& \quad \quad \quad lub Fst (e tl_1 rl_1 r_1)) \\
& \quad \wedge c dominates DTR\_row r_1)
\end{aligned}$$

**W\_lemma**

$$\begin{aligned}
\vdash & \forall tl_0 tl_1 rl_0 rl_1 rl_1 rl_2 c e \\
& \bullet e \in OK\_VC_d c \cap OK\_VC_c c \\
& \quad \wedge Map (HideDerTable c) tl_0 \\
& \quad = Map (HideDerTable c) tl_1
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{Map } (\text{HideDerTableRow } c) \text{ } rl_0 \\
& = \text{Map } (\text{HideDerTableRow } c) \text{ } rl_1 \\
& \wedge \text{Map } (\text{HideDerTableRow } c) \text{ } rl_1 \\
& = \text{Map } (\text{HideDerTableRow } c) \text{ } rl_2 \\
& \Rightarrow \text{Map } (\text{HideDerTableRow } c) \text{ } (W \text{ } c \text{ } tl_0 \text{ } rl_1 \text{ } rl_0 \text{ } e) \\
& = \text{Map } (\text{HideDerTableRow } c) \text{ } (W \text{ } c \text{ } tl_1 \text{ } rl_2 \text{ } rl_1 \text{ } e)
\end{aligned}$$

**Where\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
& \vdash \forall tl_0 \text{ } tl_1 \text{ } rl_0 \text{ } rl_1 \text{ } c \text{ } e \\
& \bullet e \in OK\_VC_d \text{ } c \cap OK\_VC_c \text{ } c \\
& \wedge \text{Map } (\text{HideDerTable } c) \text{ } tl_0 \\
& = \text{Map } (\text{HideDerTable } c) \text{ } tl_1 \\
& \wedge \text{HideDerTableData } c \text{ } rl_0 \\
& = \text{HideDerTableData } c \text{ } rl_1 \\
& \Rightarrow \text{Map } (\text{HideDerTableRow } c) \text{ } (\text{Where } c \text{ } tl_0 \text{ } rl_0 \text{ } e) \\
& = \text{Map } (\text{HideDerTableRow } c) \text{ } (\text{Where } c \text{ } tl_1 \text{ } rl_1 \text{ } e)
\end{aligned}$$

**MakeGroups\_lemma1**

$$\vdash \forall xs \text{ } gpbby \bullet \neg [] \in \text{Elems } (\text{MakeGroups } gpbby \text{ } xs)$$

**GroupA\_cols\_lemma1**

$$\begin{aligned}
& \vdash \forall r_1 \text{ } r_2 \text{ } c \\
& \bullet \text{HideDerTableRow } c \text{ } r_1 = \text{HideDerTableRow } c \text{ } r_2 \\
& \Rightarrow \text{Map } \text{Fst } (\text{DTR\_cols } r_1) = \text{Map } \text{Fst } (\text{DTR\_cols } r_2)
\end{aligned}$$

**GroupA\_cols\_lemma2**

$$\begin{aligned}
& \vdash \forall r_1 \text{ } r_2 \text{ } c \text{ } gbsterling \text{ } gbclass \\
& \bullet c \\
& \quad \text{dominates lubl} \\
& \quad (\text{ListNth} \\
& \quad \quad \text{gbsterling} \\
& \quad \quad (\text{Map } \text{Fst } (\text{DTR\_cols } r_1))) \\
& \wedge \text{HideDerTableRow } c \text{ } r_1 = \text{HideDerTableRow } c \text{ } r_2 \\
& \Rightarrow \text{ListNth } gbsterling \text{ } (\text{Map } \text{Snd } (\text{DTR\_cols } r_1)) \\
& = \text{ListNth } gbsterling \text{ } (\text{Map } \text{Snd } (\text{DTR\_cols } r_2))
\end{aligned}$$

**MakeGroups\_lemma2**

$$\begin{aligned}
& \vdash \forall rl_0 \text{ } rl_1 \text{ } g \\
& \bullet \text{Map } g \text{ } rl_0 = \text{Map } g \text{ } rl_1 \\
& \Rightarrow \text{Map } (\text{Map } g) \text{ } (\text{MakeGroups } g \text{ } rl_0) \\
& = \text{Map } (\text{Map } g) \text{ } (\text{MakeGroups } g \text{ } rl_1)
\end{aligned}$$

**MakeGroups\_lemma3**

$$\begin{aligned}
& \vdash \forall rl_0 \text{ } rl_1 \text{ } g \text{ } h \\
& \bullet \text{Map } g \text{ } rl_0 = \text{Map } g \text{ } rl_1 \wedge \text{Map } h \text{ } rl_0 = \text{Map } h \text{ } rl_1 \\
& \Rightarrow \text{Map } (\text{Map } h) \text{ } (\text{MakeGroups } g \text{ } rl_0) \\
& = \text{Map } (\text{Map } h) \text{ } (\text{MakeGroups } g \text{ } rl_1)
\end{aligned}$$

**GroupA\_lemma**

$$\begin{aligned}
& \vdash \forall rl_0 \text{ } rl_1 \text{ } c \text{ } gbsterling \text{ } gbclass \\
& \bullet c \\
& \quad \text{dominates lubl} \\
& \quad (\text{Map} \\
& \quad \quad (\lambda \text{ row} \\
& \quad \quad \bullet \text{lubl} \\
& \quad \quad \quad (\text{ListNth}
\end{aligned}$$

$$\begin{aligned}
& \text{gbsterling} \\
& \quad (\text{Map Fst (DTR\_cols row)})) \\
& \quad \text{rl}_0) \\
& \wedge \text{Map (HideDerTableRow c) rl}_0 \\
& \quad = \text{Map (HideDerTableRow c) rl}_1 \\
\Rightarrow & \text{Map} \\
& \quad (\lambda \text{ row} \\
& \quad \bullet (\text{ListNth} \\
& \quad \quad \text{gbsterling} \\
& \quad \quad (\text{Map Snd (DTR\_cols row)}), \\
& \quad \quad \text{ListNth} \\
& \quad \quad \text{gbclass} \\
& \quad \quad (\text{Map Fst (DTR\_cols row)})) \\
& \quad \text{rl}_0 \\
& \quad = \text{Map} \\
& \quad (\lambda \text{ row} \\
& \quad \bullet (\text{ListNth} \\
& \quad \quad \text{gbsterling} \\
& \quad \quad (\text{Map Snd (DTR\_cols row)}), \\
& \quad \quad \text{ListNth} \\
& \quad \quad \text{gbclass} \\
& \quad \quad (\text{Map Fst (DTR\_cols row)})) \\
& \quad \text{rl}_1
\end{aligned}$$

**GroupA\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
& \vdash \forall \text{rl}_0 \text{rl}_1 \text{c} \text{gbsterling} \text{gbclass} \\
& \bullet \text{c dominates lubl (Map DTR\_row rl}_0) \\
& \quad \wedge \text{c dominates lubl (Map DTR\_row rl}_1) \\
& \quad \wedge \text{Map (HideDerTableRow c) rl}_0 \\
& \quad \quad = \text{Map (HideDerTableRow c) rl}_1 \\
& \quad \wedge \neg \text{Map} \\
& \quad \quad (\text{Map (HideDerTableRow c)}) \\
& \quad \quad (\text{Snd (GroupA rl}_0 \text{gbsterling gbclass)}) \\
& \quad \quad = \text{Map} \\
& \quad \quad (\text{Map (HideDerTableRow c)}) \\
& \quad \quad (\text{Snd (GroupA rl}_1 \text{gbsterling gbclass)}) \\
\Rightarrow & \neg \text{c} \\
& \quad \text{dominates Fst} \\
& \quad (\text{GroupA rl}_0 \text{gbsterling gbclass})
\end{aligned}$$

**GroupB\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
& \vdash \forall \text{tl}_0 \text{tl}_1 \text{gps}_0 \text{gps}_1 \text{c} \text{having} \\
& \bullet \text{having} \in \text{OK\_VC}_d \text{c} \cap \text{OK\_VC}_c \text{c} \\
& \quad \wedge \text{Map (HideDerTable c) tl}_0 \\
& \quad \quad = \text{Map (HideDerTable c) tl}_1 \\
& \quad \wedge \text{Map (Map (HideDerTableRow c)) gps}_0 \\
& \quad \quad = \text{Map (Map (HideDerTableRow c)) gps}_1 \\
& \quad \wedge \neg \text{Map} \\
& \quad \quad (\text{Map (HideDerTableRow c)}) \\
& \quad \quad (\text{Snd (GroupB tl}_0 \text{gps}_0 \text{having)})
\end{aligned}$$

$$\begin{aligned}
&= \text{Map} \\
&\quad (\text{Map} (\text{HideDerTableRow } c)) \\
&\quad (\text{Snd} (\text{GroupB } tl_1 \text{ gps}_1 \text{ having})) \\
&\Rightarrow \neg c \text{ dominates } \text{Fst} (\text{GroupB } tl_0 \text{ gps}_0 \text{ having})
\end{aligned}$$
**Group\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
&\vdash \forall tl_0 \ tl_1 \ rl_0 \ rl_1 \ c \ e \ ml \ nl \\
&\quad \bullet e \in \text{OK\_VC}_d \ c \cap \text{OK\_VC}_c \ c \\
&\quad \wedge c \text{ dominates } \text{lubl} (\text{Map } \text{DTR\_row } rl_0) \\
&\quad \wedge c \text{ dominates } \text{lubl} (\text{Map } \text{DTR\_row } rl_1) \\
&\quad \wedge \text{Map} (\text{HideDerTable } c) \ tl_0 \\
&\quad = \text{Map} (\text{HideDerTable } c) \ tl_1 \\
&\quad \wedge \text{Map} (\text{HideDerTableRow } c) \ rl_0 \\
&\quad = \text{Map} (\text{HideDerTableRow } c) \ rl_1 \\
&\quad \wedge \neg \text{Map} \\
&\quad\quad (\text{Map} (\text{HideDerTableRow } c)) \\
&\quad\quad (\text{Snd} (\text{Group } c \ tl_0 \ rl_0 \ ml \ nl \ e)) \\
&\quad = \text{Map} \\
&\quad\quad (\text{Map} (\text{HideDerTableRow } c)) \\
&\quad\quad (\text{Snd} (\text{Group } c \ tl_1 \ rl_1 \ ml \ nl \ e)) \\
&\Rightarrow \neg c \text{ dominates } \text{Fst} (\text{Group } c \ tl_0 \ rl_0 \ ml \ nl \ e)
\end{aligned}$$
**GroupA\_OK<sub>c</sub>-lemma**

$$\begin{aligned}
&\vdash \forall rl_0 \ rl_1 \ c \ gbsterling \ gbclass \\
&\quad \bullet \text{Map} (\text{HideDerTableRow } c) \ rl_0 \\
&\quad = \text{Map} (\text{HideDerTableRow } c) \ rl_1 \\
&\quad \Rightarrow \text{Fst} (\text{GroupA } rl_0 \ gbsterling \ gbclass) \\
&\quad = \text{Fst} (\text{GroupA } rl_1 \ gbsterling \ gbclass)
\end{aligned}$$
**GroupB\_OK<sub>c</sub>-lemma**

$$\begin{aligned}
&\vdash \forall tl_0 \ tl_1 \ gps_0 \ gps_1 \ c \ having \\
&\quad \bullet \text{having} \in \text{OK\_VC}_c \ c \\
&\quad \wedge \text{Map} (\text{HideDerTable } c) \ tl_0 \\
&\quad = \text{Map} (\text{HideDerTable } c) \ tl_1 \\
&\quad \wedge \text{Map} (\text{Map} (\text{HideDerTableRow } c)) \ gps_0 \\
&\quad = \text{Map} (\text{Map} (\text{HideDerTableRow } c)) \ gps_1 \\
&\quad \Rightarrow \text{Fst} (\text{GroupB } tl_0 \ gps_0 \ having) \\
&\quad = \text{Fst} (\text{GroupB } tl_1 \ gps_1 \ having)
\end{aligned}$$
**Group\_OK<sub>c</sub>-lemma**

$$\begin{aligned}
&\vdash \forall tl_0 \ tl_1 \ rl_0 \ rl_1 \ c \ e \ ml \ nl \\
&\quad \bullet e \in \text{OK\_VC}_c \ c \\
&\quad \wedge \text{Map} (\text{HideDerTableRow } c) \ rl_0 \\
&\quad = \text{Map} (\text{HideDerTableRow } c) \ rl_1 \\
&\quad \wedge \text{Map} (\text{HideDerTable } c) \ tl_0 \\
&\quad = \text{Map} (\text{HideDerTable } c) \ tl_1 \\
&\quad \Rightarrow \text{Fst} (\text{Group } c \ tl_0 \ rl_0 \ ml \ nl \ e) \\
&\quad = \text{Fst} (\text{Group } c \ tl_1 \ rl_1 \ ml \ nl \ e)
\end{aligned}$$
**ProjectData\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
&\vdash \forall tl_0 \ tl_1 \ gps_0 \ gps_1 \ c \ sl \\
&\quad \bullet \text{Elems } sl \subseteq \text{OK\_VC}_d \ c \cap \text{OK\_VC}_c \ c \\
&\quad \wedge \text{Map} (\text{HideDerTable } c) \ tl_0
\end{aligned}$$

$$\begin{aligned}
&= \text{Map } (\text{HideDerTable } c) \text{ } tl_1 \\
&\wedge \text{Map } (\text{Map } (\text{HideDerTableRow } c)) \text{ } gps_0 \\
&= \text{Map } (\text{Map } (\text{HideDerTableRow } c)) \text{ } gps_1 \\
&\Rightarrow \text{Map} \\
&\quad (\text{HideDerTableRow } c) \\
&\quad (\text{ProjectData } tl_0 \text{ } sl \text{ } gps_0) \\
&= \text{Map} \\
&\quad (\text{HideDerTableRow } c) \\
&\quad (\text{ProjectData } tl_1 \text{ } sl \text{ } gps_1)
\end{aligned}$$

**TableContents\_OK<sub>d</sub>-lemma**

$$\vdash \forall c \ i \bullet \text{TableContents } i \in \text{OK\_TC}_d \ c$$

**AllTuples\_OK<sub>d</sub>-lemma**

$$\begin{aligned}
&\vdash \forall c \ esl \ tel \ e1 \ ml \ nl \ e2 \\
&\quad \bullet \text{Elms } (\text{Map } \text{Fst } esl) \subseteq \text{OK\_VC}_d \ c \cap \text{OK\_VC}_c \ c \\
&\quad \wedge \text{Elms } tel \subseteq \text{OK\_TC}_d \ c \\
&\quad \wedge e1 \in \text{OK\_VC}_d \ c \cap \text{OK\_VC}_c \ c \\
&\quad \wedge e2 \in \text{OK\_VC}_d \ c \cap \text{OK\_VC}_c \ c \\
&\Rightarrow \text{AllTuples } c \ esl \ tel \ e1 \ ml \ nl \ e2 \in \text{OK\_TC}_d \ c
\end{aligned}$$

**TableContents\_OK<sub>c</sub>-lemma**

$$\vdash \forall c \ i \bullet \text{TableContents } i \in \text{OK\_TC}_c \ c$$

**AllTuples\_lemma2**

$$\begin{aligned}
&\vdash \forall c \ tl_0 \ tl_1 \ tel \\
&\quad \bullet \text{Elms } tel \subseteq \text{OK\_TC}_c \ c \\
&\quad \wedge \text{Map } (\text{HideDerTable } c) \ tl_0 \\
&\quad = \text{Map } (\text{HideDerTable } c) \ tl_1 \\
&\Rightarrow \text{lubl } (\text{Fst } (\text{Split } (\text{Map } (\lambda te \bullet te \ tl_0) \ tel))) \\
&\quad = \text{lubl } (\text{Fst } (\text{Split } (\text{Map } (\lambda te \bullet te \ tl_1) \ tel)))
\end{aligned}$$

**AllTuples\_OK<sub>c</sub>-lemma**

$$\begin{aligned}
&\vdash \forall c \ esl \ tel \ e1 \ ml \ nl \ e2 \\
&\quad \bullet e1 \in \text{OK\_VC}_c \ c \cap \text{OK\_VC}_d \ c \\
&\quad \wedge \text{Elms } tel \subseteq \text{OK\_TC}_d \ c \cap \text{OK\_TC}_c \ c \\
&\quad \wedge e2 \in \text{OK\_VC}_c \ c \\
&\Rightarrow \text{AllTuples } c \ esl \ tel \ e1 \ ml \ nl \ e2 \in \text{OK\_TC}_c \ c
\end{aligned}$$

**32 THE THEORY fef036****32.1 Parents**

fef035

**32.2 Children**

fef042

### 32.3 Theorems

#### table\_computation\_induction\_thm

$$\begin{aligned}
& \vdash \forall cc\ tcs\ vcs \\
& \bullet (\forall ci \bullet \text{DenoteConstant } ci \in vcs) \\
& \quad \wedge (\forall i \bullet \text{Contents } i \in vcs) \\
& \quad \wedge (\forall i \bullet \text{Classification } i \in vcs) \\
& \quad \wedge \text{CountAll} \in vcs \\
& \quad \wedge (\forall f\ e \bullet e \in vcs \Rightarrow \text{MonOp } f\ e \in vcs) \\
& \quad \wedge (\forall f\ e1\ e2 \\
& \quad \bullet e1 \in vcs \wedge e2 \in vcs \Rightarrow \text{BinOp } f\ e1\ e2 \in vcs) \\
& \quad \wedge (\forall f\ e1\ e2\ e3 \\
& \quad \bullet e1 \in vcs \wedge e2 \in vcs \wedge e3 \in vcs \\
& \quad \quad \Rightarrow \text{TriOp } f\ e1\ e2\ e3 \in vcs) \\
& \quad \wedge (\forall el \bullet \text{Elems } el \subseteq vcs \Rightarrow \text{BinOpAnd } cc\ el \in vcs) \\
& \quad \wedge (\forall el \bullet \text{Elems } el \subseteq vcs \Rightarrow \text{BinOpOr } cc\ el \in vcs) \\
& \quad \wedge (\forall te\ cel\ ee \\
& \quad \bullet te \in vcs \\
& \quad \quad \wedge \text{Elems } (\text{Map } \text{Fst } cel) \subseteq vcs \\
& \quad \quad \wedge \text{Elems } (\text{Map } \text{Snd } cel) \subseteq vcs \\
& \quad \quad \wedge ee \in vcs \\
& \quad \quad \Rightarrow \text{CaseVal } cc\ te\ cel\ ee \in vcs) \\
& \quad \wedge (\forall cel\ ee \\
& \quad \bullet \text{Elems } (\text{Map } \text{Fst } cel) \subseteq vcs \\
& \quad \quad \wedge \text{Elems } (\text{Map } \text{Snd } cel) \subseteq vcs \\
& \quad \quad \wedge ee \in vcs \\
& \quad \quad \Rightarrow \text{Case } cc\ cel\ ee \in vcs) \\
& \quad \wedge (\forall e \bullet e \in vcs \Rightarrow \text{SetFuncAllAnd } cc\ e \in vcs) \\
& \quad \wedge (\forall e \bullet e \in vcs \Rightarrow \text{SetFuncAllOr } cc\ e \in vcs) \\
& \quad \wedge (\forall e \bullet e \in vcs \Rightarrow \text{CountNonNull } e \in vcs) \\
& \quad \wedge (\forall e \bullet e \in vcs \Rightarrow \text{CountDistinct } e \in vcs) \\
& \quad \wedge (\forall e \bullet e \in vcs \Rightarrow \text{CommonValue } e \in vcs) \\
& \quad \wedge (\forall f\ e \bullet e \in vcs \Rightarrow \text{SetFuncAll } f\ e \in vcs) \\
& \quad \wedge (\forall f\ e \bullet e \in vcs \Rightarrow \text{SetFuncDistinct } f\ e \in vcs) \\
& \quad \wedge (\forall te \bullet te \in tcs \Rightarrow \text{ExistsTuples } cc\ te \in vcs) \\
& \quad \wedge (\forall te \bullet te \in tcs \Rightarrow \text{SingleValue } cc\ te \in vcs) \\
& \quad \wedge \text{JoinedRowExistence } cc \in vcs \\
& \quad \wedge (\forall i \bullet \text{TableContents } i \in tcs) \\
& \quad \wedge (\forall esl\ tel\ e1\ ml\ nl\ e2 \\
& \quad \bullet \text{Elems } (\text{Map } \text{Fst } esl) \subseteq vcs \\
& \quad \quad \wedge \text{Elems } tel \subseteq tcs \\
& \quad \quad \wedge e1 \in vcs \\
& \quad \quad \wedge e2 \in vcs \\
& \quad \quad \Rightarrow \text{AllTuples } cc\ esl\ tel\ e1\ ml\ nl\ e2 \in tcs) \\
& \Rightarrow \text{TableComputations } cc \subseteq tcs \\
& \quad \wedge \text{ValueComputations } cc \subseteq vcs
\end{aligned}$$

#### ok\_vc\_tc\_lemmas

$$\vdash (\forall c\ ci \bullet \text{DenoteConstant } ci \in \text{OK\_VC}_d\ c \cap \text{OK\_VC}_c\ c)$$

---


$$\begin{aligned}
& \wedge (\forall c \ i \bullet \text{Contents } i \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ i \bullet \text{Classification } i \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \bullet \text{CountAll} \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ f \ e \\
& \bullet e \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{MonOp } f \ e \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ f \ e1 \ e2 \\
& \bullet e1 \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge e2 \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{BinOp } f \ e1 \ e2 \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ f \ e1 \ e2 \ e3 \\
& \bullet e1 \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge e2 \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge e3 \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{TriOp } f \ e1 \ e2 \ e3 \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ el \\
& \bullet \text{Elems } el \subseteq OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{BinOpAnd } c \ el \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ el \\
& \bullet \text{Elems } el \subseteq OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{BinOpOr } c \ el \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ te \ cel \ ee \\
& \bullet te \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge \text{Elems } (\text{Map } Fst \ cel) \subseteq OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge \text{Elems } (\text{Map } Snd \ cel) \subseteq OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge ee \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{CaseVal } c \ te \ cel \ ee \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ cel \ ee \\
& \bullet \text{Elems } (\text{Map } Fst \ cel) \subseteq OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge \text{Elems } (\text{Map } Snd \ cel) \subseteq OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \wedge ee \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{Case } c \ cel \ ee \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ e \\
& \bullet e \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{SetFuncAllAnd } c \ e \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ e \\
& \bullet e \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{SetFuncAllOr } c \ e \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ e \\
& \bullet e \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{CountNonNull } e \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ e \\
& \bullet e \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{CountDistinct } e \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ e \\
& \bullet e \in OK\_VC_d \ c \cap OK\_VC_c \ c \\
& \quad \Rightarrow \text{CommonValue } e \in OK\_VC_d \ c \cap OK\_VC_c \ c) \\
& \wedge (\forall c \ f \ e
\end{aligned}$$


---

- $e \in OK\_VC_d\ c \cap OK\_VC_c\ c$   
 $\Rightarrow SetFuncAll\ f\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c$   
 $\wedge (\forall\ c\ f\ e$
- $e \in OK\_VC_d\ c \cap OK\_VC_c\ c$   
 $\Rightarrow SetFuncDistinct\ f\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c)$   
 $\wedge (\forall\ c\ te$
- $te \in OK\_TC_d\ c \cap OK\_TC_c\ c$   
 $\Rightarrow ExistsTuples\ c\ te \in OK\_VC_d\ c \cap OK\_VC_c\ c)$   
 $\wedge (\forall\ c\ te$
- $te \in OK\_TC_d\ c \cap OK\_TC_c\ c$   
 $\Rightarrow SingleValue\ c\ te \in OK\_VC_d\ c \cap OK\_VC_c\ c)$   
 $\wedge (\forall\ c$
- $JoinedRowExistence\ c \in OK\_VC_d\ c \cap OK\_VC_c\ c)$   
 $\wedge (\forall\ c\ i \bullet TableContents\ i \in OK\_TC_d\ c \cap OK\_TC_c\ c)$   
 $\wedge (\forall\ c\ esl\ tel\ e1\ ml\ nl\ e2$
- $Elms\ (Map\ Fst\ esl) \subseteq OK\_VC_d\ c \cap OK\_VC_c\ c$   
 $\wedge Elms\ tel \subseteq OK\_TC_d\ c \cap OK\_TC_c\ c$   
 $\wedge e1 \in OK\_VC_d\ c \cap OK\_VC_c\ c$   
 $\wedge e2 \in OK\_VC_d\ c \cap OK\_VC_c\ c$   
 $\Rightarrow AllTuples\ c\ esl\ tel\ e1\ ml\ nl\ e2$   
 $\in OK\_TC_d\ c \cap OK\_TC_c\ c)$

**fef036\_main\_lemma** $\vdash \forall\ cc$ 

- $TableComputations\ cc \subseteq OK\_TC_d\ cc \cap OK\_TC_c\ cc$   
 $\wedge ValueComputations\ cc \subseteq OK\_VC_d\ cc \cap OK\_VC_c\ cc$

**TableComputationsSecure\_thm** $\vdash TableComputationsSecure$ **Theorem5**

*Architecture\_Secure,*  
*Subsys\_SecureA,*  
*Subsys\_SecureB,*  
*Subsys\_SecureC,*  
*Subsys\_SecureD,*  
*View\_t\_secureE,*  
*outputFilter\_secureE,*  
*Correct\_Compile\_OkSTP*  
 $\vdash FE\_SWORD\_SYSTEM\_secure$

**33 THE THEORY fef040****33.1 Parents***fef010 fef003***33.2 Children***fef042*

### 33.3 Constants

$\$ \downarrow$	$Class \rightarrow Class \mathbb{P}$
$\$ \uparrow$	$Class \rightarrow Class \mathbb{P}$
$\$ \perp$	$Class \mathbb{P} \rightarrow Class \mathbb{P}$
<b>Equivalence</b>	$('a \leftrightarrow 'a) \mathbb{P}$
<b>IndexedEquiv</b>	$(Class \mathbb{P} \rightarrow 'a \leftrightarrow 'a) \mathbb{P}$
<b>LiftRel</b>	$(Class \rightarrow 'a \leftrightarrow 'a) \rightarrow Class \mathbb{P} \rightarrow 'a \leftrightarrow 'a$
<b>Independent</b>	$(Class \mathbb{P} \rightarrow 'a \leftrightarrow 'a) \mathbb{P}$
<b>x_ml_secure</b>	$(Class \mathbb{P} \rightarrow 'I \leftrightarrow 'I) \rightarrow (Class \mathbb{P} \rightarrow 'O \leftrightarrow 'O) \rightarrow ('I \rightarrow 'O) \mathbb{P}$
<b>ml_secure</b>	$(Class \rightarrow 'I \leftrightarrow 'I) \rightarrow (Class \rightarrow 'O \leftrightarrow 'O) \rightarrow ('I \rightarrow 'O) \mathbb{P}$
<b>Influenced</b>	$(Class \mathbb{P} \rightarrow 'a \leftrightarrow 'a) \rightarrow 'a \rightarrow ('a \rightarrow 'b) \rightarrow Class \mathbb{P}$
<b>ObservedValue</b>	$('a, 'b) OBSERVATION \rightarrow 'a \rightarrow Class \times 'b$
<b>SameLabVal</b>	$Class \rightarrow (Class \times 'b) \leftrightarrow (Class \times 'b)$
<b>BoundedObs</b>	$(Class \mathbb{P} \rightarrow 'a \leftrightarrow 'a)$ $\rightarrow Class \leftrightarrow ((a \rightarrow Class) \times ('a \rightarrow 'b))$

### 33.4 Type Abbreviations

$('a, 'b) OBSERVATION$   
 $('a, 'b) OBSERVATION$

### 33.5 Fixity

Postfix 300:  $\downarrow \quad \uparrow \quad \perp$

### 33.6 Definitions

$\downarrow$	$\vdash \forall c \bullet c \downarrow = \{d \mid c \text{ dominates } d\}$
$\uparrow$	$\vdash \forall c \bullet c \uparrow = \{d \mid d \text{ dominates } c\}$
$\perp$	$\vdash \forall A$ $\bullet A \perp$ $= \{c$ $\mid \forall d \bullet d \in A \Rightarrow \neg c \text{ dominates } d \wedge \neg d \text{ dominates } c\}$
<b>Equivalence</b>	$\vdash Equivalence = Reflexive \cap Symmetric \cap Transitive$
<b>IndexedEquiv</b>	$\vdash IndexedEquiv$ $= \{s$ $\mid (\forall A \bullet s A \in Equivalence)$ $\wedge (\forall A B \bullet A \subseteq B \Rightarrow s B \subseteq s A)\}$
<b>LiftRel</b>	$\vdash \forall R \bullet LiftRel R = (\lambda A \bullet \cap \{r \mid \exists c \bullet c \in A \wedge r = R c\})$
<b>Independent</b>	$\vdash Independent$ $= \{s$ $\mid \forall A x$ $\bullet (\exists y \bullet \neg (x, y) \in s A)$ $\Rightarrow (\exists z \bullet \neg (x, z) \in s A \wedge (x, z) \in s (A \perp))\}$
<b>x_ml_secure</b>	$\vdash \forall s_I s_O b$

---

	<ul style="list-style-type: none"> <li>• <math>b \in x\_ml\_secure\ s_I\ s_O</math>  <math>\Leftrightarrow s_I \in IndexedEquiv</math>  <math>\wedge s_O \in IndexedEquiv</math>  <math>\wedge (\forall c\ i_1\ i_2</math> <ul style="list-style-type: none"> <li>• <math>(i_1, i_2) \in s_I\ (c \downarrow)</math>  <math>\Rightarrow (b\ i_1, b\ i_2) \in s_O\ (c \downarrow))</math></li> </ul> </li> </ul>
<b>ml_secure</b>	$\vdash \forall r_I\ r_O$ <ul style="list-style-type: none"> <li>• <math>ml\_secure\ r_I\ r_O</math>  <math>= x\_ml\_secure\ (LiftRel\ r_I)\ (LiftRel\ r_O)</math></li> </ul>
<b>Influenced</b>	$\vdash \forall s\ x\ f$ <ul style="list-style-type: none"> <li>• <math>Influenced\ s\ x\ f</math>  <math>= \{c</math>  <math> \exists y\bullet (x, y) \in s\ (\sim (c \uparrow)) \wedge \neg f\ x = f\ y\}</math></li> </ul>
<b>ObservedValue</b>	$\vdash \forall c\ C\ V\ x$ <ul style="list-style-type: none"> <li>• <math>ObservedValue\ (c, C, V)\ x = (C\ x, V\ x)</math></li> </ul>
<b>SameLabVal</b>	$\vdash \forall c$ <ul style="list-style-type: none"> <li>• <math>SameLabVal\ c</math>  <math>= \{(c_1, v_1), c_2, v_2\}</math>  <math> c_1 = c_2 \wedge (c\ dominates\ c_1 \Rightarrow v_1 = v_2)\}</math></li> </ul>
<b>BoundedObs</b>	$\vdash \forall s$ <ul style="list-style-type: none"> <li>• <math>BoundedObs\ s</math>  <math>= \{(c, C, V)</math>  <math> \forall x</math> <ul style="list-style-type: none"> <li>• <math>Influenced\ s\ x\ C \subseteq c \downarrow</math>  <math>\wedge Influenced\ s\ x\ V \subseteq C\ x \downarrow\}</math></li> </ul> </li> </ul>

### 33.7 Theorems

<b><math>\uparrow\downarrow\_thm</math></b>	$\vdash \forall l\ a\ b\bullet l \uparrow a \uparrow b = l \uparrow (a \cap b)$
<b>not_lattice_top_thm</b>	$\vdash \forall c$ <ul style="list-style-type: none"> <li>• <math>\neg c = lattice\_top \Rightarrow \neg lattice\_top = lattice\_bottom</math></li> </ul>
<b>up_down_thm1</b>	$\vdash lattice\_bottom \uparrow = Universe$ $\wedge lattice\_top \downarrow = Universe$
<b>up_down_thm2</b>	$\vdash \forall c$ <ul style="list-style-type: none"> <li>• <math>c \in c \downarrow</math>  <math>\wedge c \in c \uparrow</math>  <math>\wedge lattice\_bottom \in c \downarrow</math>  <math>\wedge lattice\_top \in c \uparrow</math></li> </ul>
<b>up_down_clauses</b>	$\vdash \forall c$ <ul style="list-style-type: none"> <li>• <math>(lattice\_bottom \uparrow = Universe</math>  <math>\wedge lattice\_top \downarrow = Universe)</math>  <math>\wedge c \in c \downarrow</math>  <math>\wedge c \in c \uparrow</math>  <math>\wedge lattice\_bottom \in c \downarrow</math>  <math>\wedge lattice\_top \in c \uparrow</math></li> </ul>
<b>up_down_thm3</b>	$\vdash \forall c\ d$

---

•  $\neg d$  dominates  $c \Rightarrow d \downarrow \subseteq \sim (c \uparrow)$

**lift\_rel\_indexed\_equiv\_thm**  
 $\vdash \forall R$   
 •  $(\forall c \bullet R \ c \in \text{Equivalence}) \Rightarrow \text{LiftRel } R \in \text{IndexedEquiv}$

**same\_ins\_equiv\_thm**  
 $\vdash \forall c \bullet \text{same\_ins } c \in \text{Equivalence}$

**same\_outs\_equiv\_thm**  
 $\vdash \forall c \bullet \text{same\_outs } c \in \text{Equivalence}$

**same\_ins\_same\_outs\_indexed\_equiv\_thm**  
 $\vdash \text{LiftRel same\_ins} \in \text{IndexedEquiv}$   
 $\wedge \text{LiftRel same\_outs} \in \text{IndexedEquiv}$

**equiv\_anti\_mono\_lift\_rel\_thm**  
 $\vdash \forall R$   
 •  $(\forall c \bullet R \ c \in \text{Equivalence})$   
 $\wedge (\forall c \ d \bullet c \text{ dominates } d \Rightarrow R \ c \subseteq R \ d)$   
 $\Rightarrow (\forall c \bullet \text{LiftRel } R \ (c \downarrow) = R \ c)$

**same\_ins\_anti\_mono\_thm**  
 $\vdash \forall c \ d \bullet c \text{ dominates } d \Rightarrow \text{same\_ins } c \subseteq \text{same\_ins } d$

**same\_outs\_anti\_mono\_thm**  
 $\vdash \forall c \ d \bullet c \text{ dominates } d \Rightarrow \text{same\_outs } c \subseteq \text{same\_outs } d$

**lift\_rel\_same\_ins\_same\_outs\_down\_set\_thm**  
 $\vdash \forall c$   
 •  $\text{LiftRel same\_ins } (c \downarrow) = \text{same\_ins } c$   
 $\wedge \text{LiftRel same\_outs } (c \downarrow) = \text{same\_outs } c$

**thm\_040\_1**  $\vdash \forall bm \bullet bm \in \text{secure} \Leftrightarrow bm \in \text{ml\_secure same\_ins same\_outs}$

**same\_lab\_val\_equiv\_thm**  
 $\vdash \forall c \bullet \text{SameLabVal } c \in \text{Equivalence}$

**lift\_rel\_same\_lab\_val\_equiv\_thm**  
 $\vdash \text{LiftRel SameLabVal} \in \text{IndexedEquiv}$

**same\_lab\_val\_anti\_mono\_thm**  
 $\vdash \forall c \ d \bullet c \text{ dominates } d \Rightarrow \text{SameLabVal } c \subseteq \text{SameLabVal } d$

**lift\_rel\_same\_lab\_val\_down\_set\_thm**  
 $\vdash \forall c$   
 •  $\text{LiftRel SameLabVal } (c \downarrow) = \text{SameLabVal } c$

**thm\_040\_2**  $\vdash \forall s$   
 •  $s \in \text{IndexedEquiv}$   
 $\Rightarrow (\forall c \ C \ V$   
 •  $\text{ObservedValue } (c, C, V)$   
 $\in x\_ml\_secure \ s \ (\text{LiftRel SameLabVal})$   
 $\Rightarrow (c, C, V) \in \text{BoundedObs } s)$

**down\_∩\_comp\_up\_thm**  
 $\vdash \forall c$   
 •  $c \downarrow$   
 $= \bigcap$   
 $\{A \mid \exists d \bullet \neg c \text{ dominates } d \wedge A = \sim (d \uparrow)\}$

---

## 34 THE THEORY fef042

### 34.1 Parents

*fef036 wrk057 fef040*

### 34.2 Children

*fef043*

### 34.3 Constants

<b>objectRefers</b>	$Obj \rightarrow Obj\ LIST$
<b>objectContains</b>	$Obj \rightarrow Text$
<b>objectClass</b>	$Obj \rightarrow Class$
<b>MkObj</b>	$Class \times Text \times Obj\ LIST \rightarrow Obj$
<b>reqSsql</b>	$Req \rightarrow Obj$
<b>reqClearance</b>	$Req \rightarrow Class$
<b>MkReq</b>	$Class \rightarrow Obj \rightarrow Req$
<b>identicalObj</b>	$Class \rightarrow Obj \leftrightarrow Obj$
<b>identicalObjs</b>	$Class \rightarrow Obj\ LIST \leftrightarrow Obj\ LIST$
<b>VisibleReq</b>	$Class \rightarrow Req\ \mathbb{P}$
<b>sameRequest</b>	$Class \rightarrow Req \leftrightarrow Req$
<b>sameRequests</b>	$Class \rightarrow Req\ LIST \leftrightarrow Req\ LIST$
<b>VisibleOutput</b>	$Class \rightarrow Obj\ \mathbb{P}$
<b>sameOutputs</b>	$Class \rightarrow Obj\ LIST \leftrightarrow Obj\ LIST$
<b>SWORD_ml_secure</b>	$ML\_BEHAVIOUR\ \mathbb{P}$
<b>Init</b>	$'State\ Machine \rightarrow 'State$
<b>Output</b>	$'State\ Machine \rightarrow 'State \times Req \rightarrow Obj$
<b>Next</b>	$'State\ Machine \rightarrow 'State \times Req \rightarrow 'State$
<b>MkMachine</b>	$('State \times Req \rightarrow 'State)$ $\rightarrow ('State \times Req \rightarrow Obj)$ $\rightarrow 'State$ $\rightarrow 'State\ Machine$
<b>lift_machine</b>	$'State\ Machine \rightarrow 'State \times Req\ LIST \rightarrow Obj\ LIST \times 'State$
<b>Behaviours</b>	$'State\ Machine \rightarrow ML\_BEHAVIOUR$
<b>FilterObj</b>	$Class \rightarrow Obj \rightarrow Obj$
<b>SWORD_construction</b>	$'State\ Machine \rightarrow ML\_BEHAVIOUR$
<b>sameFilterInputs</b>	$Class \rightarrow Obj\ LIST \leftrightarrow Obj\ LIST$
<b>FlowSecureMachine</b>	$'State\ Machine\ \mathbb{P}$

### 34.4 Types

**Req**

'1 Machine

### 34.5 Type Abbreviations

**Text**            *Text***Obj**             *Obj***ML\_BEHAVIOUR**            *ML\_BEHAVIOUR*

### 34.6 Definitions

**MkObj****objectClass****objectContains****objectRefers**     $\vdash$  *ConstSpec*

$$\begin{aligned}
& (\lambda \\
& \quad (MkObj', objectClass', objectContains', \\
& \quad \quad objectRefers') \\
& \bullet \forall c t os \\
& \bullet MkObj' (c, t, os) = MkTree ((c, t), os) \\
& \quad \wedge objectClass' (MkObj' (c, t, os)) = c \\
& \quad \wedge objectContains' (MkObj' (c, t, os)) = t \\
& \quad \wedge objectRefers' (MkObj' (c, t, os)) = os \\
& (MkObj, objectClass, objectContains, objectRefers)
\end{aligned}$$
**Req** $\vdash \exists f \bullet TypeDefn (\lambda x \bullet true) f$ **MkReq****reqClearance****reqSsql**

$$\begin{aligned}
& \vdash \forall t x1 x2 \\
& \bullet reqClearance (MkReq x1 x2) = x1 \\
& \quad \wedge reqSsql (MkReq x1 x2) = x2 \\
& \quad \wedge MkReq (reqClearance t) (reqSsql t) = t
\end{aligned}$$
**identicalObj** $\vdash$  *ConstSpec*

$$\begin{aligned}
& (\lambda identicalObj' \\
& \bullet \forall c \\
& \bullet identicalObj' c \\
& \quad = \{(o_1, o_2) \\
& \quad \mid \exists c_1 t_1 os_1 c_2 t_2 os_2 \\
& \quad \bullet o_1 = MkObj (c_1, t_1, os_1) \\
& \quad \quad \wedge o_2 = MkObj (c_2, t_2, os_2) \\
& \quad \quad \wedge c_1 = c_2 \\
& \quad \quad \wedge (c \text{ dominates } c_1 \\
& \quad \quad \Rightarrow t_1 = t_2 \\
& \quad \quad \wedge \# os_1 = \# os_2 \\
& \quad \quad \wedge Elems (Combine os_1 os_2) \\
& \quad \quad \subseteq identicalObj' c)\}
\end{aligned}$$
*identicalObj*

**identicalObjs**

$$\vdash \forall c$$

- $identicalObjs\ c$   
 $= \{(s_1, s_2) \mid \# s_1 = \# s_2 \wedge Elems\ (Combine\ s_1\ s_2) \subseteq identicalObj\ c\}$

**VisibleReq**

$$\vdash \forall c$$

- $VisibleReq\ c$   
 $= \{r \mid c\ \text{dominates}\ objectClass\ (reqSql\ r)\}$

**sameRequest**

$$\vdash \forall c$$

- $sameRequest\ c$   
 $= \{(r_1, r_2) \mid (reqSql\ r_1, reqSql\ r_2) \in identicalObj\ c\}$

**sameRequests**

$$\vdash \forall c$$

- $sameRequests\ c$   
 $= \{(rs_1, rs_2) \mid let\ rs_3 = rs_1 \upharpoonright VisibleReq\ c$   
 $in\ let\ rs_4 = rs_2 \upharpoonright VisibleReq\ c$   
 $in\ \# rs_3 = \# rs_4 \wedge Elems\ (Combine\ rs_3\ rs_4) \subseteq sameRequest\ c\}$

**VisibleOutput**

$$\vdash \forall c$$

- $VisibleOutput\ c = \{ob \mid c\ \text{dominates}\ objectClass\ ob\}$

**sameOutputs**

$$\vdash \forall c$$

- $sameOutputs\ c$   
 $= \{(os_1, os_2) \mid os_1 \upharpoonright VisibleOutput\ c = os_2 \upharpoonright VisibleOutput\ c\}$

**SWORD\_ml\_secure**

$$\vdash SWORD\_ml\_secure = ml\_secure\ sameRequests\ sameOutputs$$
**Machine**

$$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f$$
**MkMachine****Next****Output****Init**

$$\vdash \forall t\ x1\ x2\ x3$$

- $Next\ (MkMachine\ x1\ x2\ x3) = x1$   
 $\wedge Output\ (MkMachine\ x1\ x2\ x3) = x2$   
 $\wedge Init\ (MkMachine\ x1\ x2\ x3) = x3$   
 $\wedge MkMachine\ (Next\ t)\ (Output\ t)\ (Init\ t) = t$

**lift\_machine**

$$\vdash ConstSpec$$

- ( $\lambda lift\_machine'$ 
  - $\forall mch\ s\ r\ rl$ 
    - $lift\_machine'\ mch\ (s, []) = ([], s)$   
 $\wedge lift\_machine'\ mch\ (s, Cons\ r\ rl)$   
 $= (let\ out = Output\ mch\ (s, r)$   
 $and\ s' = Next\ mch\ (s, r)$   
 $in\ let\ (outl, final\_state)$   
 $= lift\_machine'\ mch\ (s', rl)$

---

*in (Cons out outl, final\_state)))*

**Behaviours**  $\vdash \forall mch\ rs$

- *Behaviours mch rs*  
= *Fst (lift\_machine mch (Init mch, rs))*

**FilterObj**  $\vdash ConstSpec$

( $\lambda FilterObj'$

- $\forall c\ d\ t\ os$
- *FilterObj' c (MkObj (d, t, os))*  
= (*if c dominates d*  
*then MkObj (d, t, Map (FilterObj' c) os)*  
*else MkObj (d, Arbitrary, Arbitrary)*)

*FilterObj*

**SWORD\_construction**

$\vdash \forall mch$

- *SWORD\_construction mch*  
= (*let sec\_output (st, r)*  
*= FilterObj*  
*(reqClearance r)*  
*(Output mch (st, r))*  
*in let sec\_mch*  
*= MkMachine*  
*(Next mch)*  
*sec\_output*  
*(Init mch) in Behaviours sec\_mch)*

**sameFilterInputs**

$\vdash \forall c$

- *sameFilterInputs c*  
=  $\{os_1, os_2\}$   
*|let os<sub>3</sub> = os<sub>1</sub>  $\uparrow$  VisibleOutput c*  
*in let os<sub>4</sub> = os<sub>2</sub>  $\uparrow$  VisibleOutput c*  
*in # os<sub>3</sub> = # os<sub>4</sub>*  
 *$\wedge$  Elems (Combine os<sub>3</sub> os<sub>4</sub>)*  
 *$\subseteq$  identicalObj c}*

**FlowSecureMachine**

$\vdash FlowSecureMachine$

= {*mch*

|*Behaviours mch*

$\in ml\_secure\ sameRequests\ sameFilterInputs$ }

### 34.7 Theorems

#### mk\_tree\_one\_one\_thm

$\vdash \forall x\ y \bullet MkTree\ x = MkTree\ y \Rightarrow x = y$

#### mk\_tree\_onto\_thm

$\vdash \forall t \bullet \exists x \bullet t = MkTree\ x$

#### MkObj\_consistent

#### objectClass\_consistent

**objectContains\_consistent**  
**objectRefers\_consistent**

$\vdash$  *Consistent*  
( $\lambda$   
  (*MkObj'*, *objectClass'*, *objectContains'*,  
  *objectRefers'*)  
  •  $\forall c t os$   
  • *MkObj'* (*c*, *t*, *os*) = *MkTree* ((*c*, *t*), *os*)  
     $\wedge$  *objectClass'* (*MkObj'* (*c*, *t*, *os*)) = *c*  
     $\wedge$  *objectContains'* (*MkObj'* (*c*, *t*, *os*)) = *t*  
     $\wedge$  *objectRefers'* (*MkObj'* (*c*, *t*, *os*)) = *os*)

**obj\_prim\_rec\_thm**

$\vdash \forall d$   
  •  $\exists_1 h$   
  •  $\forall c t os$  • *h* (*MkObj* (*c*, *t*, *os*)) = *d c t* (*Map h os*)

**object\_clauses**

$\vdash (\forall x$   
  • *MkObj* *x*  
    = *MkTree* ((*Fst x*, *Fst (Snd x)*), *Snd (Snd x)*)  
   $\wedge (\forall ctos$   
  • *objectClass* (*MkTree ctos*) = *Fst (Fst ctos)*  
     $\wedge$  *objectContains* (*MkTree ctos*) = *Snd (Fst ctos)*  
     $\wedge$  *objectRefers* (*MkTree ctos*) = *Snd ctos*)

**lift\_machine\_consistent**

$\vdash$  *Consistent*  
( $\lambda$  *lift\_machine'*  
  •  $\forall mch s r rl$   
  • *lift\_machine'* *mch* (*s*, []) = ([], *s*)  
     $\wedge$  *lift\_machine'* *mch* (*s*, *Cons r rl*)  
      = (*let out = Output mch (s, r)*  
        *and s' = Next mch (s, r)*  
        *in let (outl, final\_state)*  
          = *lift\_machine' mch (s', rl)*  
          *in (Cons out outl, final\_state)*)

**identicalObj\_consistent**

$\vdash$  *Consistent*  
( $\lambda$  *identicalObj'*  
  •  $\forall c$   
  • *identicalObj'* *c*  
    = {(*o*<sub>1</sub>, *o*<sub>2</sub>)  
    |  $\exists c_1 t_1 os_1 c_2 t_2 os_2$   
      • *o*<sub>1</sub> = *MkObj* (*c*<sub>1</sub>, *t*<sub>1</sub>, *os*<sub>1</sub>)  
         $\wedge$  *o*<sub>2</sub> = *MkObj* (*c*<sub>2</sub>, *t*<sub>2</sub>, *os*<sub>2</sub>)  
         $\wedge$  *c*<sub>1</sub> = *c*<sub>2</sub>  
         $\wedge$  (*c* *dominates* *c*<sub>1</sub>)  
         $\Rightarrow$  *t*<sub>1</sub> = *t*<sub>2</sub>  
         $\wedge$  # *os*<sub>1</sub> = # *os*<sub>2</sub>  
         $\wedge$  *Elms* (*Combine os*<sub>1</sub> *os*<sub>2</sub>)

$$\subseteq \text{identicalObj}' c\}})$$
**FilterObj\_consistent**

$$\begin{aligned} &\vdash \text{Consistent} \\ &(\lambda \text{FilterObj}' \\ &\bullet \forall c d t \text{ os} \\ &\bullet \text{FilterObj}' c (\text{MkObj } (d, t, \text{os})) \\ &= (\text{if } c \text{ dominates } d \\ &\text{then } \text{MkObj } (d, t, \text{Map } (\text{FilterObj}' c) \text{ os}) \\ &\text{else } \text{MkObj } (d, \text{Arbitrary}, \text{Arbitrary})) \end{aligned}$$
**elems\_combine\_thm**

$$\begin{aligned} &\vdash \forall l x y \\ &\bullet (x, y) \in \text{Elems } (\text{Combine } l) \Leftrightarrow y \in \text{Elems } l \wedge x = y \end{aligned}$$
**elems\_combine\_elems\_thm**

$$\begin{aligned} &\vdash \forall l1 l2 x y \\ &\bullet \# l1 = \# l2 \wedge (x, y) \in \text{Elems } (\text{Combine } l1 l2) \\ &\Rightarrow x \in \text{Elems } l1 \end{aligned}$$
**elems\_combine\_swap\_thm**

$$\begin{aligned} &\vdash \forall l1 l2 x1 x2 \\ &\bullet \# l1 = \# l2 \wedge (x1, x2) \in \text{Elems } (\text{Combine } l1 l2) \\ &\Rightarrow (x2, x1) \in \text{Elems } (\text{Combine } l2 l1) \end{aligned}$$
**elems\_combine\_map\_thm**

$$\begin{aligned} &\vdash \forall f l1 l2 x1 x2 \\ &\bullet \# l1 = \# l2 \\ &\wedge (x1, x2) \in \text{Elems } (\text{Combine } l1 l2) \\ &\wedge \text{Map } f l1 = \text{Map } f l2 \\ &\Rightarrow f x1 = f x2 \end{aligned}$$
**elems\_combine\_map\_thm1**

$$\begin{aligned} &\vdash \forall f l1 l2 \\ &\bullet \# l1 = \# l2 \\ &\wedge (\forall y1 y2 \\ &\bullet (y1, y2) \in \text{Elems } (\text{Combine } l1 l2) \\ &\Rightarrow f y1 = f y2) \\ &\Rightarrow \text{Map } f l1 = \text{Map } f l2 \end{aligned}$$
**obj\_induction\_thm**

$$\begin{aligned} &\vdash \forall P \\ &\bullet (\forall c t ts \\ &\bullet (\forall t \bullet t \in \text{Elems } ts \Rightarrow P t) \\ &\Rightarrow P (\text{MkObj } (c, t, ts))) \\ &\Rightarrow (\forall t \bullet P t) \end{aligned}$$
**identical\_obj\_filter\_obj\_thm1**

$$\begin{aligned} &\vdash \forall c ob1 ob2 \\ &\bullet (ob1, ob2) \in \text{identicalObj } c \\ &\Rightarrow \text{FilterObj } c ob1 = \text{FilterObj } c ob2 \end{aligned}$$
**identical\_obj\_filter\_obj\_thm2**

$$\begin{aligned} &\vdash \forall c ob1 ob2 \\ &\bullet \text{FilterObj } c ob1 = \text{FilterObj } c ob2 \\ &\Rightarrow (ob1, ob2) \in \text{identicalObj } c \end{aligned}$$
**identical\_obj\_filter\_obj\_thm**

$$\vdash \forall c$$

- $identicalObj\ c$   
 $= \{(ob1, ob2) \mid FilterObj\ c\ ob1 = FilterObj\ c\ ob2\}$

**identical\_obj\_refl\_thm**

$$\vdash \forall c\ ob \bullet (ob, ob) \in identicalObj\ c$$

**identical\_obj\_sym\_thm**

$$\vdash \forall c\ ob1\ ob2$$

- $(ob1, ob2) \in identicalObj\ c$   
 $\Rightarrow (ob2, ob1) \in identicalObj\ c$

**identical\_obj\_trans\_thm**

$$\vdash \forall c\ ob1\ ob2\ ob3$$

- $(ob1, ob2) \in identicalObj\ c$   
 $\wedge (ob2, ob3) \in identicalObj\ c$   
 $\Rightarrow (ob1, ob3) \in identicalObj\ c$

**identical\_obj\_rft\_thm**

$$\vdash \forall c$$

- $identicalObj\ c \in Reflexive$   
 $\wedge identicalObj\ c \in Symmetric$   
 $\wedge identicalObj\ c \in Transitive$

**identical\_obj\_equiv\_thm**

$$\vdash \forall c \bullet identicalObj\ c \in Equivalence$$

**same\_requests\_filter\_obj\_thm**

$$\vdash \forall c$$

- $sameRequests\ c$   
 $= \{(rs1, rs2) \mid$   
 $Map\ (FilterObj\ c\ o\ reqSsql)\ (rs1 \upharpoonright VisibleReq\ c)$   
 $= Map$   
 $(FilterObj\ c\ o\ reqSsql)$   
 $(rs2 \upharpoonright VisibleReq\ c)\}$

**same\_requests\_equiv\_thm**

$$\vdash \forall c \bullet sameRequests\ c \in Equivalence$$

**same\_outputs\_equiv\_thm**

$$\vdash \forall c \bullet sameOutputs\ c \in Equivalence$$

**same\_requests\_indexed\_equiv\_thm**

$$\vdash LiftRel\ sameRequests \in IndexedEquiv$$

**same\_outputs\_indexed\_equiv\_thm**

$$\vdash LiftRel\ sameOutputs \in IndexedEquiv$$

## 35 THE THEORY fef043

### 35.1 Parents

*fef042*

---

## 35.2 Constants

$\$f$   $'State\ Factor \rightarrow 'State\ Factor \rightarrow 'State\ Factor$   
**same\_to\_level**  $Worth \rightarrow Obj \leftrightarrow Obj$   
**factor\_level**  $Worth \rightarrow 'State\ Factor\ \mathbb{P}$   
**factor3**  $'State\ Factorisation \rightarrow 'State\ Factor$   
**factor2**  $'State\ Factorisation \rightarrow 'State\ Factor$   
**factor1**  $'State\ Factorisation \rightarrow 'State\ Factor$   
**factor0**  $'State\ Factorisation \rightarrow 'State\ Factor$   
**MkFactorisation**  
 $'State\ Factor$   
 $\rightarrow 'State\ Factor$   
 $\rightarrow 'State\ Factor$   
 $\rightarrow 'State\ Factor$   
 $\rightarrow 'State\ Factorisation$   
**composite**  $'State\ Factorisation \rightarrow 'State\ Factor$   
**factor\_out**  $'State\ Factorisation \rightarrow 'State \times Req \rightarrow Obj$   
**levelled\_factorisation**  
 $'State\ Factorisation\ \mathbb{P}$   
**factors**  $'State\ FactoredMachine \rightarrow 'State\ Factorisation$   
**machine**  $'State\ FactoredMachine \rightarrow 'State\ Machine$   
**MkFactoredMachine**  
 $'State\ Machine$   
 $\rightarrow 'State\ Factorisation$   
 $\rightarrow 'State\ FactoredMachine$   
**well\_factored**  
 $'State\ FactoredMachine\ \mathbb{P}$   
**special\_machine**  
 $'State\ Machine \times 'State\ Factor \times 'State\ Factor$   
 $\rightarrow Req\ LIST$   
 $\rightarrow Req$   
 $\rightarrow Req\ LIST$   
 $\rightarrow Obj$   
**same\_at\_c\_below\_level**  
 $Worth \rightarrow Class \rightarrow Obj \leftrightarrow Obj$   
**label\_secure\_to**  
 $Worth$   
 $\rightarrow 'State\ Machine \leftrightarrow ('State\ Factor \times 'State\ Factor)$   
**label\_secure**  $'State\ FactoredMachine\ \mathbb{P}$   
**simplest\_witness**  
 $'State\ FactoredMachine$   
**identity\_witness**  
 $'State\ FactoredMachine$   
**purge\_above\_level**  
 $Worth \rightarrow Class \rightarrow Obj \rightarrow Obj$

### 35.3 Types

'1 Factorisation

'1 FactoredMachine

### 35.4 Type Abbreviations

'State Factor

'State Factor

### 35.5 Fixity

Right Infix 250:

%f

### 35.6 Definitions

%f  $\vdash \forall f1\ f2\ c\ s\ obj$   
•  $(f1\ \%f\ f2)\ c\ obj\ s = f2\ c\ (f1\ c\ obj\ s)\ s$

same\_to\_level

$\vdash \forall n\ obj1\ obj2$   
• *same\_to\_level 0*  
=  $\{(obj1, obj2)\}$   
| *objectClass obj1 = objectClass obj2*  
 $\wedge$  *same\_to\_level (n + 1)*  
=  $\{(obj1, obj2)\}$   
| *objectContains obj1 = objectContains obj2*  
 $\wedge$  *objectClass obj1 = objectClass obj2*  
 $\wedge$   $\#(objectRefers\ obj1)$   
=  $\#(objectRefers\ obj2)$   
 $\wedge$  *Elms*  
(*Combine*  
  (*objectRefers obj1*)  
  (*objectRefers obj2*))  
 $\subseteq$  *same\_to\_level n*)

factor\_level

$\vdash \forall n$   
• *factor\_level n*  
=  $\{factor$   
  |  $\forall c\ state\ obj$   
  •  $(obj, factor\ c\ obj\ state) \in same\_to\_level\ n\}$

Factorisation

$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f$

MkFactorisation

factor0

factor1

factor2

factor3

$\vdash \forall t\ x1\ x2\ x3\ x4$

---

- $factor0 (MkFactorisation\ x1\ x2\ x3\ x4) = x1$
- $\wedge factor1 (MkFactorisation\ x1\ x2\ x3\ x4) = x2$
- $\wedge factor2 (MkFactorisation\ x1\ x2\ x3\ x4) = x3$
- $\wedge factor3 (MkFactorisation\ x1\ x2\ x3\ x4) = x4$
- $\wedge MkFactorisation$
- $(factor0\ t)$
- $(factor1\ t)$
- $(factor2\ t)$
- $(factor3\ t)$
- $= t$

**composite**     $\vdash \forall f$

- $composite\ f$
- $= factor0\ f$
- $\%_f\ factor1\ f$
- $\%_f\ factor2\ f$
- $\%_f\ factor3\ f$

**factor\_out**     $\vdash \forall f\ s\ r$

- $factor\_out\ f\ (s,\ r)$
- $= composite\ f\ (reqClearance\ r)\ (reqSsql\ r)\ s$

**levelled\_factorisation**

$\vdash \forall facts$ 

- $facts \in levelled\_factorisation$
- $\Leftrightarrow factor0\ facts \in factor\_level\ 0$
- $\wedge factor1\ facts \in factor\_level\ 1$
- $\wedge factor2\ facts \in factor\_level\ 2$
- $\wedge factor3\ facts \in factor\_level\ 3$

**FactoredMachine**

$\vdash \exists f \bullet TypeDefn\ (\lambda x \bullet true)\ f$

**MkFactoredMachine**

**machine**

**factors**     $\vdash \forall t\ x1\ x2$

- $machine (MkFactoredMachine\ x1\ x2) = x1$
- $\wedge factors (MkFactoredMachine\ x1\ x2) = x2$
- $\wedge MkFactoredMachine (machine\ t)\ (factors\ t) = t$

**well\_factored**

$\vdash \forall m$

- $m \in well\_factored$
- $\Leftrightarrow factors\ m \in levelled\_factorisation$
- $\wedge Output (machine\ m) = factor\_out (factors\ m)$

**special\_machine**

$\vdash \forall (m,\ f1,\ f2)\ rl1\ rl2\ r$

- $special\_machine (m,\ f1,\ f2)\ rl1\ r\ rl2$
- $= (let\ s1 = Snd (lift\_machine\ m (Init\ m,\ rl1))$
- $and\ s2 = Snd (lift\_machine\ m (Init\ m,\ rl2))$
- $in\ let\ pe\_req$
- $= f1 (reqClearance\ r)\ (reqSsql\ r)\ s1$
- $in\ f2 (reqClearance\ r)\ pe\_req\ s2)$

**same\_at\_c\_below\_level**

---

$$\begin{aligned} &\vdash \forall n \ c \ obj1 \ obj2 \\ &\bullet \ same\_at\_c\_below\_level \ 0 \ c = identicalObj \ c \\ &\quad \wedge \ same\_at\_c\_below\_level \ (n + 1) \ c \\ &\quad = \{(obj1, obj2)\} \\ &\quad | \# (objectRefers \ obj1) = \# (objectRefers \ obj2) \\ &\quad \wedge \ Elems \\ &\quad \quad (Combine \\ &\quad \quad \quad (objectRefers \ obj1) \\ &\quad \quad \quad (objectRefers \ obj2)) \\ &\quad \subseteq \ same\_at\_c\_below\_level \ n \ c \} \end{aligned}$$
**label\_secure\_to**

$$\begin{aligned} &\vdash \forall n \\ &\bullet \ label\_secure\_to \ n \\ &\quad = \{mff \\ &\quad \quad | \forall \ rl \ r \\ &\quad \quad \bullet (let \ sm = special\_machine \ mff \ rl \ r \\ &\quad \quad \quad in \ sm \\ &\quad \quad \quad \in \ ml\_secure \\ &\quad \quad \quad \quad sameRequests \\ &\quad \quad \quad \quad (same\_at\_c\_below\_level \ n))\} \end{aligned}$$
**label\_secure**

$$\begin{aligned} &\vdash \ label\_secure \\ &\quad = \{fm \\ &\quad \quad | let \ m = machine \ fm \ and \ fs = factors \ fm \\ &\quad \quad \quad in \ let \ lf1 = factor0 \ fs \\ &\quad \quad \quad \quad and \ lf2 = factor0 \ fs \ \%_f \ factor1 \ fs \\ &\quad \quad \quad \quad and \ lf3 \\ &\quad \quad \quad \quad = factor0 \ fs \\ &\quad \quad \quad \quad \quad \%_f \ factor1 \ fs \\ &\quad \quad \quad \quad \quad \%_f \ factor2 \ fs \\ &\quad \quad \quad and \ rf1 \\ &\quad \quad \quad = factor1 \ fs \\ &\quad \quad \quad \quad \%_f \ factor2 \ fs \\ &\quad \quad \quad \quad \%_f \ factor3 \ fs \\ &\quad \quad \quad and \ rf2 = factor2 \ fs \ \%_f \ factor3 \ fs \\ &\quad \quad \quad and \ rf3 = factor3 \ fs \\ &\quad \quad \quad in \ (m, lf1, rf1) \in \ label\_secure\_to \ 1 \\ &\quad \quad \quad \quad \wedge \ (m, lf2, rf2) \in \ label\_secure\_to \ 2 \\ &\quad \quad \quad \quad \wedge \ (m, lf3, rf3) \in \ label\_secure\_to \ 3 \} \end{aligned}$$
**simplest\_witness**

$$\begin{aligned} &\vdash \ simplest\_witness \\ &\quad = (let \ next \ sr = Arbitrary \\ &\quad \quad and \ output \ sr = Arbitrary \\ &\quad \quad and \ init = Arbitrary \\ &\quad \quad and \ f0 \ c \ ob \ s = Arbitrary \\ &\quad \quad in \ let \ (f1, f2, f3) = (f0, f0, f0) \\ &\quad \quad \quad in \ let \ mach = MkMachine \ next \ output \ init \\ &\quad \quad \quad \quad and \ facs = MkFactorisation \ f0 \ f1 \ f2 \ f3 \\ &\quad \quad \quad \quad in \ MkFactoredMachine \ mach \ facs) \end{aligned}$$

**identity\_witness**

```

⊢ identity_witness
  = (let next = Fst
      and output = reqSql o Snd
      and init = Arbitrary
      and f0 c ob s = ob
      in let (f1, f2, f3) = (f0, f0, f0)
          in let mach = MkMachine next output init
              and facs = MkFactorisation f0 f1 f2 f3
              in MkFactoredMachine mach facs)

```

**purge\_above\_level**

```

⊢ ConstSpec
  (λ purge_above_level'
    • (∀ c ob
      • purge_above_level' 0 c ob = FilterObj c ob)
      ∧ (∀ n c d t os
      • purge_above_level'
        (n + 1)
          c
          (MkObj (d, t, os))
            = MkObj
              (Arbitrary, Arbitrary,
                Map (purge_above_level' n c) os)))
    purge_above_level

```

**35.7 Theorems****purge\_above\_level\_consistent**

```

⊢ Consistent
  (λ purge_above_level'
    • (∀ c ob
      • purge_above_level' 0 c ob = FilterObj c ob)
      ∧ (∀ n c d t os
      • purge_above_level'
        (n + 1)
          c
          (MkObj (d, t, os))
            = MkObj
              (Arbitrary, Arbitrary,
                Map (purge_above_level' n c) os)))

```

**purge\_above\_level\_thm**

```

⊢ ∀ n c ob
  • purge_above_level (n + 1) c ob
    = MkTree
      ((Arbitrary, Arbitrary),
        Map
          (purge_above_level n c)
            (objectRefers ob))

```

**same\_at\_c\_below\_level\_purge\_above\_level\_thm1**

$$\begin{aligned} &\vdash \forall n \ c \ ob1 \ ob2 \\ &\bullet (ob1, ob2) \in \text{same\_at\_c\_below\_level } n \ c \\ &\quad \Rightarrow \text{purge\_above\_level } n \ c \ ob1 \\ &\quad = \text{purge\_above\_level } n \ c \ ob2 \end{aligned}$$
**same\_at\_c\_below\_level\_purge\_above\_level\_thm2**

$$\begin{aligned} &\vdash \forall n \ c \ ob1 \ ob2 \\ &\bullet \text{purge\_above\_level } n \ c \ ob1 \\ &\quad = \text{purge\_above\_level } n \ c \ ob2 \\ &\quad \Rightarrow (ob1, ob2) \in \text{same\_at\_c\_below\_level } n \ c \end{aligned}$$
**same\_at\_c\_below\_level\_purge\_above\_level\_thm**

$$\begin{aligned} &\vdash \forall n \ c \ ob1 \ ob2 \\ &\bullet (ob1, ob2) \in \text{same\_at\_c\_below\_level } n \ c \\ &\quad \Leftrightarrow \text{purge\_above\_level } n \ c \ ob1 \\ &\quad = \text{purge\_above\_level } n \ c \ ob2 \end{aligned}$$
**simplest\_witness\_thm**

$$\begin{aligned} &\vdash \text{machine simplest\_witness} \\ &\quad = \text{MkMachine} \\ &\quad (\lambda \ sr \bullet \text{Arbitrary}) \\ &\quad (\lambda \ sr \bullet \text{Arbitrary}) \\ &\quad \text{Arbitrary} \\ &\wedge \text{factors simplest\_witness} \\ &\quad = \text{MkFactorisation} \\ &\quad (\lambda \ c \ ob \ s \bullet \text{Arbitrary}) \\ &\quad (\lambda \ c \ ob \ s \bullet \text{Arbitrary}) \\ &\quad (\lambda \ c \ ob \ s \bullet \text{Arbitrary}) \\ &\quad (\lambda \ c \ ob \ s \bullet \text{Arbitrary}) \end{aligned}$$
**identity\_witness\_thm**

$$\begin{aligned} &\vdash \text{machine identity\_witness} \\ &\quad = \text{MkMachine Fst (reqSsql o Snd) Arbitrary} \\ &\wedge \text{factors identity\_witness} \\ &\quad = \text{MkFactorisation} \\ &\quad (\lambda \ c \ ob \ s \bullet ob) \\ &\quad (\lambda \ c \ ob \ s \bullet ob) \\ &\quad (\lambda \ c \ ob \ s \bullet ob) \\ &\quad (\lambda \ c \ ob \ s \bullet ob) \end{aligned}$$
**same\_at\_c\_below\_level\_equiv\_thm**

$$\vdash \forall n \ c \bullet \text{same\_at\_c\_below\_level } n \ c \in \text{Equivalence}$$
**lift\_rel\_same\_at\_c\_below\_level\_indexed\_equiv\_thm**

$$\vdash \forall n \bullet \text{LiftRel (same\_at\_c\_below\_level } n) \in \text{IndexedEquiv}$$
**lift\_rel\_same\_at\_c\_below\_level\_refl\_thm**

$$\vdash \forall n \ C \ x \bullet (x, x) \in \text{LiftRel (same\_at\_c\_below\_level } n) \ C$$
**simplest\_witness\_label\_secure\_thm**

$$\vdash \text{simplest\_witness} \in \text{label\_secure}$$
**identity\_witness\_label\_secure\_thm**

$$\vdash \text{identity\_witness} \in \text{label\_secure}$$

## 36 THE THEORY `fin_set`

### 36.1 Parents

*seq*

### 36.2 Children

*fin\_thms*

### 36.3 Constants

<code>N</code>	$\mathbb{N} \mathbb{P}$
<code>Finite</code>	$'a \mathbb{P} \mathbb{P}$
<code>F</code>	$'a \mathbb{P} \rightarrow 'a \mathbb{P} \mathbb{P}$
<code>F<sub>1</sub></code>	$'a \mathbb{P} \rightarrow 'a \mathbb{P} \mathbb{P}$
<code>Min</code>	$\mathbb{N} \mathbb{P} \rightarrow \mathbb{N}$
<code>Max</code>	$\mathbb{N} \mathbb{P} \rightarrow \mathbb{N}$
<code>Size</code>	$'a \mathbb{P} \rightarrow \mathbb{N}$
<code>Iter</code>	$\mathbb{N} \rightarrow 'a \leftrightarrow 'a \rightarrow 'a \leftrightarrow 'a$
<code>\$<math>\leftrightarrow</math></code>	$'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$
<code>\$<math>\leftrightarrow</math></code>	$'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$

### 36.4 Aliases

`#`  $Size : 'a \mathbb{P} \rightarrow \mathbb{N}$

### 36.5 Fixity

*Right Infix 240:*

`$\leftrightarrow$`   `$\leftrightarrow$`

### 36.6 Definitions

<code>N</code>	$\vdash \mathbb{N} = Universe$
<code>Finite</code>	$\vdash Finite = \bigcap \{u   \{\} \in u \wedge (\forall a x \bullet a \in u \Rightarrow \{x\} \cup a \in u)\}$
<code>F</code>	$\vdash \forall x \bullet F x = \mathbb{P} x \cap Finite$
<code>F<sub>1</sub></code>	$\vdash \forall x \bullet F_1 x = F x \setminus \{\{\}\}$
<code>Min</code>	$\vdash ConstSpec$ $(\lambda Min'$ $\bullet \forall m a$ $\bullet m \in a \wedge (\forall i \bullet i \in a \Rightarrow m \leq i) \Rightarrow Min' a = m)$
<code>Max</code>	$\vdash ConstSpec$ $(\lambda Max'$ $\bullet \forall m a$

---

	<ul style="list-style-type: none"> <li>• <math>m \in a \wedge (\forall i \bullet i \in a \Rightarrow i \leq m) \Rightarrow \text{Max}' a = m</math></li> </ul>
	<i>Max</i>
<b>Size</b>	$\vdash \forall a$
	<ul style="list-style-type: none"> <li>• <math>\# a = \text{Min} \{i \mid \exists \text{list} \bullet \# \text{list} = i \wedge \text{Elems list} = a\}</math></li> </ul>
<b>Iter</b>	$\vdash \forall r n$
	<ul style="list-style-type: none"> <li>• <i>Iter 0 r = Id Universe</i></li> <li>• <math>\wedge \text{Iter } (n + 1) r = r \circ \text{Iter } n r</math></li> </ul>
$\Rightarrow$	$\vdash \forall a b \bullet a \Rightarrow b = (a \Rightarrow b) \cap \text{Finite}$
$\Rightarrow\Rightarrow$	$\vdash \forall a b \bullet a \Rightarrow\Rightarrow b = (a \Rightarrow\Rightarrow b) \cap (a \Rightarrow\Rightarrow b)$

### 36.7 Theorems

#### Min\_consistent

	$\vdash \text{Consistent}$
	$(\lambda \text{Min}'$
	<ul style="list-style-type: none"> <li>• <math>\forall m a</math></li> <li>• <math>m \in a \wedge (\forall i \bullet i \in a \Rightarrow m \leq i) \Rightarrow \text{Min}' a = m</math></li> </ul>

#### Max\_consistent

	$\vdash \text{Consistent}$
	$(\lambda \text{Max}'$
	<ul style="list-style-type: none"> <li>• <math>\forall m a</math></li> <li>• <math>m \in a \wedge (\forall i \bullet i \in a \Rightarrow i \leq m) \Rightarrow \text{Max}' a = m</math></li> </ul>

#### finite\_induction\_thm

	$\vdash \forall p$
	<ul style="list-style-type: none"> <li>• <math>p \{\}</math></li> <li>• <math>\wedge (\forall a x</math></li> <li>• <math>a \in \text{Finite} \wedge p a \wedge \neg x \in a \Rightarrow p (\{x\} \cup a)</math></li> <li><math>\Rightarrow (\forall a \bullet a \in \text{Finite} \Rightarrow p a)</math></li> </ul>

#### empty\_finite\_thm

	$\vdash \{\} \in \text{Finite}$
--	---------------------------------

#### singleton\_U\_finite\_thm

	$\vdash \forall a x \bullet a \in \text{Finite} \Rightarrow \{x\} \cup a \in \text{Finite}$
--	---

#### subseteq\_finite\_thm

	$\vdash \forall a b \bullet a \in \text{Finite} \wedge b \subseteq a \Rightarrow b \in \text{Finite}$
--	---

#### U\_finite\_thm

	$\vdash \forall a b \bullet a \cup b \in \text{Finite} \Leftrightarrow a \in \text{Finite} \wedge b \in \text{Finite}$
--	--

#### cap\_finite\_thm

	$\vdash \forall a b \bullet a \in \text{Finite} \vee b \in \text{Finite} \Rightarrow a \cap b \in \text{Finite}$
--	--

#### finite\_distinct\_elems\_thm

	$\vdash \forall a$
	<ul style="list-style-type: none"> <li>• <math>a \in \text{Finite}</math></li> <li><math>\Rightarrow (\exists \text{list} \bullet \text{list} \in \text{Distinct} \wedge \text{Elems list} = a)</math></li> </ul>

#### length\_|\_le\_thm

	$\vdash \forall \text{list } a \bullet \# (\text{list} \upharpoonright a) \leq \# \text{list}$
--	--

#### length\_|\_less\_thm

	$\vdash \forall \text{list } a$
	<ul style="list-style-type: none"> <li>• <math>\neg \text{Elems list} \setminus a = \{\} \Rightarrow \# (\text{list} \upharpoonright a) &lt; \# \text{list}</math></li> </ul>

#### elems\_|\_thm

	$\vdash \forall \text{list } a \bullet \text{Elems} (\text{list} \upharpoonright a) = \text{Elems list} \cap a$
--	---

#### distinct\_length\_le\_thm

	$\vdash \forall \text{list1 list2}$
	<ul style="list-style-type: none"> <li>• <math>\text{list1} \in \text{Distinct} \wedge \text{Elems list1} = \text{Elems list2}</math></li> </ul>

$$\Rightarrow \# list1 \leq \# list2$$

**distinct\_size\_length\_thm**

$$\vdash \forall list a$$

$$\bullet list \in Distinct \wedge Elems list = a \Rightarrow \# a = \# list$$

**size\_empty\_thm**

$$\vdash \# \{\} = 0$$

**size\_singleton\_Union\_thm**

$$\vdash \forall x a \bullet a \in Finite \wedge \neg x \in a \Rightarrow \# (\{x\} \cup a) = \# a + 1$$

**size\_singleton\_thm**

$$\vdash \forall x \bullet \# \{x\} = 1$$

**size\_Union\_thm**

$$\vdash \forall a b$$

$$\bullet a \in Finite \wedge b \in Finite$$

$$\Rightarrow \# (a \cup b) + \# (a \cap b) = \# a + \# b$$

**size\_0\_thm**

$$\vdash \forall a \bullet a \in Finite \Rightarrow (\# a = 0 \Leftrightarrow a = \{\})$$

**size\_1\_thm**

$$\vdash \forall a \bullet a \in Finite \Rightarrow (\# a = 1 \Leftrightarrow (\exists x \bullet a = \{x\}))$$

**Union\_finite\_thm**

$$\vdash \forall u \bullet u \in Finite \wedge u \subseteq Finite \Rightarrow \bigcup u \in Finite$$

**pigeon\_hole\_thm**

$$\vdash \forall u$$

$$\bullet u \in Finite \wedge u \subseteq Finite \wedge \# u < \# (\bigcup u)$$

$$\Rightarrow (\exists a \bullet a \in u \wedge \# a > 1)$$

**subseteq\_size\_lessthm**

$$\vdash \forall a b \bullet a \in Finite \wedge b \subseteq a \Rightarrow \# b \leq \# a$$

**subseteq\_size\_less\_thm**

$$\vdash \forall a b \bullet a \in Finite \wedge b \subseteq a \wedge \neg b = a \Rightarrow \# b < \# a$$

**min\_in\_thm**

$$\vdash \forall n a \bullet n \in a \Rightarrow Min a \in a$$

**min\_lessthm**

$$\vdash \forall n a \bullet n \in a \Rightarrow Min a \leq n$$

**max\_in\_thm**

$$\vdash \forall m n a \bullet (\forall i \bullet i \in a \Rightarrow i \leq m) \wedge n \in a \Rightarrow Max a \in a$$

**lessthmax\_thm**

$$\vdash \forall m n a \bullet (\forall i \bullet i \in a \Rightarrow i \leq m) \wedge n \in a \Rightarrow n \leq Max a$$

**finite\_subset\_well\_founded\_thm**

$$\vdash \forall p a$$

$$\bullet a \in Finite \wedge p a$$

$$\Rightarrow (\exists b \bullet b \subseteq a \wedge p b \wedge (\forall c \bullet c \subseteq b \wedge p c \Rightarrow c = b))$$

## 37 THE THEORY fin\_thms

### 37.1 Parents

*fin\_set*

### 37.2 Children

*lib\_thms*

**37.3 Theorems****fin\_set\_induction\_thm**

$$\begin{aligned} &\vdash \forall P \\ &\quad \bullet P \{\} \\ &\quad \wedge (\forall a x \\ &\quad \quad \bullet a \in \text{Finite} \wedge P a \wedge \neg x \in a \Rightarrow P (\{x\} \cup a)) \\ &\quad \Rightarrow (\forall a \bullet a \in \text{Finite} \Rightarrow P a) \end{aligned}$$
**min\_thm**

$$\vdash \forall m a \bullet m \in a \Rightarrow \text{Min } a \in a \wedge \text{Min } a \leq m$$
**min\_clauses**

$$\begin{aligned} &\vdash (\forall m \bullet \text{Min } \{m\} = m) \\ &\quad \wedge (\forall m n \bullet \text{Min } \{m; n\} = (\text{if } m \leq n \text{ then } m \text{ else } n)) \\ &\quad \wedge (\forall m a \\ &\quad \quad \bullet \text{Min } (\{m\} \cup a) \\ &\quad \quad = (\text{if } a = \{\} \text{ then } m \text{ else } \text{Min } \{m; \text{Min } a\})) \end{aligned}$$
**fin\_set\_thm1**

$$\begin{aligned} &\vdash \forall a \\ &\quad \bullet a \in \text{Finite} \\ &\quad \Rightarrow (\exists \text{list} \bullet a = \text{Elems list} \wedge \text{list} \in \text{Distinct}) \end{aligned}$$
**elems\_thm1**

$$\vdash \forall \text{list} \bullet \text{Elems list} = \{\} \Leftrightarrow \text{list} = []$$
**elems\_thm2**

$$\begin{aligned} &\vdash (\forall \text{list} \bullet \text{Elems list} = \{\} \Leftrightarrow \text{list} = []) \\ &\quad \wedge (\forall \text{list} \bullet \{\} = \text{Elems list} \Leftrightarrow \text{list} = []) \end{aligned}$$
**elems\_thm3**

$$\begin{aligned} &\vdash \forall \text{list1 list2} \\ &\quad \bullet \text{Elems } (\text{list1} \hat{\ } \text{list2}) = \text{Elems list1} \cup \text{Elems list2} \end{aligned}$$
**length\_thm**

$$\vdash \forall \text{list1 list2} \bullet \# (\text{list1} \hat{\ } \text{list2}) = \# \text{list1} + \# \text{list2}$$
**|\_thm1**

$$\vdash \forall \text{list } a \bullet \text{Elems } (\text{list} \upharpoonright a) = \text{Elems list} \cap a$$
**|\_thm2**

$$\vdash \forall \text{list } a \bullet \# ((\text{list} \upharpoonright a) \hat{\ } (\text{list} \upharpoonright \sim a)) = \# \text{list}$$
**|\_thm3**

$$\vdash \forall \text{list } a \bullet \text{Elems list} \subseteq a \Rightarrow \text{list} \upharpoonright a = \text{list}$$
**|\_thm4**

$$\vdash \forall \text{list } x \bullet x \in \text{Elems list} \Rightarrow \# (\text{list} \upharpoonright \sim \{x\}) < \# \text{list}$$
**distinct\_thm1**

$$\begin{aligned} &\vdash \forall \text{list1 list2} \\ &\quad \bullet \text{list1} \in \text{Distinct} \wedge \text{Elems list1} = \text{Elems list2} \\ &\quad \Rightarrow \# \text{list1} \leq \# \text{list2} \end{aligned}$$
**size\_thm1**

$$\vdash \# \{\} = 0$$
**size\_thm2**

$$\vdash \forall \text{list} \bullet \text{list} \in \text{Distinct} \Rightarrow \# (\text{Elems list}) = \# \text{list}$$
**fin\_set\_thm2**

$$\vdash \{\} \in \text{Finite}$$
**fin\_set\_thm3**

$$\vdash \forall a x \bullet a \in \text{Finite} \Rightarrow \{x\} \cup a \in \text{Finite}$$
**fin\_set\_thm4**

$$\vdash \forall \text{list} \bullet \text{Elems list} \in \text{Finite}$$
**size\_thm3**

$$\vdash \forall a b \bullet a \in \text{Finite} \wedge b \in \text{Finite} \Rightarrow a \cup b \in \text{Finite}$$
**size\_thm4**

$$\vdash \forall a b \bullet a \in \text{Finite} \wedge b \subseteq a \Rightarrow b \in \text{Finite}$$
**size\_thm5**

$$\begin{aligned} &\vdash \forall a x \\ &\quad \bullet a \in \text{Finite} \\ &\quad \Rightarrow \# (\{x\} \cup a) = (\text{if } x \in a \text{ then } \# a \text{ else } \# a + 1) \end{aligned}$$
**size\_thm6**

$$\vdash \forall a b \bullet a \in \text{Finite} \wedge b \in \text{Finite} \Rightarrow a \cap b \in \text{Finite}$$
**size\_thm7**

$$\begin{aligned} &\vdash \forall a b \\ &\quad \bullet a \in \text{Finite} \wedge b \in \text{Finite} \\ &\quad \Rightarrow \# (a \cup b) + \# (a \cap b) = \# a + \# b \end{aligned}$$
**size\_singleton\_thm**

$$\vdash \forall x \bullet \# \{x\} = 1$$
**N\_set\_thm1**

$$\vdash \forall a$$

- $a \in Finite \wedge \neg a = \{\}$   
 $\Rightarrow (\exists m \bullet m \in a \wedge (\forall i \bullet i \in a \Rightarrow i \leq m))$

**finite\_max\_thm**

- $\vdash \forall a$
- $a \in Finite \wedge \neg a = \{\}$   
 $\Rightarrow Max a \in a \wedge (\forall i \bullet i \in a \Rightarrow i \leq Max a)$

**finite\_size\_thm**

- $\vdash \forall a m$
- $(\exists list$   
  - $Elms list = a \wedge list \in Distinct \wedge \# list = m)$ $\Leftrightarrow a \in Finite \wedge \# a = m$

**length\_map\_thm**

- $\vdash \forall f list \bullet \# (Map f list) = \# list$

**elems\_map\_thm**

- $\vdash \forall f list$
- $Elms (Map f list)$   
 $= \{y \mid \exists x \bullet x \in Elms list \wedge f x = y\}$

**map\_distinct\_thm**

- $\vdash \forall f list$
- $Map f list \in Distinct$   
 $\Leftrightarrow list \in Distinct$   
 $\wedge (\forall x y$   
  - $x \in Elms list \wedge y \in Elms list \wedge f x = f y$   
 $\Rightarrow x = y)$

**pair\_eq\_thm**

- $\vdash \forall x y \bullet Fst x = Fst y \wedge Snd x = Snd y \Rightarrow x = y$

**at\_thm**

- $\vdash \forall f x$
- $f \in Functional \wedge x \in Dom f$   
 $\Rightarrow (\forall y \bullet f @ x = y \Leftrightarrow (x, y) \in f)$

**finite\_function\_thm**

- $\vdash \forall f$
- $f \in Functional \wedge (f \in Finite \vee Dom f \in Finite)$   
 $\Rightarrow f \in Finite \wedge Dom f \in Finite \wedge \# (Dom f) = \# f$

## 38 THE THEORY fun\_rel

### 38.1 Parents

*bin\_rel*

### 38.2 Children

*fun\_rel\_thms seq*

**38.3 Constants**

$\$ \leftrightarrow$              $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \rightarrow$              $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \mapsto$              $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \dashrightarrow$          $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \twoheadrightarrow$          $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \rightarrow$              $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \twoheadrightarrow$          $'a \mathbb{P} \rightarrow 'b \mathbb{P} \rightarrow ('a \leftrightarrow 'b) \mathbb{P}$   
 $\$ \text{At}$              $'a \leftrightarrow 'b \rightarrow 'a \rightarrow 'b$

**38.4 Aliases**

@             $\$ \text{At} : 'a \leftrightarrow 'b \rightarrow 'a \rightarrow 'b$

**38.5 Fixity**

*Left Infix 600:*

**At**

*Right Infix 240:*

$\twoheadrightarrow$      $\twoheadrightarrow$      $\twoheadrightarrow$      $\twoheadrightarrow$      $\twoheadrightarrow$      $\twoheadrightarrow$

**38.6 Definitions**

$\twoheadrightarrow$              $\vdash \forall a b \bullet a \twoheadrightarrow b = (a \leftrightarrow b) \cap \text{Functional}$   
 $\rightarrow$                      $\vdash \forall a b \bullet (a \rightarrow b) = (a \leftrightarrow b) \cap \text{Functional} \cap \text{Total } a$   
 $\mapsto$                    $\vdash \forall a b \bullet a \mapsto b = (a \leftrightarrow b) \cap \text{Functional} \cap \text{Injective}$   
 $\dashrightarrow$                  $\vdash \forall a b$   
                            $\bullet a \dashrightarrow b = (a \leftrightarrow b) \cap \text{Functional} \cap \text{Injective} \cap \text{Total } a$   
 $\twoheadrightarrow$                  $\vdash \forall a b \bullet a \twoheadrightarrow b = (a \leftrightarrow b) \cap \text{Functional} \cap \text{Surjective } b$   
 $\rightarrow$                      $\vdash \forall a b$   
                            $\bullet a \rightarrow b$   
                            $= (a \leftrightarrow b) \cap \text{Functional} \cap \text{Surjective } b \cap \text{Total } a$   
 $\twoheadrightarrow$                  $\vdash \forall a b$   
                            $\bullet a \twoheadrightarrow b$   
                            $= (a \leftrightarrow b)$   
                            $\cap \text{Functional}$   
                            $\cap \text{Injective}$   
                            $\cap \text{Surjective } b$   
                            $\cap \text{Total } a$   
**At**                     $\vdash \text{ConstSpec}$   
                            $(\lambda \text{At}'$   
                            $\bullet \forall f x y$   
                            $\bullet (x, y) \in f \wedge (\forall z \bullet (x, z) \in f \Rightarrow z = y)$   
                            $\Rightarrow \text{At}' f x = y)$   
 $\$ @$

## 38.7 Theorems

### At\_consistent

$$\begin{aligned} &\vdash \text{Consistent} \\ &(\lambda \text{At}' \\ &\bullet \forall f \ x \ y \\ &\bullet (x, y) \in f \wedge (\forall z \bullet (x, z) \in f \Rightarrow z = y) \\ &\quad \Rightarrow \text{At}' \ f \ x = y) \end{aligned}$$

**graph\_at\_thm**  $\vdash \forall f \ x \bullet \text{Graph } f \ @ \ x = f \ x$

### inv\_rel\_∈\_arrow\_thm

$$\begin{aligned} &\vdash \forall f \ a \ b \\ &\bullet (f \sim \in b \Leftrightarrow a \Leftrightarrow f \in a \Leftrightarrow b) \\ &\quad \wedge (f \sim \in b \Rightarrow a \Leftrightarrow f \in (a \Leftrightarrow b) \cap \text{Injective}) \\ &\quad \wedge (f \sim \in (b \rightarrow a) \\ &\quad \Leftrightarrow f \in (a \Leftrightarrow b) \cap \text{Injective} \cap \text{Surjective } b) \\ &\quad \wedge (f \sim \in b \rightsquigarrow a \\ &\quad \Leftrightarrow f \in (a \Leftrightarrow b) \cap \text{Injective} \cap \text{Functional}) \\ &\quad \wedge (f \sim \in b \rightrightarrows a \\ &\quad \Leftrightarrow f \in (a \Leftrightarrow b) \cap \text{Injective} \cap \text{Total } a) \\ &\quad \wedge (f \sim \in b \rightarrow a \\ &\quad \Leftrightarrow f \\ &\quad \quad \in (a \Leftrightarrow b) \\ &\quad \quad \cap \text{Injective} \\ &\quad \quad \cap \text{Total } a \\ &\quad \quad \cap \text{Surjective } b) \\ &\quad \wedge (f \sim \in b \twoheadrightarrow a \\ &\quad \Leftrightarrow f \\ &\quad \quad \in (a \Leftrightarrow b) \\ &\quad \quad \cap \text{Injective} \\ &\quad \quad \cap \text{Functional} \\ &\quad \quad \cap \text{Surjective } b \\ &\quad \quad \cap \text{Total } a) \\ &\quad \wedge (f \sim \in b \mapsto a \\ &\quad \Leftrightarrow f \\ &\quad \quad \in (a \Leftrightarrow b) \\ &\quad \quad \cap \text{Injective} \\ &\quad \quad \cap \text{Functional} \\ &\quad \quad \cap \text{Surjective } b) \end{aligned}$$

**at\_at\_eq\_thm**  $\vdash \forall f \ X \ Y \ x \ y$

$$\begin{aligned} &\bullet f \in X \twoheadrightarrow Y \wedge x \in \text{Dom } f \wedge y \in \text{Dom } f \\ &\quad \Rightarrow (f \ @ \ x = f \ @ \ y \\ &\quad \Leftrightarrow (\exists z \bullet (x, z) \in f \wedge (y, z) \in f)) \end{aligned}$$

## 39 THE THEORY fun\_rel\_thms

### 39.1 Parents

*set\_thms*      *fun\_rel*

### 39.2 Constants

**ReflexiveIn**  $'a \mathbb{P} \rightarrow ('a \leftrightarrow 'a) \mathbb{P}$

**Antisymmetric**

$('a \leftrightarrow 'a) \mathbb{P}$

**Chains**  $'a \leftrightarrow 'a \rightarrow 'a \mathbb{P} \mathbb{P}$

**UpperBounds**  $'a \leftrightarrow 'a \rightarrow 'a \mathbb{P} \rightarrow 'a \mathbb{P}$

**MaximalElements**

$'a \leftrightarrow 'a \rightarrow 'a \mathbb{P} \rightarrow 'a \mathbb{P}$

### 39.3 Definitions

**ReflexiveIn**  $\vdash \forall a \bullet \text{ReflexiveIn } a = \{r \mid \forall x \bullet x \in a \Rightarrow (x, x) \in r\}$

**Antisymmetric**

$\vdash \text{Antisymmetric}$

$= \{r \mid \forall x y \bullet (x, y) \in r \wedge (y, x) \in r \Rightarrow x = y\}$

**Chains**

$\vdash \forall r$

$\bullet \text{Chains } r$

$= \{a$

$\mid \forall x y \bullet x \in a \wedge y \in a \Rightarrow (x, y) \in r \vee (y, x) \in r\}$

**UpperBounds**  $\vdash \forall r a \bullet \text{UpperBounds } r a = \{x \mid \forall y \bullet y \in a \Rightarrow (y, x) \in r\}$

**MaximalElements**

$\vdash \forall r a$

$\bullet \text{MaximalElements } r a$

$= \{x \mid x \in a \wedge (\forall y \bullet y \in a \wedge (x, y) \in r \Rightarrow y = x)\}$

### 39.4 Theorems

**zorn\_thm**

$\vdash \forall r a$

$\bullet r \in \text{ReflexiveIn } a$

$\wedge r \in \text{Transitive}$

$\wedge r \in \text{Antisymmetric}$

$\wedge (\forall c$

$\bullet c \subseteq a \wedge c \in \text{Chains } r$

$\Rightarrow (\exists x \bullet x \in a \wedge x \in \text{UpperBounds } r c))$

$\Rightarrow (\exists x \bullet x \in \text{MaximalElements } r a)$

**comparability\_thm**

$\vdash \forall a b \bullet (\exists f \bullet f \in a \rightsquigarrow b) \vee (\exists g \bullet g \in b \rightsquigarrow a)$

**schroeder\_bernstein\_thm1**

$\vdash \forall a b c f$

$\bullet c \subseteq b \wedge b \subseteq a \wedge f \in a \rightsquigarrow c \Rightarrow (\exists h \bullet h \in a \rightsquigarrow b)$

**schroeder\_bernstein\_thm**

$\vdash \forall a b f g \bullet f \in a \rightsquigarrow b \wedge g \in b \rightsquigarrow a \Rightarrow (\exists h \bullet h \in a \rightsquigarrow b)$

## 40 THE THEORY hol

### 40.1 Parents

*sets sum one*

**40.2 Children***bin\_rel***41 THE THEORY init****41.1 Parents***log***41.2 Children***misc***41.3 Axioms****bool\_cases\_axiom** $\vdash \forall b \bullet (b \Leftrightarrow \text{true}) \vee (b \Leftrightarrow \text{false})$  **$\Rightarrow$ \_antisym\_axiom** $\vdash \forall b1\ b2 \bullet (b1 \Rightarrow b2) \Rightarrow (b2 \Rightarrow b1) \Rightarrow (b1 \Leftrightarrow b2)$  **$\eta$ \_axiom** $\vdash \forall f \bullet (\lambda x \bullet f\ x) = f$  **$\epsilon$ \_axiom** $\vdash \forall P\ x \bullet P\ x \Rightarrow P\ (\$ \epsilon P)$ **infinity\_axiom** $\vdash \exists f \bullet \text{OneOne}\ f \wedge \neg \text{Onto}\ f$ **42 THE THEORY lib\_thms****42.1 Parents***fn\_thms***42.2 Children***wrk057 wrk049*

### 42.3 Theorems

**rel\_ext\_thm**  $\vdash \forall r s \bullet r = s \Leftrightarrow (\forall x y \bullet (x, y) \in r \Leftrightarrow (x, y) \in s)$

**rev\_∧\_thm**  $\vdash \forall list x \bullet Rev (list \wedge [x]) = Cons x (Rev list)$

**rev\_rev\_thm**  $\vdash \forall list \bullet Rev (Rev list) = list$

**rev\_list\_induction\_thm**

$\vdash \forall p$

- $p [] \wedge (\forall list \bullet p list \Rightarrow (\forall x \bullet p (list \wedge [x])))$   
 $\Rightarrow (\forall list \bullet p list)$

**length\_∧\_thm**  $\vdash \forall list1 list2 \bullet \# (list1 \wedge list2) = \# list1 + \# list2$

**list\_rel\_thm**  $\vdash \forall list$

- $ListRel list$   
 $= \{(i, x) | 1 \leq i \wedge i \leq \# list \wedge Nth list i = x\}$

**list\_rel\_cons\_thm**

$\vdash ListRel [] = \{\}$

$\wedge (\forall x list$

- $ListRel (Cons x list)$

$= \{(i, y)$

$|\exists j \bullet j + 1 = i \wedge (j, y) \in ListRel list\}$

$\cup \{(1, x)\}$ )

**list\_rel\_∧\_thm**

$\vdash \forall list1 list2$

- $ListRel (list1 \wedge list2)$

$= ListRel list1$

$\cup \{(i, y)$

$|\exists j \bullet \# list1 + j = i \wedge (j, y) \in ListRel list2\}$

**list\_rel\_singleton\_thm**

$\vdash \forall x \bullet ListRel [x] = \{(1, x)\}$

**list\_rel\_∧\_singleton\_thm**

$\vdash \forall list x$

- $ListRel (list \wedge [x])$

$= ListRel list \cup \{(\# list + 1, x)\}$

**dom\_list\_rel\_thm**

$\vdash \forall list \bullet Dom (ListRel list) = 1 .. \# list$

**dom\_id\_ran\_id\_thm**

$\vdash \forall a \bullet Dom (Id a) = a \wedge Ran (Id a) = a$

**dot\_dot\_size\_thm**

$\vdash \forall i j$

- $i .. j \in Finite$

$\wedge \# (i .. j) = (if j < i then 0 else j - i + 1)$

**enumerate\_thm**

$\vdash \forall a$

- $Enumerate a$

$= \{(m, n) | n \in a \wedge \# \{i | i \in a \wedge i \leq n\} = m\}$

**squash\_id\_thm**

$\vdash \forall a \bullet Squash (Id a) = Enumerate a$

**squash\_thm**

$\vdash \forall r$

---

- *Squash*  $r$   
 $= \{(m, y) \mid \exists n \bullet (n, y) \in r \wedge \# \{i \mid i \in \text{Dom } r \wedge i \leq n\} = m\}$

**U\_▷\_thm**  $\vdash \forall r s a \bullet r \cup s \triangleright a = (r \triangleright a) \cup (s \triangleright a)$   
**dom\_U\_thm**  $\vdash \forall r s \bullet \text{Dom } (r \cup s) = \text{Dom } r \cup \text{Dom } s$   
**enumerate\_U\_thm**  
 $\vdash \forall a b$ 

- $(\forall i j \bullet i \in a \wedge j \in b \Rightarrow i < j)$   
 $\Rightarrow \text{Enumerate } (a \cup b)$   
 $= \text{Enumerate } a$   
 $\cup \{(m, n) \mid n \in b$   
 $\wedge (\exists j$ 
  - $\# a + j = m \wedge (j, n) \in \text{Enumerate } b)\}$

**elems\_set\_comp\_thm**  
 $\vdash \forall list$ 

- *Elems*  $list$   
 $= \{x \mid \exists i \bullet 1 \leq i \wedge i \leq \# list \wedge Nth list i = x\}$

## 43 THE THEORY list

### 43.1 Parents

$\mathbb{N}$

### 43.2 Children

*char*

### 43.3 Constants

<b>IsListRep</b>	$(Worth \rightarrow 'a) \times Worth \rightarrow Bool$
<b>[]</b>	$'a LIST$
<b>Cons</b>	$'a \rightarrow 'a LIST \rightarrow 'a LIST$
<b>Length</b>	$'a LIST \rightarrow Worth$
<b>Hd</b>	$'a LIST \rightarrow 'a$
<b>Tl</b>	$'a LIST \rightarrow 'a LIST$
<b>Append</b>	$'a LIST \rightarrow 'a LIST \rightarrow 'a LIST$
<b>Rev</b>	$'a LIST \rightarrow 'a LIST$
<b>Split</b>	$('a \times 'b) LIST \rightarrow 'a LIST \times 'b LIST$
<b>Combine</b>	$'a LIST \rightarrow 'b LIST \rightarrow ('a \times 'b) LIST$
<b>Map</b>	$('a \rightarrow 'b) \rightarrow 'a LIST \rightarrow 'b LIST$
<b>Fold</b>	$('a \rightarrow 'b \rightarrow 'b) \rightarrow 'a LIST \rightarrow 'b \rightarrow 'b$

### 43.4 Aliases

@  $Append : 'a LIST \rightarrow 'a LIST \rightarrow 'a LIST$

### 43.5 Types

'1 LIST

### 43.6 Fixity

Right Infix 300:

@

### 43.7 Definitions

IsListRep

is\_list\_rep\_def

$$\begin{aligned}
& \vdash \exists \text{ nil} \\
& \quad \text{mLmk\_var}(" \text{cons} ", \ulcorner ('a \\
& \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth} \\
& \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth})^\neg)^\neg \\
& \bullet \text{ IsListRep nil} \\
& \quad \wedge (\forall x \\
& \quad \bullet \text{ IsListRep } x \\
& \quad \quad \Rightarrow (\forall h \\
& \quad \quad \bullet \neg \text{mLmk\_var}(" \text{cons} ", \ulcorner ('a \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth} \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth})^\neg)^\neg \\
& \quad \quad \quad h \\
& \quad \quad \quad x \\
& \quad \quad \quad = \text{nil})) \\
& \quad \wedge (\forall x y \\
& \quad \bullet \text{ IsListRep } x \wedge \text{ IsListRep } y \\
& \quad \quad \Rightarrow (\forall a b \\
& \quad \quad \bullet \text{mLmk\_var}(" \text{cons} ", \ulcorner ('a \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth} \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth})^\neg)^\neg \\
& \quad \quad \quad a \\
& \quad \quad \quad x \\
& \quad \quad \quad = \text{mLmk\_var}(" \text{cons} ", \ulcorner ('a \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth} \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth})^\neg)^\neg \\
& \quad \quad \quad b \\
& \quad \quad \quad y \\
& \quad \quad \quad \Leftrightarrow a = b \wedge x = y)) \\
& \quad \wedge (\forall x \\
& \quad \bullet \text{ IsListRep } x \\
& \quad \quad \Rightarrow (\forall h \\
& \quad \quad \bullet \text{ IsListRep} \\
& \quad \quad \quad (\text{mLmk\_var}(" \text{cons} ", \ulcorner ('a \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth} \\
& \quad \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth})^\neg)^\neg
\end{aligned}$$

$$\begin{aligned}
& h \\
& x))) \\
& \wedge (\forall p \\
& \bullet p \text{ nil} \\
& \wedge (\forall m \\
& \bullet p m \\
& \Rightarrow (\forall h \\
& \bullet p \\
& \quad (\text{mk\_var}(\text{"cons"}, \ulcorner 'a \\
& \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth} \\
& \quad \quad \rightarrow (\text{Worth} \rightarrow 'a) \times \text{Worth})^\neg)^\neg \\
& \quad h \\
& \quad m))) \\
& \Rightarrow (\forall n \bullet \text{IsListRep } n \Rightarrow p n)
\end{aligned}$$

**LIST**list\_def  $\vdash \exists f \bullet \text{TypeDefn } \text{IsListRep } f$ **Nil****Cons**

nil\_cons\_def  $\vdash (\forall x \text{ list} \bullet \neg \text{Cons } x \text{ list} = [])$   
 $\wedge (\forall x1 \ x2 \ \text{list1} \ \text{list2}$   
 $\bullet \text{Cons } x1 \ \text{list1} = \text{Cons } x2 \ \text{list2}$   
 $\Leftrightarrow x1 = x2 \wedge \text{list1} = \text{list2})$   
 $\wedge (\forall p$   
 $\bullet p [] \wedge (\forall \text{list} \bullet p \ \text{list} \Rightarrow (\forall x \bullet p (\text{Cons } x \ \text{list})))$   
 $\Rightarrow (\forall \text{list} \bullet p \ \text{list}))$

**Length**length\_def  $\vdash \# [] = 0 \wedge (\forall h \ \text{list} \bullet \# (\text{Cons } h \ \text{list}) = \# \ \text{list} + 1)$ **Hd**hd\_def  $\vdash \forall h \ \text{list} \bullet \text{Head } (\text{Cons } h \ \text{list}) = h$ **Tl**tl\_def  $\vdash \forall h \ \text{list} \bullet \text{Tail } (\text{Cons } h \ \text{list}) = \text{list}$ **Append**append\_def  $\vdash (\forall \text{list} \bullet [] @ \text{list} = \text{list})$   
 $\wedge (\forall h \ \text{list} \ \text{list}'$   
 $\bullet \text{Cons } h \ \text{list} @ \text{list}' = \text{Cons } h (\text{list} @ \text{list}'))$ **Rev**rev\_def  $\vdash \text{Rev } [] = []$   
 $\wedge (\forall h \ \text{list} \bullet \text{Rev } (\text{Cons } h \ \text{list}) = \text{Rev } \text{list} @ [h])$ **Split**split\_def  $\vdash \text{Split } [] = ([], [])$   
 $\wedge (\forall \text{list} \ h1 \ h2$   
 $\bullet \text{Split } (\text{Cons } (h1, h2) \ \text{list})$   
 $= (\text{Cons } h1 (\text{Fst } (\text{Split } \ \text{list})),$   
 $\text{Cons } h2 (\text{Snd } (\text{Split } \ \text{list})))$ **Combine**combine\_def  $\vdash \text{Combine } [] \ [] = []$   
 $\wedge (\forall h1 \ h2 \ \text{list1} \ \text{list2}$   
 $\bullet \text{Combine } (\text{Cons } h1 \ \text{list1}) (\text{Cons } h2 \ \text{list2})$

---


$$= \text{Cons } (h1, h2) (\text{Combine } list1 \ list2))$$
**Map****map\_def**

$$\vdash (\forall g \bullet \text{Map } g \ [] = [])$$

$$\wedge (\forall h \ g \ list$$

- $\text{Map } g (\text{Cons } h \ list) = \text{Cons } (g \ h) (\text{Map } g \ list))$

**Fold****fold\_def**

$$\vdash (\forall g \ x \bullet \text{Fold } g \ [] \ x = x)$$

$$\wedge (\forall h \ g \ x \ list$$

- $\text{Fold } g (\text{Cons } h \ list) \ x = g \ h (\text{Fold } g \ list \ x)$

**43.8 Theorems****list\_induction\_thm**

$$\vdash \forall p$$

- $p \ [] \wedge (\forall list \bullet p \ list \Rightarrow (\forall x \bullet p (\text{Cons } x \ list)))$
- $\Rightarrow (\forall list \bullet p \ list)$

**list\_prim\_rec\_thm**

$$\vdash \forall n \ c$$

- $\exists_1 f$
- $f \ [] = n$
- $\wedge (\forall list \ a$ 
  - $f (\text{Cons } a \ list) = c \ a (f \ list) \ list)$

**list\_clauses**

$$\vdash \forall x1 \ x2 \ list1 \ list2$$

- $\neg \text{Cons } x1 \ list1 = []$
- $\wedge \neg [] = \text{Cons } x1 \ list1$
- $\wedge (\text{Cons } x1 \ list1 = \text{Cons } x2 \ list2$ 
  - $\Leftrightarrow x1 = x2 \wedge list1 = list2)$
  - $\wedge \text{Head } (\text{Cons } x1 \ list1) = x1$
  - $\wedge \text{Tail } (\text{Cons } x1 \ list1) = list1$

**list\_cases\_thm**

$$\vdash \forall list1$$

- $list1 = [] \vee (\exists x \ list2 \bullet list1 = \text{Cons } x \ list2)$

**44 THE THEORY log****44.1 Parents***min***44.2 Children***init*

### 44.3 Constants

<b>T</b>	<i>Bool</i>
<b>\$V</b>	$( 'a \rightarrow Bool ) \rightarrow Bool$
<b>\$E</b>	$( 'a \rightarrow Bool ) \rightarrow Bool$
<b>F</b>	<i>Bool</i>
<b>\$¬</b>	$Bool \rightarrow Bool$
<b>\$∧</b>	$Bool \rightarrow Bool \rightarrow Bool$
<b>\$∨</b>	$Bool \rightarrow Bool \rightarrow Bool$
<b>OneOne</b>	$( 'a \rightarrow 'b ) \rightarrow Bool$
<b>Onto</b>	$( 'a \rightarrow 'b ) \rightarrow Bool$
<b>TypeDefn</b>	$( 'b \rightarrow Bool ) \rightarrow ( 'a \rightarrow 'b ) \rightarrow Bool$

### 44.4 Fixity

<i>Binder:</i>	∀	∃
<i>Right Infix 30:</i>	∨	
<i>Right Infix 40:</i>	∧	
<i>Prefix 50:</i>	¬	

### 44.5 Terminators

∀    ∃    ¬    ∧    ∨

### 44.6 Definitions

<b>T</b>	
<b>t_def</b>	$\vdash true \Leftrightarrow (\lambda x \bullet x) = (\lambda x \bullet x)$
<b>∨</b>	
<b>∨_def</b>	$\vdash \$\vee = (\lambda P \bullet P = (\lambda x \bullet true))$
<b>∃</b>	
<b>∃_def</b>	$\vdash \$\exists = (\lambda P \bullet P (\$e P))$
<b>F</b>	
<b>f_def</b>	$\vdash false \Leftrightarrow (\forall b \bullet b)$
<b>¬</b>	
<b>¬_def</b>	$\vdash \$\neg = (\lambda b \bullet b \Rightarrow false)$
<b>∧</b>	
<b>∧_def</b>	$\vdash \$\wedge = (\lambda b1 b2 \bullet \forall b \bullet (b1 \Rightarrow b2 \Rightarrow b) \Rightarrow b)$
<b>∨</b>	
<b>∨_def</b>	$\vdash \$\vee = (\lambda b1 b2 \bullet \forall b \bullet (b1 \Rightarrow b) \Rightarrow (b2 \Rightarrow b) \Rightarrow b)$
<b>OneOne</b>	
<b>one_one_def</b>	$\vdash OneOne = (\lambda f \bullet \forall x1 x2 \bullet f x1 = f x2 \Rightarrow x1 = x2)$
<b>Onto</b>	
<b>onto_def</b>	$\vdash Onto = (\lambda f \bullet \forall y \bullet \exists x \bullet y = f x)$
<b>TypeDefn</b>	

**type\_defn\_def**

$$\begin{aligned} &\vdash \text{TypeDefn} \\ &= (\lambda P \text{ rep} \\ &\quad \bullet \text{OneOne } \text{rep} \wedge (\forall x \bullet P \ x \Leftrightarrow (\exists y \bullet x = \text{rep } y))) \end{aligned}$$
**45 THE THEORY min****45.1 Children** $log$ **45.2 Constants**

$$\begin{aligned} \$\Rightarrow & \quad Bool \rightarrow Bool \rightarrow Bool \\ \$= & \quad 'a \rightarrow 'a \rightarrow Bool \\ \$\epsilon & \quad ('a \rightarrow Bool) \rightarrow 'a \end{aligned}$$
**45.3 Types** $'1 \rightarrow '2$ **Bool****IND****45.4 Fixity**

$$\begin{aligned} \text{Binder:} & \quad \epsilon \quad \lambda \\ \text{Right Infix } 20: & \\ & \Rightarrow \\ \text{Right Infix } 100: & \\ & \rightarrow \\ \text{Right Infix } 200: & \\ & = \end{aligned}$$
**45.5 Terminators** $\rightarrow \Rightarrow = \lambda \epsilon$ **46 THE THEORY misc****46.1 Parents** $init$

## 46.2 Children

*pair*

## 46.3 Constants

<b>Arbitrary</b>	$'a$
$\$ \exists_1$	$('a \rightarrow Bool) \rightarrow Bool$
<b>Let</b>	$('a \rightarrow 'b) \rightarrow 'a \rightarrow 'b$
<b>Cond</b>	$Bool \rightarrow 'a \rightarrow 'a \rightarrow 'a$
<b>pp'TS</b>	$Bool \rightarrow Bool$

## 46.4 Aliases

$\Leftrightarrow$   $\$ = : Bool \rightarrow Bool \rightarrow Bool$

## 46.5 Fixity

<i>Binder:</i>	$\exists_1$
<i>Right Infix 10:</i>	$\Leftrightarrow$

## 46.6 Terminators

$\exists_1 \quad \Leftrightarrow$

## 46.7 Definitions

$\exists_1$	
$\exists_1\_def$	$\vdash \$ \exists_1 = (\lambda P \bullet \exists t \bullet P t \wedge (\forall x \bullet P x \Rightarrow x = t))$
<b>Let</b>	
$let\_def$	$\vdash Let = (\lambda f x \bullet f x)$
<b>Cond</b>	
$cond\_def$	$\vdash Cond$ $= (\lambda b x1 x2$ $\bullet \epsilon x$ $\bullet ((b \Leftrightarrow true) \Rightarrow x = x1) \wedge ((b \Leftrightarrow false) \Rightarrow x = x2))$
<b>pp'TS</b>	
$pp'ts\_def$	$\vdash \forall x \bullet pp'TS x \Leftrightarrow x$

## 46.8 Theorems

<b>t_thm</b>	$\vdash true$
<b>v_thm</b>	$\vdash \forall t1\ t2 \bullet t1 \vee t2 \Leftrightarrow (\forall b \bullet (t1 \Rightarrow b) \Rightarrow (t2 \Rightarrow b) \Rightarrow b)$
<b>not_thm</b>	$\vdash \forall t \bullet \neg t \Leftrightarrow t \Rightarrow false$
<b>f_thm</b>	$\vdash \neg false$
<b>not_t_thm</b>	$\vdash \neg true \Leftrightarrow false$
<b>and_thm</b>	$\vdash \forall t1\ t2 \bullet t1 \wedge t2 \Leftrightarrow (\forall b \bullet (t1 \Rightarrow t2 \Rightarrow b) \Rightarrow b)$
<b>not_and1</b>	$\vdash \forall t \bullet \neg t \Leftrightarrow t \Leftrightarrow false$
<b>exists_intro_thm</b>	$\vdash \forall P\ x \bullet P\ x \Rightarrow \exists P$
<b>cond_thm</b>	$\vdash \forall a\ t1\ t2$ $\bullet (if\ a\ then\ t1\ else\ t2)$ $= (\epsilon\ x$ $\bullet ((a \Leftrightarrow true) \Rightarrow x = t1) \wedge ((a \Leftrightarrow false) \Rightarrow x = t2))$
<b>not_forall_thm</b>	$\vdash \forall p \bullet \neg \forall p \Leftrightarrow (\exists x \bullet \neg p\ x)$
<b>not_exists_thm</b>	$\vdash \forall p \bullet \neg \exists p \Leftrightarrow (\forall x \bullet \neg p\ x)$
<b>exists1_thm</b>	$\vdash \forall P \bullet \exists_1 P \Leftrightarrow (\exists t \bullet P\ t \wedge (\forall x \bullet P\ x \Rightarrow x = t))$
<b>iff_thm</b>	$\vdash \forall a\ b \bullet (a \Leftrightarrow b) \Leftrightarrow (a \Rightarrow b) \wedge (b \Rightarrow a)$
<b>implies_thm</b>	$\vdash \forall a\ b \bullet a \Rightarrow b \Leftrightarrow \neg a \vee b$
<b>not_not_thm</b>	$\vdash \forall a \bullet \neg \neg a \Leftrightarrow a$
<b>not_or_thm</b>	$\vdash \forall a\ b \bullet \neg (a \vee b) \Leftrightarrow \neg a \wedge \neg b$
<b>not_and_thm</b>	$\vdash \forall a\ b \bullet \neg (a \wedge b) \Leftrightarrow \neg a \vee \neg b$
<b>not_implies_thm</b>	$\vdash \forall a\ b \bullet \neg (a \Rightarrow b) \Leftrightarrow a \wedge \neg b$
<b>not_iff_thm</b>	$\vdash \forall a\ b \bullet \neg (a \Leftrightarrow b) \Leftrightarrow a \wedge \neg b \vee b \wedge \neg a$
<b>not_f_thm</b>	$\vdash \neg false \Leftrightarrow true$
<b>not_if_thm</b>	$\vdash \forall a\ b\ c$ $\bullet \neg (if\ a\ then\ b\ else\ c) \Leftrightarrow (if\ a\ then\ \neg b\ else\ \neg c)$
<b>if_thm</b>	$\vdash \forall a\ b\ c \bullet (if\ a\ then\ b\ else\ c) \Leftrightarrow a \wedge b \vee \neg a \wedge c$
<b>eq_rewrite_thm</b>	$\vdash \forall x \bullet x = x \Leftrightarrow true$
<b>iff_rewrite_thm</b>	$\vdash \forall t$ $\bullet ((true \Leftrightarrow t) \Leftrightarrow t)$ $\wedge ((t \Leftrightarrow true) \Leftrightarrow t)$ $\wedge ((false \Leftrightarrow t) \Leftrightarrow \neg t)$ $\wedge ((t \Leftrightarrow false) \Leftrightarrow \neg t)$
<b>not_rewrite_thm</b>	$\vdash \forall t$ $\bullet (\neg \neg t \Leftrightarrow t) \wedge (\neg true \Leftrightarrow false) \wedge (\neg false \Leftrightarrow true)$
<b>and_rewrite_thm</b>	$\vdash \forall t$ $\bullet (true \wedge t \Leftrightarrow t)$ $\wedge (t \wedge true \Leftrightarrow t)$ $\wedge \neg (false \wedge t)$ $\wedge \neg (t \wedge false)$ $\wedge (t \wedge t \Leftrightarrow t)$
<b>forall_rewrite_thm</b>	$\vdash \forall t$

---

- $(true \vee t)$
- $\wedge (t \vee true)$
- $\wedge (false \vee t \Leftrightarrow t)$
- $\wedge (t \vee false \Leftrightarrow t)$
- $\wedge (t \vee t \Leftrightarrow t)$

 **$\Rightarrow$ \_rewrite\_thm**     $\vdash \forall t$ 

- $(true \Rightarrow t \Leftrightarrow t)$
- $\wedge (false \Rightarrow t \Leftrightarrow true)$
- $\wedge (t \Rightarrow true \Leftrightarrow true)$
- $\wedge (t \Rightarrow t \Leftrightarrow true)$
- $\wedge (t \Rightarrow false \Leftrightarrow \neg t)$

**if\_rewrite\_thm**     $\vdash \forall t1\ t2$ 

- $(if\ true\ then\ t1\ else\ t2) = t1$
- $\wedge (if\ false\ then\ t1\ else\ t2) = t2$

 **$\forall$ \_rewrite\_thm**     $\vdash \forall t \bullet (\forall x \bullet t) \Leftrightarrow t$ 
 **$\exists$ \_rewrite\_thm**     $\vdash \forall t \bullet (\exists x \bullet t) \Leftrightarrow t$ 
 **$\beta$ \_rewrite\_thm**     $\vdash \forall t1\ t2 \bullet (\lambda x \bullet t1)\ t2 = t1$ 
**one\_one\_thm**     $\vdash \forall f \bullet OneOne\ f \Leftrightarrow (\forall x\ y \bullet f\ x = f\ y \Leftrightarrow x = y)$ 
**ext\_thm**     $\vdash \forall f\ g \bullet f = g \Leftrightarrow (\forall x \bullet f\ x = g\ x)$ 
**type\_lemmas\_thm**     $\vdash \forall pred$ 

- $(\exists f \bullet TypeDefn\ pred\ f)$
- $\Rightarrow (\exists abs\ rep$ 
  - $(\forall a \bullet abs\ (rep\ a) = a)$
  - $\wedge (\forall r \bullet pred\ r \Leftrightarrow rep\ (abs\ r) = r))$

**fun\_rel\_thm**     $\vdash \forall r$ 

- $(\exists f \bullet \forall x\ y \bullet f\ x = y \Leftrightarrow r\ x\ y)$
- $\Leftrightarrow (\forall x \bullet \exists y \bullet r\ x\ y \wedge (\forall z \bullet r\ x\ z \Rightarrow z = y))$

## 47 THE THEORY one

### 47.1 Parents

*basic\_hol*

### 47.2 Children

*hol*

### 47.3 Constants

**IsOneRep**    *BOOL*  $\rightarrow$  *BOOL*  
**One**    *ONE*

## 47.4 Types

ONE

## 47.5 Definitions

IsOneRep

is\_one\_rep\_def

$\vdash \exists one \bullet \forall x \bullet IsOneRep\ x \Leftrightarrow x \Leftrightarrow one$

ONE

one\_def1

$\vdash \exists f \bullet TypeDefn\ IsOneRep\ f$

One

one\_def

$\vdash \forall x \bullet x = One$

## 47.6 Theorems

one\_fns\_thm  $\vdash \forall f \bullet f = (\lambda x \bullet One)$

# 48 THE THEORY orders

## 48.1 Parents

*sets*

## 48.2 Children

$\mathbb{R}$

## 48.3 Constants

Irrefl  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

Antisym  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

Trans  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

StrictPartialOrder

$'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

Trich  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

StrictLinearOrder

$'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

\$DenseIn  $'a\ SET \rightarrow 'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

Dense  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

UpperBound  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \times 'a \rightarrow BOOL$

\$HasSupremum  $'a\ SET \rightarrow 'a \times 'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

UnboundedAbove

$'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

UnboundedBelow

$'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

Complete  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow BOOL$

Cuts  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \rightarrow 'a\ SET\ SET$

DownSet  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \times 'a \rightarrow 'a\ SET$

DownSets  $'a\ SET \times ('a \rightarrow 'a \rightarrow BOOL) \times 'a\ SET \rightarrow 'a\ SET\ SET$

## 48.4 Fixity

*Right Infix 210:***DenseIn**      **HasSupremum**      <<      <<<

## 48.5 Definitions

<b>Irrefl</b>	$\vdash \forall X \$\llcorner \bullet \text{Irrefl}(X, \$\llcorner) \Leftrightarrow (\forall x \bullet x \in X \Rightarrow \neg x \llcorner x)$
<b>Antisym</b>	$\vdash \forall X \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>\text{Antisym}(X, \\$\llcorner)</math>  <math>\Leftrightarrow (\forall x y</math>  <ul style="list-style-type: none"> <li>• <math>x \in X \wedge y \in X \wedge \neg x = y \Rightarrow \neg (x \llcorner y \wedge y \llcorner x))</math></li> </ul> </li> </ul>
<b>Trans</b>	$\vdash \forall X \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>\text{Trans}(X, \\$\llcorner)</math>  <math>\Leftrightarrow (\forall x y z</math>  <ul style="list-style-type: none"> <li>• <math>x \in X \wedge y \in X \wedge z \in X \wedge x \llcorner y \wedge y \llcorner z</math>  <math>\Rightarrow x \llcorner z)</math></li> </ul> </li> </ul>
<b>StrictPartialOrder</b>	$\vdash \forall X \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>\text{StrictPartialOrder}(X, \\$\llcorner)</math>  <math>\Leftrightarrow \text{Irrefl}(X, \\$\llcorner)</math>  <math>\wedge \text{Antisym}(X, \\$\llcorner)</math>  <math>\wedge \text{Trans}(X, \\$\llcorner)</math></li> </ul>
<b>Trich</b>	$\vdash \forall X \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>\text{Trich}(X, \\$\llcorner)</math>  <math>\Leftrightarrow (\forall x y</math>  <ul style="list-style-type: none"> <li>• <math>x \in X \wedge y \in X \wedge \neg x = y \Rightarrow x \llcorner y \vee y \llcorner x)</math></li> </ul> </li> </ul>
<b>StrictLinearOrder</b>	$\vdash \forall X \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>\text{StrictLinearOrder}(X, \\$\llcorner)</math>  <math>\Leftrightarrow \text{StrictPartialOrder}(X, \\$\llcorner) \wedge \text{Trich}(X, \\$\llcorner)</math></li> </ul>
<b>DenseIn</b>	$\vdash \forall A X \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>A \text{DenseIn}(X, \\$\llcorner)</math>  <math>\Leftrightarrow (\forall x y</math>  <ul style="list-style-type: none"> <li>• <math>x \in X \wedge y \in X \wedge x \llcorner y</math>  <math>\Rightarrow (\exists a \bullet a \in A \wedge x \llcorner a \wedge a \llcorner y))</math></li> </ul> </li> </ul>
<b>Dense</b>	$\vdash \forall X \$\llcorner \bullet \text{Dense}(X, \$\llcorner) \Leftrightarrow X \text{DenseIn}(X, \$\llcorner)$
<b>UpperBound</b>	$\vdash \forall A \$\llcorner x$ <ul style="list-style-type: none"> <li>• <math>\text{UpperBound}(A, \\$\llcorner, x) \Leftrightarrow (\forall a \bullet a \in A \Rightarrow a \llcorner x)</math></li> </ul>
<b>HasSupremum</b>	$\vdash \forall A \$\llcorner x X$ <ul style="list-style-type: none"> <li>• <math>A \text{HasSupremum}(x, X, \\$\llcorner)</math>  <math>\Leftrightarrow \text{UpperBound}(A, \\$\llcorner, x)</math>  <math>\wedge (\forall y</math>  <ul style="list-style-type: none"> <li>• <math>y \in X \wedge \text{UpperBound}(A, \\$\llcorner, y) \Rightarrow \neg y \llcorner x)</math></li> </ul> </li> </ul>
<b>UnboundedAbove</b>	$\vdash \forall A \$\llcorner$ <ul style="list-style-type: none"> <li>• <math>\text{UnboundedAbove}(A, \\$\llcorner)</math>  <math>\Leftrightarrow (\forall b \bullet b \in A \Rightarrow (\exists c \bullet c \in A \wedge b \llcorner c))</math></li> </ul>

**UnboundedBelow**

$\vdash \forall A \ \$\lll$   
 • *UnboundedBelow* ( $A, \ \$\lll$ )  
 $\Leftrightarrow (\forall b \bullet b \in A \Rightarrow (\exists c \bullet c \in A \wedge c \lll b))$

**Complete**

$\vdash \forall X \ \$\lll$   
 • *Complete* ( $X, \ \$\lll$ )  
 $\Leftrightarrow (\forall A \ x$   
 •  $\neg A = \{\}$   
 $\wedge A \subseteq X$   
 $\wedge \text{UnboundedAbove} (A, \ \$\lll)$   
 $\wedge x \in X$   
 $\wedge \text{UpperBound} (A, \ \$\lll, x)$   
 $\Rightarrow (\exists y \bullet y \in X \wedge A \text{ HasSupremum} (y, X, \ \$\lll)))$

**Cuts**

$\vdash \forall X \ \$\lll \ A$   
 •  $A \in \text{Cuts} (X, \ \$\lll)$   
 $\Leftrightarrow A \subseteq X$   
 $\wedge \neg A = \{\}$   
 $\wedge \text{UnboundedAbove} (A, \ \$\lll)$   
 $\wedge (\exists x \bullet x \in X \wedge \text{UpperBound} (A, \ \$\lll, x))$   
 $\wedge (\forall a \ b \bullet a \in A \wedge b \in X \wedge b \lll a \Rightarrow b \in A)$

**DownSet**

$\vdash \forall X \ \$\lll \ x \bullet \text{DownSet} (X, \ \$\lll, x) = \{y \mid y \in X \wedge y \lll x\}$

**DownSets**

$\vdash \forall X \ \$\lll \ A \ s$   
 •  $s \in \text{DownSets} (X, \ \$\lll, A)$   
 $\Leftrightarrow (\exists x \bullet x \in A \wedge s = \text{DownSet} (X, \ \$\lll, x))$

**48.6 Theorems**

**$\lll\_irrefl\_thm$**   $\vdash \forall V \bullet \text{Irrefl} (V, \ \$\lll)$

 **$\lll\_antisym\_thm$** 

$\vdash \forall V \bullet \text{Antisym} (V, \ \$\lll)$

**$\lll\_trans\_thm$**   $\vdash \forall V \bullet \text{Trans} (V, \ \$\lll)$

 **$\lll\_cuts\_trich\_thm$** 

$\vdash \forall X \ \$\lll$   
 •  $\text{Trans} (X, \ \$\lll) \wedge \text{Trich} (X, \ \$\lll)$   
 $\Rightarrow \text{Trich} (\text{Cuts} (X, \ \$\lll), \ \$\lll)$

 **$\lll\_cuts\_strict\_partial\_order\_thm$** 

$\vdash \forall X \ \$\lll \bullet \text{StrictPartialOrder} (\text{Cuts} (X, \ \$\lll), \ \$\lll)$

 **$\lll\_cuts\_strict\_linear\_order\_thm$** 

$\vdash \forall X \ \$\lll$   
 •  $\text{StrictLinearOrder} (X, \ \$\lll)$   
 $\Rightarrow \text{StrictLinearOrder} (\text{Cuts} (X, \ \$\lll), \ \$\lll)$

 **$\lll\_cuts\_complete\_thm$** 

$\vdash \forall X \ \$\lll \bullet \text{Complete} (\text{Cuts} (X, \ \$\lll), \ \$\lll)$

 **$\lll\_down\_sets\_cuts\_thm$** 

$\vdash \forall X \ \$\lll \ A$   
 •  $A \subseteq X$   
 $\wedge \text{Irrefl} (X, \ \$\lll)$   
 $\wedge \text{Trans} (X, \ \$\lll)$

$$\begin{aligned} & \wedge \text{UnboundedBelow } (A, \$\ll) \\ & \wedge A \text{ DenseIn } (X, \$\ll) \\ \Rightarrow & \text{DownSets } (X, \$\ll, A) \subseteq \text{Cuts } (X, \$\ll) \end{aligned}$$

**down\_sets\_dense\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll A \\ & \bullet A \subseteq X \\ & \quad \wedge \text{StrictLinearOrder } (X, \$\ll) \\ & \quad \wedge \text{UnboundedBelow } (A, \$\ll) \\ & \quad \wedge A \text{ DenseIn } (X, \$\ll) \\ \Rightarrow & \text{DownSets } (X, \$\ll, A) \\ & \quad \text{DenseIn } (\text{Cuts } (X, \$\ll), \$\ll) \end{aligned}$$

**dense\_superset\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll A B \\ & \bullet A \subseteq B \wedge B \subseteq X \wedge A \text{ DenseIn } (X, \$\ll) \\ \Rightarrow & B \text{ DenseIn } (X, \$\ll) \end{aligned}$$

**dense\_universe\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll A \\ & \bullet A \subseteq X \wedge A \text{ DenseIn } (X, \$\ll) \Rightarrow \text{Dense } (X, \$\ll) \end{aligned}$$

**downset\_cut\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll a \\ & \bullet a \in X \\ & \quad \wedge \text{Trans } (X, \$\ll) \\ & \quad \wedge \text{UnboundedBelow } (X, \$\ll) \\ & \quad \wedge X \text{ DenseIn } (X, \$\ll) \\ \Rightarrow & \text{DownSet } (X, \$\ll, a) \in \text{Cuts } (X, \$\ll) \end{aligned}$$

**down\_sets\_less\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll a b \\ & \bullet a \in X \wedge b \in X \wedge \text{StrictLinearOrder } (X, \$\ll) \\ \Rightarrow & (\text{DownSet } (X, \$\ll, a) \subset \text{DownSet } (X, \$\ll, b)) \\ \Leftrightarrow & a \ll b \end{aligned}$$

**cuts\_unbounded\_above\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll \\ & \bullet \text{Irrefl } (X, \$\ll) \\ & \quad \wedge \text{Trans } (X, \$\ll) \\ & \quad \wedge \text{UnboundedAbove } (X, \$\ll) \\ & \quad \wedge X \text{ DenseIn } (X, \$\ll) \\ \Rightarrow & \text{UnboundedAbove } (\text{Cuts } (X, \$\ll), \$\ll) \end{aligned}$$

**cuts\_unbounded\_below\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll \\ & \bullet \text{Irrefl } (X, \$\ll) \\ & \quad \wedge \text{Trans } (X, \$\ll) \\ & \quad \wedge \text{UnboundedBelow } (X, \$\ll) \\ & \quad \wedge X \text{ DenseIn } (X, \$\ll) \\ \Rightarrow & \text{UnboundedBelow } (\text{Cuts } (X, \$\ll), \$\ll) \end{aligned}$$

**dense\_complete\_subset\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll A \\ & \bullet \text{StrictLinearOrder } (X, \$\ll) \\ & \quad \wedge A \subseteq X \end{aligned}$$

$$\begin{aligned} & \wedge \text{UnboundedAbove } (X, \$\ll) \\ & \wedge \text{UnboundedBelow } (X, \$\ll) \\ & \wedge A \text{DenseIn } (X, \$\ll) \\ & \wedge \text{Complete } (A, \$\ll) \\ \Rightarrow & A = X \end{aligned}$$
**induced\_order\_irrefl\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \wedge \text{Irrefl } (X, \$\ll) \\ \Rightarrow & \text{Irrefl } (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_antisym\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \wedge \text{Irrefl } (X, \$\ll) \wedge \text{Antisym } (X, \$\ll) \\ \Rightarrow & \text{Antisym } (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_trans\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \wedge \text{Trans } (X, \$\ll) \\ \Rightarrow & \text{Trans } (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_trich\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \wedge \text{OneOne } f \wedge \text{Trich } (X, \$\ll) \\ \Rightarrow & \text{Trich } (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_strict\_partial\_order\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \wedge \text{StrictPartialOrder } (X, \$\ll) \\ \Rightarrow & \text{StrictPartialOrder} \\ & (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_strict\_linear\_order\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \\ & \quad \wedge \text{OneOne } f \\ & \quad \wedge \text{StrictLinearOrder } (X, \$\ll) \\ \Rightarrow & \text{StrictLinearOrder} \\ & (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_complete\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \\ & \quad \wedge \text{OneOne } f \\ & \quad \wedge (\forall x \bullet x \in X \Rightarrow (\exists a \bullet x = f a)) \\ & \quad \wedge \text{Complete } (X, \$\ll) \\ \Rightarrow & \text{Complete } (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_dense\_thm**

$$\begin{aligned} & \vdash \forall X \$\ll f A \\ & \bullet (\forall a \bullet f a \in X) \\ & \quad \wedge \text{OneOne } f \\ & \quad \wedge (\forall x \bullet x \in X \Rightarrow (\exists a \bullet x = f a)) \\ & \quad \wedge \{x \mid \exists a \bullet a \in A \wedge x = f a\} \text{DenseIn } (X, \$\ll) \\ \Rightarrow & A \text{DenseIn } (\text{Universe}, (\lambda a b \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_unbounded\_above\_thm**

---


$$\begin{aligned} & \vdash \forall X \ \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \\ & \quad \wedge \text{OneOne } f \\ & \quad \wedge (\forall x \bullet x \in X \Rightarrow (\exists a \bullet x = f a)) \\ & \quad \wedge \text{UnboundedAbove } (X, \ \$\ll) \\ & \Rightarrow \text{UnboundedAbove } (\text{Universe}, (\lambda a \bullet f a \ll f b)) \end{aligned}$$
**induced\_order\_unbounded\_below\_thm**

$$\begin{aligned} & \vdash \forall X \ \$\ll f \\ & \bullet (\forall a \bullet f a \in X) \\ & \quad \wedge \text{OneOne } f \\ & \quad \wedge (\forall x \bullet x \in X \Rightarrow (\exists a \bullet x = f a)) \\ & \quad \wedge \text{UnboundedBelow } (X, \ \$\ll) \\ & \Rightarrow \text{UnboundedBelow } (\text{Universe}, (\lambda a \bullet f a \ll f b)) \end{aligned}$$
**49 THE THEORY pair****49.1 Parents***misc***49.2 Children**

N

**49.3 Constants**

<b>IsPairRep</b>	$(\ 'a \rightarrow \ 'b \rightarrow \ \text{Bool}) \rightarrow \ \text{Bool}$
<b>\$</b>	$\ 'a \rightarrow \ 'b \rightarrow \ 'a \times \ 'b$
<b>Fst</b>	$\ 'a \times \ 'b \rightarrow \ 'a$
<b>Snd</b>	$\ 'a \times \ 'b \rightarrow \ 'b$
<b>Uncurry</b>	$(\ 'a \rightarrow \ 'b \rightarrow \ 'c) \rightarrow \ 'a \times \ 'b \rightarrow \ 'c$
<b>Curry</b>	$(\ 'a \times \ 'b \rightarrow \ 'c) \rightarrow \ 'a \rightarrow \ 'b \rightarrow \ 'c$
<b>Pair</b>	$(\ 'a \rightarrow \ 'b) \times (\ 'a \rightarrow \ 'c) \rightarrow \ 'a \rightarrow \ 'b \times \ 'c$

**49.4 Types** $\ '1 \times \ '2$ **49.5 Fixity***Right Infix 150:*

,      ×

**49.6 Terminators**

×

## 49.7 Definitions

**IsPairRep****is\_pair\_rep\_def**

$$\begin{aligned} &\vdash \exists \text{comma fst snd} \\ &\bullet (\forall x y \\ &\quad \bullet \text{IsPairRep (comma x y)} \\ &\quad \quad \wedge \text{fst (comma x y) = x} \\ &\quad \quad \wedge \text{snd (comma x y) = y} \\ &\quad \wedge (\forall x y a b \\ &\quad \bullet \text{comma a b = comma x y} \Leftrightarrow a = x \wedge b = y) \\ &\quad \wedge (\forall p \bullet \text{IsPairRep p} \Rightarrow \text{comma (fst p) (snd p) = p}) \end{aligned}$$

×

**×\_def**

$$\vdash \exists f \bullet \text{TypeDefn IsPairRep f}$$
**comma\_def**

$$\begin{aligned} &\vdash \exists \text{fst snd} \\ &\bullet (\forall x y \bullet \text{fst (x, y) = x} \wedge \text{snd (x, y) = y}) \\ &\quad \wedge (\forall x y a b \bullet (a, b) = (x, y) \Leftrightarrow a = x \wedge b = y) \\ &\quad \wedge (\forall p \bullet (\text{fst p, snd p}) = p) \end{aligned}$$
**fst\_def**

$$\begin{aligned} &\vdash \exists \text{snd} \\ &\bullet (\forall x y \bullet \text{Fst (x, y) = x} \wedge \text{snd (x, y) = y}) \\ &\quad \wedge (\forall x y a b \bullet (a, b) = (x, y) \Leftrightarrow a = x \wedge b = y) \\ &\quad \wedge (\forall p \bullet (\text{Fst p, snd p}) = p) \end{aligned}$$

,

**Fst****Snd****pair\_ops\_def**

$$\begin{aligned} \text{snd\_def} &\quad \vdash (\forall x y \bullet \text{Fst (x, y) = x} \wedge \text{Snd (x, y) = y}) \\ &\quad \wedge (\forall x y a b \bullet (a, b) = (x, y) \Leftrightarrow a = x \wedge b = y) \\ &\quad \wedge (\forall p \bullet (\text{Fst p, Snd p}) = p) \end{aligned}$$
**Uncurry**

$$\text{uncurry\_def} \quad \vdash \forall f x y \bullet \text{Uncurry f (x, y) = f x y}$$
**Curry**

$$\text{curry\_def} \quad \vdash \forall f x y \bullet \text{Curry f x y = f (x, y)}$$
**Pair**

$$\text{pair\_def} \quad \vdash \forall f g \bullet \text{Pair (f, g) = } (\lambda x \bullet (f x, g x))$$

## 49.8 Theorems

$$\begin{aligned} \text{pair\_clauses} &\quad \vdash \forall x y a b p fu fc fg \\ &\bullet \text{Fst (x, y) = x} \\ &\quad \wedge \text{Snd (x, y) = y} \\ &\quad \wedge ((a, b) = (x, y) \Leftrightarrow a = x \wedge b = y) \\ &\quad \wedge (\text{Fst p, Snd p}) = p \\ &\quad \wedge \text{Curry fc x y = fc (x, y)} \\ &\quad \wedge \text{Uncurry fu (x, y) = fu x y} \\ &\quad \wedge \text{Uncurry fu p = fu (Fst p) (Snd p)} \\ &\quad \wedge ((a, b) = p \Leftrightarrow a = \text{Fst p} \wedge b = \text{Snd p}) \\ &\quad \wedge (p = (a, b) \Leftrightarrow \text{Fst p} = a \wedge \text{Snd p} = b) \end{aligned}$$

$$\wedge \text{Pair } fg \ x = (\text{Fst } fg \ x, \text{Snd } fg \ x)$$

## 50 THE THEORY seq

### 50.1 Parents

*fun\_rel*

### 50.2 Children

*fn\_set*

### 50.3 Constants

<b>Elms</b>	$'a \text{ LIST} \rightarrow 'a \mathbb{P}$
<b>Distinct</b>	$'a \text{ LIST } \mathbb{P}$
<b>Lists</b>	$'a \mathbb{P} \rightarrow 'a \text{ LIST } \mathbb{P}$
<b>Lists<sub>1</sub></b>	$'a \mathbb{P} \rightarrow 'a \text{ LIST } \mathbb{P}$
<b>InjLists</b>	$'a \mathbb{P} \rightarrow 'a \text{ LIST } \mathbb{P}$
<b>Nth</b>	$'a \text{ LIST} \rightarrow \mathbb{N} \rightarrow 'a$
<b>\$..</b>	$\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N} \mathbb{P}$
<b>ListRel</b>	$'a \text{ LIST} \rightarrow \mathbb{N} \leftrightarrow 'a$
<b>RelList</b>	$\mathbb{N} \leftrightarrow 'a \rightarrow 'a \text{ LIST}$
<b>Last</b>	$'a \text{ LIST} \rightarrow 'a$
<b>Front</b>	$'a \text{ LIST} \rightarrow 'a \text{ LIST}$
<b>\$ </b>	$'a \text{ LIST} \rightarrow 'a \mathbb{P} \rightarrow 'a \text{ LIST}$
<b>Enumerate</b>	$\mathbb{N} \mathbb{P} \rightarrow \mathbb{N} \leftrightarrow \mathbb{N}$
<b>Squash</b>	$\mathbb{N} \leftrightarrow 'a \rightarrow \mathbb{N} \leftrightarrow 'a$
<b>Extract</b>	$\mathbb{N} \mathbb{P} \rightarrow 'a \text{ LIST} \rightarrow 'a \text{ LIST}$
<b>\$Prefix</b>	$'a \text{ LIST} \rightarrow 'a \text{ LIST} \rightarrow \text{BOOL}$
<b>\$Suffix</b>	$'a \text{ LIST} \rightarrow 'a \text{ LIST} \rightarrow \text{BOOL}$
<b>\$In</b>	$'a \text{ LIST} \rightarrow 'a \text{ LIST} \rightarrow \text{BOOL}$
<b>Flat</b>	$'a \text{ LIST } \text{LIST} \rightarrow 'a \text{ LIST}$

### 50.4 Aliases

<b>#</b>	$\text{Length} : 'a \text{ LIST} \rightarrow \mathbb{N}$
<b>^</b>	$\text{Append} : 'a \text{ LIST} \rightarrow 'a \text{ LIST} \rightarrow 'a \text{ LIST}$
<b>Head</b>	$\text{Hd} : 'a \text{ LIST} \rightarrow 'a$
<b>Tail</b>	$\text{Tl} : 'a \text{ LIST} \rightarrow 'a \text{ LIST}$

### 50.5 Fixity

*Left Infix 300:*

†

*Right Infix 290:*

..

*Right Infix 300:*

**In Prefix Suffix ^**

## 50.6 Definitions

<b>Elms</b>	$\vdash \text{Elms } [] = \{\}$ $\wedge (\forall x l \bullet \text{Elms } (\text{Cons } x l) = \{x\} \cup \text{Elms } l)$
<b>Distinct</b>	$\vdash [] \in \text{Distinct}$ $\wedge (\forall x l$ $\bullet \text{Cons } x l \in \text{Distinct}$ $\Leftrightarrow \neg x \in \text{Elms } l \wedge l \in \text{Distinct})$
<b>Lists</b>	$\vdash \forall a \bullet \text{Lists } a = \{l \mid \text{Elms } l \subseteq a\}$
<b>Lists<sub>1</sub></b>	$\vdash \forall a \bullet \text{Lists}_1 a = \text{Lists } a \setminus \{\{\}\}$
<b>InjLists</b>	$\vdash \forall a \bullet \text{InjLists } a = \text{Lists } a \cap \text{Distinct}$
<b>Nth</b>	$\vdash \forall l x n$ $\bullet \text{Nth } (\text{Cons } x l) n$ $= (\text{if } n = 1 \text{ then } x \text{ else } \text{Nth } l (n - 1))$
<b>..</b>	$\vdash \forall m n \bullet m .. n = \{i \mid m \leq i \wedge i \leq n\}$
<b>ListRel</b>	$\vdash \forall l \bullet \text{ListRel } l = 1 .. \# l \triangleleft \text{Graph } (\text{Nth } l)$
<b>RelList</b>	$\vdash \text{ConstSpec}$ $(\lambda \text{RelList}' \bullet \forall l \bullet \text{RelList}' (\text{ListRel } l) = l)$ $\text{RelList}$
<b>Last</b>	$\vdash \forall x l$ $\bullet \text{Last } (\text{Cons } x l) = (\text{if } l = [] \text{ then } x \text{ else } \text{Last } l)$
<b>Front</b>	$\vdash \forall x l$ $\bullet \text{Front } (\text{Cons } x l)$ $= (\text{if } l = [] \text{ then } [] \text{ else } \text{Cons } x (\text{Front } l))$
<b>↑</b>	$\vdash \forall a x l$ $\bullet [] \uparrow a = []$ $\wedge \text{Cons } x l \uparrow a$ $= (\text{if } x \in a \text{ then } \text{Cons } x (l \uparrow a) \text{ else } l \uparrow a)$
<b>Enumerate</b>	$\vdash \forall a$ $\bullet \text{Enumerate } a$ $= \{(i, j)$ $\mid \exists l$ $\bullet j \in a$ $\wedge l \in \text{Distinct}$ $\wedge \# l = i$ $\wedge \text{Elms } l = a \cap 0 .. j\}$
<b>Squash</b>	$\vdash \forall r \bullet \text{Squash } r = \text{Enumerate } (\text{Dom } r) \circledast r$
<b>Extract</b>	$\vdash \forall a l \bullet \text{Extract } a l = \text{RelList } (\text{Squash } (a \triangleleft \text{ListRel } l))$
<b>Prefix</b>	$\vdash \forall s t \bullet s \text{Prefix } t \Leftrightarrow (\exists v \bullet s \hat{\ } v = t)$
<b>Suffix</b>	$\vdash \forall s t \bullet s \text{Suffix } t \Leftrightarrow (\exists u \bullet u \hat{\ } s = t)$
<b>In</b>	$\vdash \forall s t \bullet s \text{In } t \Leftrightarrow (\exists u v \bullet u \hat{\ } s \hat{\ } v = t)$
<b>Flat</b>	$\vdash \forall e l \bullet \text{Flat } [] = [] \wedge \text{Flat } (\text{Cons } e l) = e \hat{\ } \text{Flat } l$

## 51 THE THEORY sets

### 51.1 Parents

*basic\_hol*

### 51.2 Children

*orders set\_thms*     $\mathbb{Z}$     *hol*

### 51.3 Constants

<b>IsSetRep</b>	$('a \rightarrow \text{BOOL}) \rightarrow \text{BOOL}$
<b>\$SetComp</b>	$('a \rightarrow \text{BOOL}) \rightarrow 'a \text{ SET}$
<b>\$∈</b>	$'a \rightarrow 'a \text{ SET} \rightarrow \text{BOOL}$
<b>{}</b>	$'a \text{ SET}$
<b>Universe</b>	$'a \text{ SET}$
<b>Insert</b>	$'a \rightarrow 'a \text{ SET} \rightarrow 'a \text{ SET}$
<b>~</b>	$'a \text{ SET} \rightarrow 'a \text{ SET}$
<b>\$∪</b>	$'a \text{ SET} \rightarrow 'a \text{ SET} \rightarrow 'a \text{ SET}$
<b>\$∩</b>	$'a \text{ SET} \rightarrow 'a \text{ SET} \rightarrow 'a \text{ SET}$
<b>\$\</b>	$'a \text{ SET} \rightarrow 'a \text{ SET} \rightarrow 'a \text{ SET}$
<b>\$⊖</b>	$'a \text{ SET} \rightarrow 'a \text{ SET} \rightarrow 'a \text{ SET}$
<b>\$⊆</b>	$'a \text{ SET} \rightarrow 'a \text{ SET} \rightarrow \text{BOOL}$
<b>\$⊂</b>	$'a \text{ SET} \rightarrow 'a \text{ SET} \rightarrow \text{BOOL}$
<b>∪</b>	$'a \text{ SET SET} \rightarrow 'a \text{ SET}$
<b>∩</b>	$'a \text{ SET SET} \rightarrow 'a \text{ SET}$
<b>ℙ</b>	$'a \text{ SET} \rightarrow 'a \text{ SET SET}$

### 51.4 Types

$'1 \text{ SET}$

### 51.5 Fixity

<i>Binder:</i>	<b>SetComp</b>
<i>Left Infix 265:</i>	$\backslash$
<i>Right Infix 230:</i>	$\subseteq$ $\in$ $\subset$
<i>Right Infix 250:</i>	$\ominus$
<i>Right Infix 260:</i>	$\cup$
<i>Right Infix 270:</i>	$\cap$

## 51.6 Definitions

### IsSetRep

is\_set\_rep\_def

$$\vdash \text{IsSetRep} = (\lambda x \bullet T)$$

### SET

set\_def

$$\vdash \exists f \bullet \text{TypeDefn IsSetRep } f$$

SetComp

∈

set\_comp\_def

$$\vdash \forall x \ p \ a \ b \\ \bullet (x \in \{v|p \ v\} \Leftrightarrow p \ x) \\ \wedge (a = b \Leftrightarrow (\forall x \bullet x \in a \Leftrightarrow x \in b))$$

### Empty

Universe

Insert

insert\_def

$$\vdash \forall x \ y \ a \\ \bullet \neg x \in \{\} \\ \wedge x \in \text{Universe} \\ \wedge (x \in \text{Insert } y \ a \Leftrightarrow x = y \vee x \in a)$$

~

complement\_def

$$\vdash \forall x \ a \bullet x \in \sim a \Leftrightarrow \neg x \in a$$

U

U\_def

$$\vdash \forall x \ a \ b \bullet x \in a \cup b \Leftrightarrow x \in a \vee x \in b$$

∩

∩\_def

$$\vdash \forall x \ a \ b \bullet x \in a \cap b \Leftrightarrow x \in a \wedge x \in b$$

\

set\_dif\_def

$$\vdash \forall x \ a \ b \bullet x \in a \setminus b \Leftrightarrow x \in a \wedge \neg x \in b$$

⊖

⊖\_def

$$\vdash \forall x \ a \ b \bullet x \in a \ominus b \Leftrightarrow \neg (x \in a \Leftrightarrow x \in b)$$

⊆

⊆\_def

$$\vdash \forall a \ b \bullet a \subseteq b \Leftrightarrow (\forall x \bullet x \in a \Rightarrow x \in b)$$

⊂

⊂\_def

$$\vdash \forall a \ b \bullet a \subset b \Leftrightarrow a \subseteq b \wedge (\exists x \bullet \neg x \in a \wedge x \in b)$$

∪

∪\_def

$$\vdash \forall x \ a \bullet x \in \bigcup a \Leftrightarrow (\exists s \bullet x \in s \wedge s \in a)$$

∩

∩\_def

$$\vdash \forall x \ a \bullet x \in \bigcap a \Leftrightarrow (\forall s \bullet s \in a \Rightarrow x \in s)$$

ℙ

ℙ\_def

$$\vdash \forall x \ a \bullet x \in \mathbb{P} a \Leftrightarrow x \subseteq a$$

## 51.7 Theorems

sets\_clauses

$$\vdash \forall x \ y \ p \ q \\ \bullet (x \in \{\} \Leftrightarrow F) \\ \wedge (x \in \text{Universe} \Leftrightarrow T) \\ \wedge (x \in \{v|q\} \Leftrightarrow q) \\ \wedge (x \in \{v|p \ v\} \Leftrightarrow p \ x)$$

---

	$\wedge (x \in \{v v = y\} \Leftrightarrow x = y)$
	$\wedge (x \in \{y\} \Leftrightarrow x = y)$
<b>complement_clauses</b>	$\vdash (\forall x a \bullet x \in \sim a \Leftrightarrow \neg x \in a)$
	$\wedge \sim Universe = \{\}$
	$\wedge \sim \{\} = Universe$
<b>U_clauses</b>	$\vdash \forall a$
	$\bullet a \cup \{\} = a$
	$\wedge \{\} \cup a = a$
	$\wedge a \cup Universe = Universe$
	$\wedge Universe \cup a = Universe$
	$\wedge a \cup a = a$
<b><math>\cap</math>_clauses</b>	$\vdash \forall a$
	$\bullet a \cap \{\} = \{\}$
	$\wedge \{\} \cap a = \{\}$
	$\wedge a \cap Universe = a$
	$\wedge Universe \cap a = a$
	$\wedge a \cap a = a$
<b>set_dif_clauses</b>	$\vdash \forall a$
	$\bullet a \setminus \{\} = a$
	$\wedge \{\} \setminus a = \{\}$
	$\wedge a \setminus Universe = \{\}$
	$\wedge Universe \setminus a = \sim a$
	$\wedge a \setminus a = \{\}$
<b><math>\ominus</math>_clauses</b>	$\vdash \forall a$
	$\bullet a \ominus \{\} = a$
	$\wedge \{\} \ominus a = a$
	$\wedge a \ominus Universe = \sim a$
	$\wedge Universe \ominus a = \sim a$
	$\wedge a \ominus a = \{\}$
<b><math>\subseteq</math>_clauses</b>	$\vdash \forall a \bullet a \subseteq a \wedge \{\} \subseteq a \wedge a \subseteq Universe$
<b><math>\subset</math>_clauses</b>	$\vdash \forall a \bullet \neg a \subset a \wedge \neg a \subset \{\} \wedge \{\} \subset Universe$
<b><math>\cup</math>_clauses</b>	$\vdash \cup \{\} = \{\} \wedge \cup Universe = Universe$
<b><math>\cap</math>_clauses</b>	$\vdash \cap \{\} = Universe \wedge \cap Universe = \{\}$
<b><math>\mathbb{P}</math>_clauses</b>	$\vdash \forall a$
	$\bullet \mathbb{P} \{\} = \{\{\}\}$
	$\wedge \mathbb{P} Universe = Universe$
	$\wedge a \in \mathbb{P} a$
	$\wedge \{\} \in \mathbb{P} a$
<b><math>\emptyset</math>_clauses</b>	$\vdash \forall x a$
	$\bullet \{x F\} = \{\}$
	$\wedge \neg x \in \{\}$
	$\wedge \{\} \cup a = a$
	$\wedge a \cup \{\} = a$
	$\wedge \{\} \cap a = \{\}$
	$\wedge a \cap \{\} = \{\}$
	$\wedge a \setminus \{\} = a$

---

---


$$\begin{aligned}
& \wedge \{\} \setminus a = \{\} \\
& \wedge a \ominus \{\} = a \\
& \wedge \{\} \ominus a = a \\
& \wedge \{\} \subseteq a \\
& \wedge (a \subseteq \{\} \Leftrightarrow a = \{\}) \\
& \wedge (\{\} \subset a \Leftrightarrow \neg a = \{\}) \\
& \wedge \neg a \subset \{\} \\
& \wedge \neg x \in \bigcup \{\} \\
& \wedge x \in \bigcap \{\} \\
& \wedge \{\} \in \mathbb{P} a \\
& \wedge \mathbb{P} \{\} = \{\{\}\} \\
\in\_in\_clauses \quad & \vdash (\forall x y a \\
& \bullet (x \in Universe \Leftrightarrow T) \\
& \quad \wedge (x \in \{\} \Leftrightarrow F) \\
& \quad \wedge (x \in Insert\ y\ a \Leftrightarrow x = y \vee x \in a)) \\
& \wedge (\forall x a b \\
& \bullet (x \in a \cup b \Leftrightarrow x \in a \vee x \in b) \\
& \quad \wedge (x \in a \cap b \Leftrightarrow x \in a \wedge x \in b) \\
& \quad \wedge (x \in a \setminus b \Leftrightarrow x \in a \wedge \neg x \in b) \\
& \quad \wedge (x \in a \ominus b \Leftrightarrow \neg (x \in a \Leftrightarrow x \in b))) \\
& \wedge (\forall x a \\
& \bullet (x \in \bigcup a \Leftrightarrow (\exists s \bullet x \in s \wedge s \in a)) \\
& \quad \wedge (x \in \bigcap a \Leftrightarrow (\forall s \bullet s \in a \Rightarrow x \in s)) \\
& \quad \wedge (\forall x a \bullet x \in \mathbb{P} a \Leftrightarrow x \subseteq a)) \\
sets\_ext\_clauses \quad & \vdash \forall a b \\
& \bullet (a \subset b \\
& \quad \Leftrightarrow (\forall x \bullet x \in a \Rightarrow x \in b) \\
& \quad \quad \wedge (\exists x \bullet \neg x \in a \wedge x \in b)) \\
& \wedge (a \subseteq b \Leftrightarrow (\forall x \bullet x \in a \Rightarrow x \in b)) \\
& \wedge (a = b \Leftrightarrow (\forall x \bullet x \in a \Leftrightarrow x \in b))
\end{aligned}$$

## 52 THE THEORY set\_thms

### 52.1 Parents

*sets*

### 52.2 Children

*fun\_rel\_thms*

### 52.3 Constants

**Nest**  $'a \text{ SET SET} \rightarrow \text{BOOL}$   
**SubsetClosed**  $'a \text{ SET SET} \rightarrow \text{BOOL}$   
**NestClosed**  $'a \text{ SET SET} \rightarrow \text{BOOL}$   
**Maximal $\subseteq$**   $'a \text{ SET SET} \rightarrow 'a \text{ SET} \rightarrow \text{BOOL}$   
**Choose**  $'a \text{ SET} \rightarrow 'a$

### 52.4 Definitions

**Nest**  $\vdash \forall u \bullet \text{Nest } u \Leftrightarrow (\forall a \ b \bullet a \in u \wedge b \in u \Rightarrow a \subseteq b \vee b \subseteq a)$   
**SubsetClosed**  $\vdash \forall u \bullet \text{SubsetClosed } u \Leftrightarrow (\forall a \ b \bullet a \in u \wedge b \subseteq a \Rightarrow b \in u)$   
**NestClosed**  $\vdash \forall u \bullet \text{NestClosed } u \Leftrightarrow (\forall v \bullet v \subseteq u \wedge \text{Nest } v \Rightarrow \bigcup v \in u)$   
**Maximal $\subseteq$**   $\vdash \forall u \ a$   
 $\bullet \text{Maximal}_{\subseteq} u \ a$   
 $\Leftrightarrow a \in u \wedge (\forall b \bullet b \in u \wedge a \subseteq b \Rightarrow b = a)$   
**Choose**  $\vdash \text{ConstSpec}$   
 $(\lambda \text{Choose}' \bullet \forall a \bullet \neg a = \{\} \Leftrightarrow \text{Choose}' a \in a)$   
 $\text{Choose}$

### 52.5 Theorems

#### Choose\_consistent

$\vdash \text{Consistent } (\lambda \text{Choose}' \bullet \forall a \bullet \neg a = \{\} \Leftrightarrow \text{Choose}' a \in a)$

#### zorn\_thm1

$\vdash \forall u$   
 $\bullet \neg u = \{\} \wedge \text{SubsetClosed } u \wedge \text{NestClosed } u$   
 $\Rightarrow (\exists a \bullet \text{Maximal}_{\subseteq} u \ a)$

## 53 THE THEORY sum

### 53.1 Parents

*combin*

### 53.2 Children

*hol*

### 53.3 Constants

**IsSumRep**  $'a \times 'b \times \text{BOOL} \rightarrow \text{BOOL}$   
**InL**  $'a \rightarrow 'a + 'b$   
**InR**  $'b \rightarrow 'a + 'b$   
**OutL**  $'a + 'b \rightarrow 'a$   
**OutR**  $'a + 'b \rightarrow 'b$   
**IsL**  $'a + 'b \rightarrow \text{BOOL}$   
**IsR**  $'a + 'b \rightarrow \text{BOOL}$

**53.4 Types**

'1 + '2

**53.5 Fixity***Right Infix 300:*

+

**53.6 Definitions****IsSumRep****is\_sum\_rep\_def**

$$\begin{aligned}
&\vdash \exists \text{ inl inr outl outr isl isr} \\
&\bullet \forall x1\ x2\ y1\ y2\ z \\
&\bullet \text{IsSumRep (inl x1)} \\
&\quad \wedge \text{IsSumRep (inr y1)} \\
&\quad \wedge (\text{IsSumRep } z \\
&\quad \quad \Rightarrow \text{inl (outl } z) = z \vee \text{inr (outr } z) = z) \\
&\quad \wedge (\text{inl } x1 = \text{inl } x2 \Leftrightarrow x1 = x2) \\
&\quad \wedge (\text{inr } y1 = \text{inr } y2 \Leftrightarrow y1 = y2) \\
&\quad \wedge \neg \text{inl } x1 = \text{inr } y1 \\
&\quad \wedge \neg \text{inr } y1 = \text{inl } x1 \\
&\quad \wedge \text{outl (inl } x1) = x1 \\
&\quad \wedge \text{outr (inr } y1) = y1 \\
&\quad \wedge (\text{IsSumRep } z \Rightarrow (\text{isl } z \Leftrightarrow \text{inl (outl } z) = z)) \\
&\quad \wedge (\text{IsSumRep } z \Rightarrow (\text{isr } z \Leftrightarrow \text{inr (outr } z) = z))
\end{aligned}$$

+

**sum\_def** $\vdash \exists f \bullet \text{TypeDefn IsSumRep } f$ **InL****InR****OutL****OutR****IsL****IsR****sum\_clauses**

$$\begin{aligned}
&\vdash \forall x1\ x2\ y1\ y2\ z \\
&\bullet (\text{InL (OutL } z) = z \vee \text{InR (OutR } z) = z) \\
&\quad \wedge (\text{InL } x1 = \text{InL } x2 \Leftrightarrow x1 = x2) \\
&\quad \wedge (\text{InR } y1 = \text{InR } y2 \Leftrightarrow y1 = y2) \\
&\quad \wedge \neg \text{InL } x1 = \text{InR } y1 \\
&\quad \wedge \neg \text{InR } y1 = \text{InL } x1 \\
&\quad \wedge \text{OutL (InL } x1) = x1 \\
&\quad \wedge \text{OutR (InR } y1) = y1 \\
&\quad \wedge (\text{IsL } z \Leftrightarrow \text{InL (OutL } z) = z) \\
&\quad \wedge (\text{IsR } z \Leftrightarrow \text{InR (OutR } z) = z)
\end{aligned}$$

### 53.7 Theorems

**sum\_cases\_thm**

$$\vdash \forall x \bullet (\exists y \bullet x = \text{InL } y) \vee (\exists z \bullet x = \text{InR } z)$$
**sum\_fns\_thm**

$$\vdash \forall f \ g \bullet \exists_1 h \bullet h \circ \text{InL} = f \wedge h \circ \text{InR} = g$$

## 54 THE THEORY wrk049

### 54.1 Parents

*lib\_thms*

### 54.2 Children

*fef003*

### 54.3 Constants

$$\$_{E1} \quad 'a \rightarrow 'a \text{ LIST} \rightarrow \text{BOOL}$$

### 54.4 Fixity

*Right Infix 230:* $\in_1$ *Postfix 300:* $\sim$ 

### 54.5 Axioms

**fin\_enumerate\_ax**

$$\begin{aligned} &\vdash \forall s \\ &\bullet s \in \text{Finite} \\ &\Rightarrow \text{Enumerate } s \in \text{Finite} \wedge \# s = \# (\text{Enumerate } s) \end{aligned}$$
**fin\_squash\_eq\_ax**

$$\begin{aligned} &\vdash \forall r_1 \ r_2 \\ &\bullet r_1 \in \text{Finite} \\ &\quad \wedge r_2 \in \text{Finite} \\ &\quad \wedge \text{Squash } r_1 = \text{Squash } r_2 \\ &\Rightarrow \# (\text{Dom } r_1) = \# (\text{Dom } r_2) \end{aligned}$$
**extract\_∧\_single\_ax**

$$\begin{aligned} &\vdash \forall l \ a \ x \\ &\bullet \text{Extract } a \ (l \wedge [x]) \\ &\quad = (\text{if } \# l + 1 \in a \\ &\quad \text{then } \text{Extract } a \ l \wedge [x] \\ &\quad \text{else } \text{Extract } a \ l) \end{aligned}$$
**rel\_list\_⊕\_ax1**

$$\vdash \forall r \ l \ x$$

$$\begin{aligned}
& \bullet r \in \text{Functional} \wedge \text{Dom } r \subseteq \text{Dom } (\text{ListRel } l) \\
& \Rightarrow \text{RelList } (\text{ListRel } (l \hat{\ } [x]) \oplus r) \\
& = \text{RelList } (\text{ListRel } l \oplus r) \hat{\ } [x]
\end{aligned}$$

**rel\_list\_⊕\_ax2**

$$\begin{aligned}
& \vdash \forall r \ l \ x \ y \\
& \bullet r \in \text{Functional} \wedge \text{Dom } r \subseteq \text{Dom } (\text{ListRel } l) \\
& \Rightarrow \text{RelList} \\
& \quad (\text{ListRel } (l \hat{\ } [x]) \oplus r \cup \{(\# \ l + 1, y)\}) \\
& = \text{RelList } (\text{ListRel } l \oplus r) \hat{\ } [y]
\end{aligned}$$

## 54.6 Definitions

$$\begin{aligned}
\in_l & \quad \vdash (\forall x \bullet \neg x \in_l []) \\
& \quad \wedge (\forall h \ t \ x \bullet x \in_l \text{Cons } h \ t \Leftrightarrow x = h \vee x \in_l t)
\end{aligned}$$

## 54.7 Theorems

**one\_one\_inv\_thm**

$$\vdash \forall f \bullet \text{OneOne } f \Rightarrow (\exists g \bullet \forall x \bullet g (f \ x) = x)$$

**fin\_set\_thm5**  $\vdash \forall x \bullet \{x\} \in \text{Finite}$ **dom\_singleton\_thm**

$$\vdash \forall x \bullet \text{Dom } \{x\} = \{\text{Fst } x\}$$

**fin\_dom\_thm**  $\vdash \forall r \bullet r \in \text{Finite} \Rightarrow \text{Dom } r \in \text{Finite}$ **squash\_id\_fin\_thm**

$$\vdash \forall r \bullet r \in \text{Finite} \Rightarrow \text{Squash } (\text{Id } (\text{Dom } r)) \in \text{Finite}$$

**id\_singleton\_thm**

$$\vdash \forall x \bullet \text{Id } \{x\} = \{(x, x)\}$$

**id\_⊔\_thm**  $\vdash \forall a \ b \bullet \text{Id } (a \cup b) = \text{Id } a \cup \text{Id } b$ **fin\_id\_thm**  $\vdash \forall r \bullet r \in \text{Finite} \Rightarrow \text{Id } r \in \text{Finite}$ **dom\_singleton\_thm1**

$$\vdash \forall x \ y \ a \ b \bullet \text{Dom } x = \{y\} \wedge (a, b) \in x \Rightarrow a = y$$

**set\_thm1**  $\vdash \forall n \bullet \{i \mid i = n \wedge i \leq n\} = \{n\}$ **squash\_single\_thm**

$$\begin{aligned}
& \vdash \forall r_1 \ r_2 \ n_1 \ n_2 \\
& \bullet \text{Dom } r_1 = \{n_1\} \\
& \quad \wedge \text{Dom } r_2 = \{n_2\} \\
& \quad \wedge \text{Ran } r_1 = \text{Ran } r_2 \\
& \Rightarrow \text{Squash } r_1 = \text{Squash } r_2
\end{aligned}$$

**enumerate\_singleton\_thm**

$$\vdash \forall x \bullet \text{Enumerate } \{x\} = \{(1, x)\}$$

**⊔\_⊔\_thm**  $\vdash \forall a \ b \ r \bullet a \cup b \ ; r = (a \ ; r) \cup (b \ ; r)$ **⊔\_⊔\_1\_thm1**  $\vdash \forall a \ b \ r \bullet r \ ; a \cup b = (r \ ; a) \cup (r \ ; b)$ **⊔\_singleton\_thm**

$$\vdash \forall x_1 \ x_2 \bullet \{x_1\} \ ; \{x_2\} \in \text{Finite}$$

**fin\_⊔\_thm**

$$\begin{aligned}
& \vdash \forall r_1 \ r_2 \\
& \bullet r_1 \in \text{Finite} \wedge (r_2 \in \text{Functional} \vee r_2 \in \text{Finite}) \\
& \Rightarrow r_1 \ ; r_2 \in \text{Finite}
\end{aligned}$$

**fin\_squash\_thm**

$$\vdash \forall r_1 \bullet r_1 \in \text{Finite} \Rightarrow \text{Squash } r_1 \in \text{Finite}$$

**rel\_combine\_U\_thm**

$$\vdash \forall r_1 r_2 r_3$$

$$\bullet \text{RelCombine } r_1 (r_2 \cup r_3)$$

$$= \text{RelCombine } r_1 r_2 \cup \text{RelCombine } r_1 r_3$$

**rel\_combine\_U\_thm1**

$$\vdash \forall r_1 r_2 r_3$$

$$\bullet \text{RelCombine } (r_1 \cup r_2) r_3$$

$$= \text{RelCombine } r_1 r_3 \cup \text{RelCombine } r_2 r_3$$

**rel\_combine\_singleton\_thm**

$$\vdash \forall x_1 x_2 \bullet \text{RelCombine } \{x_1\} \{x_2\} \in \text{Finite}$$

**fin\_rel\_combine\_thm**

$$\vdash \forall r_1 r_2$$

$$\bullet r_1 \in \text{Finite} \wedge r_2 \in \text{Finite}$$

$$\Rightarrow \text{RelCombine } r_1 r_2 \in \text{Finite}$$

**fin\_D\_thm**

$$\vdash \forall r s \bullet r \in \text{Finite} \Rightarrow r \triangleright s \in \text{Finite}$$

**inv\_rel\_U\_thm**

$$\vdash \forall a b \bullet (a \cup b) \sim = a \sim \cup b \sim$$

**inv\_rel\_singleton\_thm**

$$\vdash \forall x y \bullet \{(x, y)\} \sim = \{(y, x)\}$$

**fin\_inv\_rel\_thm**

$$\vdash \forall r \bullet r \in \text{Finite} \Rightarrow r \sim \in \text{Finite}$$

**squash\_null\_thm**

$$\vdash \text{Squash } \{\} = \{\}$$

**squash\_null\_thm1**

$$\vdash \forall r$$

$$\bullet (\text{Squash } r = \{\} \Leftrightarrow r = \{\}) \wedge (\{\} = \text{Squash } r \Leftrightarrow r = \{\})$$

**dom\_U\_greater\_thm**

$$\vdash \forall r x y n$$

$$\bullet (\forall i j \bullet i \in \text{Dom } r \wedge j \in \text{Dom } x \Rightarrow j > i) \wedge (n, y) \in r$$

$$\Rightarrow \{i \mid i \in \text{Dom } r \wedge i \leq n\}$$

$$= \{i \mid i \in \text{Dom } (r \cup x) \wedge i \leq n\}$$

**squash\_U\_thm**

$$\vdash \forall r_1 r_2 x y$$

$$\bullet \text{Squash } r_1 = \text{Squash } r_2$$

$$\wedge \text{Squash } x = \text{Squash } y$$

$$\wedge r_1 \in \text{Finite}$$

$$\wedge r_2 \in \text{Finite}$$

$$\wedge x \in \text{Finite}$$

$$\wedge y \in \text{Finite}$$

$$\wedge (\forall i j \bullet i \in \text{Dom } r_1 \wedge j \in \text{Dom } x \Rightarrow j > i)$$

$$\wedge (\forall i j \bullet i \in \text{Dom } r_2 \wedge j \in \text{Dom } y \Rightarrow j > i)$$

$$\Rightarrow \text{Squash } (r_1 \cup x) = \text{Squash } (r_2 \cup y)$$

**map\_null\_thm**

$$\vdash \forall f l$$

$$\bullet (\text{Map } f l = [] \Leftrightarrow l = []) \wedge ([] = \text{Map } f l \Leftrightarrow l = [])$$

**¬\_cons\_thm**

$$\vdash \forall l x \bullet \neg \text{Cons } x l = l$$

**^\_Cons\_thm**

$$\vdash \forall a l \bullet [a] \wedge l = \text{Cons } a l$$

**rel\_ext\_clauses**

---

	$\vdash \forall a b$
	<ul style="list-style-type: none"> <li><math>\bullet (a \subseteq b \Leftrightarrow (\forall x y \bullet (x, y) \in a \Rightarrow (x, y) \in b))</math></li> <li><math>\quad \wedge (a = b \Leftrightarrow (\forall x y \bullet (x, y) \in a \Leftrightarrow (x, y) \in b))</math></li> </ul>
<b>pair_eq_thm1</b>	$\vdash \forall x y \bullet Fst x = Fst y \wedge Snd x = Snd y \Leftrightarrow x = y$
<b>rel_thm</b>	$\vdash \forall r r_1 a b$
	<ul style="list-style-type: none"> <li><math>\bullet a \text{ } \textcircled{\%} \text{ } Graph r = b \text{ } \textcircled{\%} \text{ } Graph r</math></li> <li><math>\quad \wedge (\forall x x_1 \bullet r x = r x_1 \Rightarrow r_1 x = r_1 x_1)</math></li> <li><math>\quad \Rightarrow a \text{ } \textcircled{\%} \text{ } Graph r_1 = b \text{ } \textcircled{\%} \text{ } Graph r_1</math></li> </ul>
<b>at_thm1</b>	$\vdash \forall f x y \bullet f \in Functional \wedge (x, y) \in f \Rightarrow f @ x = y$
<b>\%_fun_thm</b>	$\vdash \forall f r$
	<ul style="list-style-type: none"> <li><math>\bullet f \in Functional \wedge Graph r \in Functional</math></li> <li><math>\quad \Rightarrow f \text{ } \textcircled{\%} \text{ } Graph r \in Functional</math></li> </ul>
<b>\%_fun_thm1</b>	$\vdash \forall r_1 r_2$
	<ul style="list-style-type: none"> <li><math>\bullet r_1 \in Functional \wedge r_2 \in Functional</math></li> <li><math>\quad \Rightarrow r_1 \text{ } \textcircled{\%} \text{ } r_2 \in Functional</math></li> </ul>
<b>rel_combine_fun_thm</b>	$\vdash \forall r_1 r_2$
	<ul style="list-style-type: none"> <li><math>\bullet r_1 \in Functional \wedge r_2 \in Functional</math></li> <li><math>\quad \Rightarrow RelCombine r_1 r_2 \in Functional</math></li> </ul>
<b>list_rel_null_thm</b>	$\vdash ListRel [] = \{\}$
<b>\cup_null_thm</b>	$\vdash \forall a b \bullet a \cup b = \{\} \Leftrightarrow a = \{\} \wedge b = \{\}$
<b>map_^\wedge_thm</b>	$\vdash \forall f x l \bullet Map f (l \wedge [x]) = Map f l \wedge [f x]$
<b>\%_graph_null_thm</b>	$\vdash \forall r \bullet \{\} \text{ } \textcircled{\%} \text{ } Graph r = \{\}$
<b>\triangleright_null_thm</b>	$\vdash \forall r \bullet \{\} \triangleright r = \{\}$
<b>\%_null_thm</b>	$\vdash \forall r_1 r_2 \bullet \{\} = r_1 \text{ } \textcircled{\%} \text{ } r_2 \Leftrightarrow Ran r_1 \cap Dom r_2 = \{\}$
<b>\%_null_thm1</b>	$\vdash \forall r_1 r_2 \bullet r_1 \text{ } \textcircled{\%} \text{ } r_2 = \{\} \Leftrightarrow Ran r_1 \cap Dom r_2 = \{\}$
<b>dom_null_thm</b>	$\vdash \forall r \bullet Dom r = \{\} \Leftrightarrow r = \{\}$
<b>id_dom_null_thm</b>	$\vdash Id (Dom \{\}) = \{\}$
<b>inv_rel_\%_null_thm</b>	$\vdash \forall r \bullet \{\} \sim \text{ } \textcircled{\%} \text{ } r = \{\}$
<b>rel_combine_null_thm</b>	$\vdash \forall r \bullet RelCombine \{\} r = \{\}$
<b>rel_combine_null_thm1</b>	$\vdash \forall r \bullet RelCombine r \{\} = \{\}$
<b>\triangleright_singleton_thm</b>	$\vdash \forall a b s$
	<ul style="list-style-type: none"> <li><math>\bullet \{(a, b)\} \triangleright s = (\text{if } b \in s \text{ then } \{(a, b)\} \text{ else } \{\})</math></li> </ul>
<b>list_rel_fun_thm</b>	$\vdash \forall l \bullet ListRel l \in Functional$
<b>list_rel_\triangleright_fun_thm</b>	$\vdash \forall l s \bullet ListRel l \triangleright s \in Functional$
<b>rel_combine_null_thm2</b>	$\vdash \forall r_1 r_2$
	<ul style="list-style-type: none"> <li><math>\bullet Dom r_1 \cap Dom r_2 = \{\} \Rightarrow RelCombine r_1 r_2 = \{\}</math></li> </ul>
<b>dom_thm</b>	$\vdash \forall x r \bullet x \in Dom r \Leftrightarrow (\exists y \bullet (x, y) \in r)$

---

---

**ran\_thm**  $\vdash \forall y r \bullet y \in \text{Ran } r \Leftrightarrow (\exists x \bullet (x, y) \in r)$   
**fin\_list\_rel\_thm**  $\vdash \forall l \bullet \text{ListRel } l \in \text{Finite}$   
**fin\_list\_rel\_▷\_thm**  $\vdash \forall l s \bullet \text{ListRel } l \triangleright s \in \text{Finite}$   
**length\_∧\_one\_thm**  $\vdash \forall l x \bullet \# (l \wedge [x]) = \# l + 1$   
**size\_list\_rel\_thm**  $\vdash \forall l \bullet \# (\text{ListRel } l) = \# l$   
**size\_squash\_thm**  $\vdash \forall r_1 r_2$   
 $\bullet r_1 \in \text{Functional}$   
 $\wedge r_2 \in \text{Functional}$   
 $\wedge r_1 \in \text{Finite}$   
 $\wedge r_2 \in \text{Finite}$   
 $\wedge \# r_1 = \# r_2$   
 $\Rightarrow \# (\text{Squash } (\text{Id } (\text{Dom } r_1)))$   
 $= \# (\text{Squash } (\text{Id } (\text{Dom } r_2)))$   
**size\_squash\_id\_dom\_thm**  $\vdash \forall l l' s$   
 $\bullet \# (\text{ListRel } l \triangleright s) = \# (\text{ListRel } l' \triangleright s)$   
 $\Rightarrow \# (\text{Squash } (\text{Id } (\text{Dom } (\text{ListRel } l \triangleright s))))$   
 $= \# (\text{Squash } (\text{Id } (\text{Dom } (\text{ListRel } l' \triangleright s))))$   
**⊕\_thm**  $\vdash \forall x y r s$   
 $\bullet (x, y) \in r \oplus s$   
 $\Leftrightarrow \neg x \in \text{Dom } s \wedge (x, y) \in r \vee (x, y) \in s$   
**length\_single\_thm**  $\vdash \forall x \bullet \# [x] = 1$   
**⊕\_single**  $\vdash \forall f x y a b$   
 $\bullet (x, y) \in f \oplus \{(a, b)\}$   
 $\Leftrightarrow (x, y) = (a, b) \vee (x, y) \in f \wedge \neg x = a$   
**set\_dif\_∪\_thm**  $\vdash \forall a b c \bullet a \setminus (b \cup c) = (a \setminus b) \cap (a \setminus c)$   
**image\_∪\_thm**  $\vdash \forall a b s \bullet (a \cup b) \text{ Image } s = a \text{ Image } s \cup b \text{ Image } s$   
**∪\_∩\_thm**  $\vdash \forall a b c \bullet (a \cup b) \cap c = a \cap c \cup b \cap c$   
**◁\_thm**  $\vdash \forall x y s r \bullet (x, y) \in s \triangleleft r \Leftrightarrow \neg x \in s \wedge (x, y) \in r$   
**◁\_null\_thm**  $\vdash \forall s \bullet s \triangleleft \{\} = \{\}$   
**◁\_null\_thm**  $\vdash \forall s \bullet s \triangleleft \{\} = \{\}$   
**∧\_|\_thm**  $\vdash \forall l_1 l_2 s \bullet (l_1 \wedge l_2) \upharpoonright s = (l_1 \upharpoonright s) \wedge (l_2 \upharpoonright s)$   
**list\_rel\_∧\_▷\_thm**  $\vdash \forall l x s$   
 $\bullet \text{ListRel } (l \wedge [x]) \triangleright s$   
 $= (\text{if } x \in s$   
 $\text{then } (\text{ListRel } l \triangleright s) \cup \{(\# (\text{ListRel } l) + 1, x)\}$   
 $\text{else } \text{ListRel } l \triangleright s)$   
**size\_list\_rel\_∧\_▷\_thm**  $\vdash \forall l x s$

---

---

	<ul style="list-style-type: none"> <li>• <math>\# (ListRel (l \hat{\ } [x]) \triangleright s)</math>  <math>= (if\ x \in s</math>  <math>\quad then\ \# (ListRel\ l \triangleright s) + 1</math>  <math>\quad else\ \# (ListRel\ l \triangleright s))</math></li> </ul>
<b>squash_<math>\hat{\}</math>_thm</b>	$\vdash \forall l\ x\ s$ <ul style="list-style-type: none"> <li>• <math>Squash (Id (Dom (ListRel (l \hat{\ } [x]) \triangleright s)))</math>  <math>= (if\ \neg x \in s</math>  <math>\quad then\ Squash (Id (Dom (ListRel\ l \triangleright s)))</math>  <math>\quad else</math>  <math>\quad Squash (Id (Dom (ListRel\ l \triangleright s)))</math>  <math>\quad \cup \{(\# (Squash (Id (Dom (ListRel\ l \triangleright s))))</math>  <math>\quad \quad + 1, \# l + 1)\}</math></li> </ul>
<b><math>\oplus</math>_null_thm</b>	$\vdash \{\} \oplus \{\} = \{\}$
<b><math>\triangleleft</math>_U_thm</b>	$\vdash \forall a\ b\ r \bullet a \cup b \triangleleft r = (a \triangleleft r) \cup (b \triangleleft r)$
<b>map_<math>\hat{\}</math>_thm1</b>	$\vdash \forall f\ l_1\ l_2$ <ul style="list-style-type: none"> <li>• <math>Map\ f (l_1 \hat{\ } l_2) = Map\ f\ l_1 \hat{\ } Map\ f\ l_2</math></li> </ul>
<b>nth_<math>\hat{\}</math>_thm</b>	$\vdash \forall l\ x\ n$ <ul style="list-style-type: none"> <li>• <math>n \in 1 .. \# l + 1</math>  <math>\Rightarrow Nth (l \hat{\ } [x])\ n</math>  <math>= (if\ n = \# l + 1\ then\ x\ else\ Nth\ l\ n)</math></li> </ul>
<b>nth_length_one_thm</b>	$\vdash \forall l\ last \bullet Nth (l \hat{\ } [last]) (\# l + 1) = last$
<b>image_single_thm</b>	$\vdash \forall a\ b\ s$ <ul style="list-style-type: none"> <li>• <math>\{(a, b)\} Image\ s = (if\ a \in s\ then\ \{b\}\ else\ \{\})</math></li> </ul>
<b>length_set_def_thm</b>	$\vdash \forall l\ x \bullet 1 .. \# (l \hat{\ } [x]) \setminus \{\# l + 1\} = 1 .. \# l$
<b>length_<math>\cap</math>_thm</b>	$\vdash \forall l\ s \bullet \neg \# l + 1 \in s \cap 1 .. \# l$
<b>nth_<math>\hat{\}</math>_thm1</b>	$\vdash \forall l\ x\ last$ <ul style="list-style-type: none"> <li>• <math>x \in 1 .. \# l \Rightarrow Nth (l \hat{\ } [last])\ x = Nth\ l\ x</math></li> </ul>
<b><math>\neg</math>_≤_plus1_thm</b>	$\vdash \forall m\ n \bullet \neg (m \leq n \wedge m = n + 1)$
<b>≤_plus_one_thm</b>	$\vdash \forall l\ x \bullet x \leq \# l \Rightarrow x \leq \# l + 1$
<b><math>\oplus</math>_single_thm1</b>	$\vdash \forall x\ y\ g \bullet x \in Dom\ g \Rightarrow \{(x, y)\} \oplus g = g$
<b><math>\oplus</math>_single_thm2</b>	$\vdash \forall x\ y\ g \bullet \neg x \in Dom\ g \Rightarrow \{(x, y)\} \oplus g = \{(x, y)\} \cup g$
<b>dot_dot_single_thm</b>	$\vdash \forall x\ n \bullet x \in n .. n \Leftrightarrow x = n$
<b><math>\hat{\}</math>_thm</b>	$\vdash \forall l\ x\ y \bullet (l \hat{\ } [x]) \hat{\ } [y] = l \hat{\ } [x] \hat{\ } [y]$
<b><math>\hat{\}</math>_thm1</b>	$\vdash \forall l\ x\ y \bullet (l \hat{\ } [x]) \hat{\ } [y] = l \hat{\ } [x; y]$
<b>list_rel_list_thm</b>	$\vdash \forall l\ s \bullet ListRel\ l \triangleright s = \{\} \Leftrightarrow l \upharpoonright s = []$
<b>list_rel_<math>\oplus</math>_thm</b>	

---

---

$\vdash \forall l x s$   
 $\bullet ListRel (l \hat{\ } [x]) \oplus s$   
 $= (ListRel l \oplus s) \cup (\{(\# l + 1, x)\} \oplus s)$

**$\oplus\_null\_thm1$**   $\vdash \forall f g \bullet f \oplus g = \{\} \Leftrightarrow f = \{\} \wedge g = \{\}$

**$\oplus\_null\_thm2$**   $\vdash \forall f \bullet f \oplus \{\} = f$

**list\_rel\_null\_thm1**  
 $\vdash \forall l \bullet ListRel l = \{\} \Leftrightarrow l = []$

**list\_rel\_null\_thm2**  
 $\vdash \forall l \bullet \{\} = ListRel l \Leftrightarrow l = []$

**dot\_dot\_eq\_thm**  
 $\vdash \forall n_1 n_2 \bullet 1 .. n_1 = 1 .. n_2 \Leftrightarrow n_1 = n_2$

**size\_list\_rel\_eq\_thm**  
 $\vdash \forall l_1 l_2 \bullet ListRel l_1 = ListRel l_2 \Rightarrow \# l_1 = \# l_2$

**list\_rel\_^\\_eq\_thm**  
 $\vdash \forall l_1 l_2 x_1 x_2$   
 $\bullet ListRel (l_1 \hat{\ } [x_1]) = ListRel (l_2 \hat{\ } [x_2])$   
 $\Rightarrow ListRel l_1 = ListRel l_2 \wedge x_1 = x_2$

**list\_rel\_one\_one\_thm**  
 $\vdash OneOne ListRel$

**RelList\_consistent**  
 $\vdash Consistent$   
 $(\lambda RelList' \bullet \forall l \bullet RelList' (ListRel l) = l)$

**rel\_list\_null\_thm**  
 $\vdash RelList \{\} = []$

**distinct\_single\_thm**  
 $\vdash \forall x \bullet [x] \in Distinct$

**elems\_^\\_thm**  $\vdash \forall l x \bullet Elems (l \hat{\ } [x]) = \{x\} \cup Elems l$

**distinct\_^\\_thm**  
 $\vdash \forall l x$   
 $\bullet l \hat{\ } [x] \in Distinct \Leftrightarrow \neg x \in Elems l \wedge l \in Distinct$

**size\_squash\_plus1\_thm**  
 $\vdash \forall l x s$   
 $\bullet x \in s$   
 $\Rightarrow \# (Squash (Id (Dom (ListRel (l \hat{\ } [x]) \triangleright s))))$   
 $= \# (Squash (Id (Dom (ListRel l \triangleright s)))) + 1$

**size\_^\\_one\_thm**  
 $\vdash \forall l x s$   
 $\bullet x \in s$   
 $\Rightarrow (\# (Squash (Id (Dom (ListRel (l \hat{\ } [x]) \triangleright s)))),$   
 $\# l + 1)$   
 $\in Squash (Id (Dom (ListRel (l \hat{\ } [x]) \triangleright s)))$

**$\uparrow\_null\_thm$**   $\vdash \forall l \bullet l \uparrow \{\} = []$

**$\uparrow\_extract\_null\_thm$**   
 $\vdash \forall l s a \bullet l \uparrow s = [] \Rightarrow Extract a l \uparrow s = []$

**$\in_1\_^\_thm$**   $\vdash \forall x l_1 l_2 \bullet x \in_l l_1 \hat{\ } l_2 \Leftrightarrow x \in_l l_1 \vee x \in_l l_2$

**$\in_1\_elems\_thm$**

---

$$\vdash \forall x l \bullet x \in_l l \Leftrightarrow x \in \text{Elems } l$$

$$\in_1\text{-extract\_thm}$$

$$\vdash \forall x a l \bullet x \in_l \text{Extract } a l \Rightarrow x \in_l l$$

## 55 THE THEORY wrk057

### 55.1 Parents

$$\text{lib\_thms}$$

### 55.2 Children

$$\text{fef042}$$

### 55.3 Constants

**FinitaryRecType**

$$('D \rightarrow 'T) \\ \Leftrightarrow (('D \rightarrow 'T \mathbb{P}) \times ('T \rightarrow \mathbb{N}) \times (('T \rightarrow 'y) \rightarrow 'D \rightarrow 'Y))$$

**LocalFunctional**

$$(('T \rightarrow 'y) \rightarrow 'D \rightarrow 'Y) \Leftrightarrow ('D \rightarrow 'T \mathbb{P})$$

**Tree**

$$(\mathbb{N} \times 'a) \text{ LIST } \mathbb{P}$$

**Unparse**

$$(\mathbb{N} \times 'a) \text{ LIST } \rightarrow \mathbb{N} \rightarrow (\mathbb{N} \times 'a) \text{ LIST}$$

**MkTree**

$$'a \times 'a \text{ TREE LIST } \rightarrow 'a \text{ TREE}$$

### 55.4 Types

'1 TREE

### 55.5 Definitions

**FinitaryRecType**

$$\vdash \text{FinitaryRecType} \\ = \{(k, c, w, M) \\ | \text{OneOne } k \\ \wedge (\forall t \bullet \exists x \bullet c x \subseteq \{z | w z < w t\} \wedge t = k x) \\ \wedge (\forall i g_1 g_2 \\ \bullet (\forall y \bullet w y < i \Rightarrow g_1 y = g_2 y) \\ \Rightarrow (\forall x \\ \bullet c x \subseteq \{y | w y < i\} \Rightarrow M g_1 x = M g_2 x))\}$$

**LocalFunctional**

$$\vdash \text{LocalFunctional} \\ = \{(M, c) \\ | \forall I g_1 g_2 \\ \bullet (\forall y \bullet y \in I \Rightarrow g_1 y = g_2 y) \\ \Rightarrow (\forall x \bullet c x \subseteq I \Rightarrow M g_1 x = M g_2 x)\}$$

---

**Tree**  $\vdash Tree$   
 $= \bigcap$   
 $\{A$   
 $|\forall lab\ trees$   
 $\bullet Elems\ trees \subseteq A$   
 $\Rightarrow Cons\ (\#\ trees, lab)\ (Flat\ trees) \in A\}$

**Unparse**  $\vdash \forall i\ nv\ more$   
 $\bullet Unparse\ []\ i = []$   
 $\wedge Unparse\ (Cons\ nv\ more)\ i$   
 $= (if\ i = 0$   
 $then\ []$   
 $else$   
 $Cons\ nv\ (Unparse\ more\ ((Fst\ nv + i) - 1)))$

**TREE**  
**tree\_def**  $\vdash \exists f \bullet TypeDefn\ (\lambda t \bullet t \in Tree)\ f$   
**MkTree**  $\vdash ConstSpec$   
 $(\lambda MkTree'$   
 $\bullet OneOne\ MkTree'$   
 $\wedge (\exists w$   
 $\bullet \forall t$   
 $\bullet \exists x$   
 $\bullet (\forall z \bullet z \in Elems\ (Snd\ x) \Rightarrow w\ z < w\ t)$   
 $\wedge t = MkTree'\ x))$   
 $MkTree$

## 55.6 Theorems

**$\wedge\_empty\_thm$**   $\vdash \forall l1\ l2 \bullet l1 \wedge l2 = [] \Leftrightarrow l1 = [] \wedge l2 = []$

**flat\_empty\_thm**

$\vdash \forall ls \bullet Flat\ ls = [] \Leftrightarrow Elems\ ls \subseteq \{[]\}$

**append\_assoc\_thm**

$\vdash \forall l1\ l2\ l3 \bullet (l1 \wedge l2) \wedge l3 = l1 \wedge l2 \wedge l3$

**flat\_append\_thm**

$\vdash \forall ls1\ ls2 \bullet Flat\ (ls1 \wedge ls2) = Flat\ ls1 \wedge Flat\ ls2$

**length\_append\_thm**

$\vdash \forall ls1\ ls2 \bullet \# (ls1 \wedge ls2) = \# ls1 + \# ls2$

**elems\_append\_thm**

$\vdash \forall l1\ l2 \bullet Elems\ (l1 \wedge l2) = Elems\ l1 \cup Elems\ l2$

**append\_empty\_thm**

$\vdash \forall l \bullet l \wedge [] = l$

**append\_cancel\_thm**

$\vdash \forall l1\ l2\ l3 \bullet l1 \wedge l2 = l1 \wedge l3 \Leftrightarrow l2 = l3$

**map\_map\_id\_thm**

$\vdash \forall f\ g\ l$

$\bullet (\forall x \bullet x \in Elems\ l \Rightarrow f\ (g\ x) = x)$   
 $\Rightarrow Map\ f\ (Map\ g\ l) = l$

**length\_length\_flat\_thm**

$$\vdash \forall ll \bullet l \in \text{Elems } ll \Rightarrow \# l \leq \# (\text{Flat } ll)$$
**elems\_map\_thm**

$$\vdash \forall f l$$

- $\text{Elems } (\text{Map } f l) = \{y \mid \exists x \bullet x \in \text{Elems } l \wedge f x = y\}$

**length\_0\_thm**  $\vdash \forall l \bullet \# l = 0 \Leftrightarrow l = []$ **one\_one\_left\_inv\_thm**

$$\vdash \forall f \bullet \text{OneOne } f \Rightarrow (\exists g \bullet \forall x \bullet g (f x) = x)$$
**onto\_right\_inv\_thm**

$$\vdash \forall f \bullet \text{Onto } f \Rightarrow (\exists g \bullet \forall y \bullet f (g y) = y)$$
**one\_one\_onto\_inv\_thm**

$$\vdash \forall f$$

- $\text{OneOne } f \wedge \text{Onto } f$

$$\Rightarrow (\exists g \bullet (\forall x \bullet g (f x) = x) \wedge (\forall y \bullet f (g y) = y))$$
**fin\_rec\_type\_induction\_thm**

$$\vdash \forall k c w M$$

- $(k, c, w, M) \in \text{FinitaryRecType}$

$$\Rightarrow (\forall X \bullet (\forall x \bullet c x \subseteq X \Rightarrow k x \in X) \Rightarrow (\forall t \bullet t \in X))$$
**fin\_rec\_type\_induction\_thm1**

$$\vdash \forall k c w M$$

- $(k, c, w, M) \in \text{FinitaryRecType}$

$$\Rightarrow (\forall P$$

- $(\forall x \bullet (\forall t \bullet t \in c x \Rightarrow P t) \Rightarrow P (k x))$

$$\Rightarrow (\forall t \bullet P t))$$
**fin\_rec\_type\_prim\_rec\_lemma1**

$$\vdash \forall k c w$$

- $(k, c, w, M) \in \text{FinitaryRecType}$

$$\Rightarrow (\exists \delta$$

- $(\forall x \bullet \delta (k x) = x)$

$$\wedge (\forall y \bullet k (\delta y) = y)$$

$$\wedge (\forall d$$

- $\exists H$

- $(\forall y \bullet H 0 y = d (\delta y, M \text{Arbitrary } (\delta y)))$

$$\wedge (\forall i y$$

- $H (i + 1) y$

$$= (\text{if } w y \leq i$$

$$\text{then } H i y$$

$$\text{else } d (\delta y, M (H i) (\delta y)))$$

$$\wedge (\forall j y \bullet H (w y + j) y = H (w y) y)))$$
**fin\_rec\_type\_prim\_rec\_exists\_thm**

$$\vdash \forall k c w M$$

- $(k, c, w, M) \in \text{FinitaryRecType}$

$$\Rightarrow (\forall d \bullet \exists h \bullet \forall x \bullet h (k x) = d (x, M h x))$$
**fin\_rec\_type\_prim\_rec\_unique\_thm**

$$\vdash \forall k c w M$$

- $(k, c, w, M) \in \text{FinitaryRecType}$

$$\Rightarrow (\forall d h_1 h_2$$

- $(\forall x \bullet h_1 (k x) = d (x, M h_1 x))$

$$\wedge (\forall x \bullet h_2 (k x) = d (x, M h_2 x))$$

$$\Rightarrow h_1 = h_2)$$

**fin\_rec\_type\_prim\_rec\_thm**

$$\begin{aligned} &\vdash \forall k \ c \ w \ M \\ &\bullet (k, c, w, M) \in \text{FinitaryRecType} \\ &\Rightarrow (\forall d \bullet \exists_1 h \bullet \forall x \bullet h (k \ x) = d \ x \ (M \ h \ x)) \end{aligned}$$

**local\_functional\_thm**

$$\begin{aligned} &\vdash \forall c \ w \ M \\ &\bullet (M, c) \in \text{LocalFunctional} \\ &\Rightarrow (\forall i \ g_1 \ g_2 \\ &\bullet (\forall y \bullet w \ y < i \Rightarrow g_1 \ y = g_2 \ y) \\ &\Rightarrow (\forall x \\ &\bullet c \ x \subseteq \{y \mid w \ y < i\} \Rightarrow M \ g_1 \ x = M \ g_2 \ x)) \end{aligned}$$

$$\mathbf{i\_local\_thm} \quad \vdash ((\lambda f \bullet f), (\lambda x \bullet \{x\})) \in \text{LocalFunctional}$$

$$\mathbf{k\_local\_thm} \quad \vdash ((\lambda f \ x \bullet \{ \}), (\lambda x \bullet \{x\})) \in \text{LocalFunctional}$$

$$\begin{aligned} \mathbf{\times\_local\_thm} \quad &\vdash \forall M_1 \ c_1 \ M_2 \ c_2 \\ &\bullet (M_1, c_1) \in \text{LocalFunctional} \\ &\quad \wedge (M_2, c_2) \in \text{LocalFunctional} \\ &\Rightarrow ((\lambda f \ (x, y) \bullet (M_1 \ f \ x, M_2 \ f \ y)), \\ &\quad (\lambda (x, y) \bullet c_1 \ x \cup c_2 \ y)) \\ &\in \text{LocalFunctional} \end{aligned}$$

**sum\_local\_thm**

$$\begin{aligned} &\vdash \forall M_1 \ c_1 \ M_2 \ c_2 \\ &\bullet (M_1, c_1) \in \text{LocalFunctional} \\ &\quad \wedge (M_2, c_2) \in \text{LocalFunctional} \\ &\Rightarrow ((\lambda f \ x \\ &\bullet \text{if } \text{IsL } x \\ &\quad \text{then } M_1 \ f \ (\text{OutL } x) \\ &\quad \text{else } M_2 \ f \ (\text{OutR } x)), \\ &(\lambda x \\ &\bullet \text{if } \text{IsL } x \\ &\quad \text{then } c_1 \ (\text{OutL } x) \\ &\quad \text{else } c_2 \ (\text{OutR } x))) \\ &\in \text{LocalFunctional} \end{aligned}$$

**list\_local\_thm**

$$\begin{aligned} &\vdash \forall M \ c \\ &\bullet (M, c) \in \text{LocalFunctional} \\ &\Rightarrow ((\lambda f \bullet \text{Map } (M \ f)), (\lambda x \bullet \cup (\text{Elems } (\text{Map } c \ x)))) \\ &\in \text{LocalFunctional} \end{aligned}$$

**list\_local\_thm1**

$$\vdash (\text{Map}, \text{Elems}) \in \text{LocalFunctional}$$

**o\_snd\_local\_thm**

$$\begin{aligned} &\vdash \forall M \ c \\ &\bullet (M, c) \in \text{LocalFunctional} \\ &\Rightarrow ((\lambda f \bullet M \ f \ o \ \text{Snd}), c \ o \ \text{Snd}) \in \text{LocalFunctional} \end{aligned}$$

**o\_fst\_local\_thm**

$$\begin{aligned} &\vdash \forall M \ c \\ &\bullet (M, c) \in \text{LocalFunctional} \\ &\Rightarrow ((\lambda f \bullet M \ f \ o \ \text{Fst}), c \ o \ \text{Fst}) \in \text{LocalFunctional} \end{aligned}$$

**tree\_induction\_lemma1**

- $$\vdash \forall X$$
- $(\forall x ts$ 
    - $Elms\ ts \subseteq X \Rightarrow Cons\ (\#\ ts, x)\ (Flat\ ts) \in X$ $\Rightarrow Tree \subseteq X$

**tree\_induction\_lemma2**

- $$\vdash \forall x ts$$
- $Elms\ ts \subseteq Tree \Rightarrow Cons\ (\#\ ts, x)\ (Flat\ ts) \in Tree$

**tree\_induction\_lemma**

- $$\vdash \forall X$$
- $(\forall x ts$ 
    - $Elms\ ts \subseteq Tree \cap X$
    - $\Rightarrow Cons\ (\#\ ts, x)\ (Flat\ ts) \in X$ $\Rightarrow Tree \subseteq X$

**tree\_induction\_tac\_lemma**

- $$\vdash \forall P$$
- $(\forall x ts$ 
    - $(\forall t \bullet t \in Elms\ ts \Rightarrow t \in Tree \wedge P\ t)$
    - $\Rightarrow P\ (Cons\ (\#\ ts, x)\ (Flat\ ts))$ $\Rightarrow (\forall t \bullet t \in Tree \Rightarrow P\ t)$

**tree\_cases\_lemma**

- $$\vdash \forall t$$
- $t \in Tree$
  - $\Rightarrow (\exists x ts$ 
    - $Elms\ ts \subseteq Tree \wedge t = Cons\ (\#\ ts, x)\ (Flat\ ts))$

 **$\neg$ \_empty\_list\_tree\_lemma**

- $$\vdash \neg [] \in Tree$$

**unparse\_thm**

- $$\vdash \forall ts\ more$$
- $Elms\ ts \subseteq Tree$
  - $\Rightarrow Unparse\ (Flat\ ts \wedge more)\ (\#\ ts) = Flat\ ts$

**unparse\_thm1**  $\vdash \forall t\ more \bullet t \in Tree \Rightarrow Unparse\ (t \wedge more)\ 1 = t$ **tree\_cases\_lemma1**

- $$\vdash \forall t$$
- $t \in Tree$
  - $\Rightarrow (\exists_1 (x, ts)$ 
    - $Elms\ ts \subseteq Tree \wedge t = Cons\ (\#\ ts, x)\ (Flat\ ts))$

**leaf\_is\_a\_tree\_thm**

- $$\vdash \forall x \bullet [(0, x)] \in Tree$$

**MkTree\_consistent**

- $$\vdash Consistent$$
- $(\lambda MkTree'$ 
    - $OneOne\ MkTree'$
    - $\wedge (\exists w$ 
      - $\forall t$
      - $\exists x$ 
        - $(\forall z \bullet z \in Elms\ (Snd\ x) \Rightarrow w\ z < w\ t)$
        - $\wedge t = MkTree'\ x)$

**tree\_local\_thm**

---

$\vdash ((\lambda f \bullet \text{Map } f \circ \text{Snd}), \text{Elems } o \text{ Snd}) \in \text{LocalFunctional}$   
**tree\_fin\_rec\_thm**  
 $\vdash \exists w$   
 $\bullet (\text{MkTree}, \text{Elems } o \text{ Snd}, w, (\lambda f \bullet \text{Map } f \circ \text{Snd}))$   
 $\in \text{FinitaryRecType}$   
**tree\_induction\_thm**  
 $\vdash \forall P$   
 $\bullet (\forall x$   
 $\bullet (\forall t \bullet t \in \text{Elems } (\text{Snd } x) \Rightarrow P t) \Rightarrow P (\text{MkTree } x))$   
 $\Rightarrow (\forall t \bullet P t)$   
**tree\_prim\_rec\_thm**  
 $\vdash \forall d \bullet \exists_1 h \bullet \forall x \bullet h (\text{MkTree } x) = d x (\text{Map } h (\text{Snd } x))$

## 56 THE THEORY $\mathbb{R}$

### 56.1 Parents

*orders dyadic*

### 56.2 Constants

<b>Is<sub>R</sub>Rep</b>	$DYADIC SET \rightarrow BOOL$
<b>\$&lt;<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow BOOL$
<b>\$≤<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow BOOL$
<b>\$&gt;<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow BOOL$
<b>\$≥<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow BOOL$
<b>Sup</b>	$\mathbb{R} SET \rightarrow \mathbb{R}$
<b>1<sub>R</sub></b>	$\mathbb{R}$
<b>0<sub>R</sub></b>	$\mathbb{R}$
<b>\$+<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
<b>~<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R}$
<b>\$-<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
<b>N<sub>R</sub></b>	$\mathbb{N} \rightarrow \mathbb{R}$
<b>Abs<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R}$
<b>\$*<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
<b>\$/<sub>R</sub></b>	$\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
<b>\$/<sub>N</sub></b>	$\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{R}$
<b>\$<sup>-1</sup></b>	$\mathbb{R} \rightarrow \mathbb{R}$
<b>\$<sup>^</sup><sub>N</sub></b>	$\mathbb{R} \rightarrow \mathbb{N} \rightarrow \mathbb{R}$
<b>Z<sub>R</sub></b>	$\mathbb{Z} \rightarrow \mathbb{R}$
<b>\$<sup>^</sup><sub>Z</sub></b>	$\mathbb{R} \rightarrow \mathbb{Z} \rightarrow \mathbb{R}$
<b>Float</b>	$\mathbb{N} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{R}$
<b>Max<sub>R</sub></b>	$\mathbb{R} LIST \rightarrow \mathbb{R}$
<b>Min<sub>R</sub></b>	$\mathbb{R} LIST \rightarrow \mathbb{R}$

### 56.3 Aliases

<	$\$<_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \text{BOOL}$
≤	$\$\leq_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \text{BOOL}$
>	$\$>_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \text{BOOL}$
≥	$\$\geq_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \text{BOOL}$
+	$\$+_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
~	$\sim_R : \mathbb{R} \rightarrow \mathbb{R}$
-	$\$-_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
<b>Abs</b>	$Abs_R : \mathbb{R} \rightarrow \mathbb{R}$
*	$\$*_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
/	$\$/_R : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
/	$\$/_N : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{R}$
^	$\$\hat{^}_N : \mathbb{R} \rightarrow \mathbb{N} \rightarrow \mathbb{R}$
^	$\$\hat{^}_Z : \mathbb{R} \rightarrow \mathbb{Z} \rightarrow \mathbb{R}$

### 56.4 Types

 $\mathbb{R}$ 

### 56.5 Fixity

*Left Infix 305:* $-_R$ *Left Infix 315:* $/ \quad /_N \quad /_R$ *Right Infix 210:* $<_R \quad >_R \quad \leq_R \quad \geq_R$ *Right Infix 300:* $+_R$ *Right Infix 310:* $*_R$ *Right Infix 320:* $\hat{^}_N \quad \hat{^}_Z$ *Postfix 320:* $-1$ 

### 56.6 Definitions

**Is\_ℝ\_Rep**  $\vdash \forall a \bullet \text{Is\_}\mathbb{R}\text{-Rep } a \Leftrightarrow a \in \text{Cuts } (\text{Universe}, \$dy\_less)$  $\mathbb{R}$ **ℝ\_def**  $\vdash \exists f \bullet \text{TypeDefn Is\_}\mathbb{R}\text{-Rep } f$  $<_R$   $\vdash \text{ConstSpec}$  $(\lambda <'_R$ 

- *StrictLinearOrder* (*Universe*,  $<'_R$ )
  - $\wedge$  *UnboundedBelow* (*Universe*,  $<'_R$ )
  - $\wedge$  *UnboundedAbove* (*Universe*,  $<'_R$ )
  - $\wedge$  *Complete* (*Universe*,  $<'_R$ )

---

	$\wedge (\exists \iota$ $\bullet (\forall a b \bullet <'_R (\iota a) (\iota b) \Leftrightarrow a \text{ dy\_less } b)$ $\wedge \{x   \exists a \bullet \iota a = x\}$ $\text{DenseIn (Universe, <'_R))})$
	\$<
$\leq_R$	$\vdash \forall x y \bullet x \leq y \Leftrightarrow x < y \vee x = y$
$>_R$	$\vdash \forall x y \bullet x > y \Leftrightarrow y < x$
$\geq_R$	$\vdash \forall x y \bullet x \geq y \Leftrightarrow y \leq x$
<b>Sup</b>	$\vdash \text{ConstSpec}$ $(\lambda \text{Sup}'$ $\bullet \forall A$ $\bullet \neg A = \{\} \wedge (\exists b \bullet \forall x \bullet x \in A \Rightarrow x \leq b)$ $\Rightarrow (\forall x \bullet x \in A \Rightarrow x \leq \text{Sup}' A)$ $\wedge (\forall b$ $\bullet (\forall x \bullet x \in A \Rightarrow x \leq b) \Rightarrow \text{Sup}' A \leq b))$
	Sup
$+_R$	
<b>0<sub>R</sub></b>	
<b>1<sub>R</sub></b>	$\vdash \text{ConstSpec}$ $(\lambda (+'_R, 0'_R, 1'_R)$ $\bullet (\forall x y z$ $\bullet +'_R (+'_R x y) z = +'_R x (+'_R y z))$ $\wedge (\forall x y \bullet +'_R x y = +'_R y x)$ $\wedge (\forall x \bullet +'_R x 0'_R = x)$ $\wedge (\forall x \bullet \exists y \bullet +'_R x y = 0'_R)$ $\wedge (\forall x y z \bullet y < z \Rightarrow +'_R x y < +'_R x z)$ $\wedge 0'_R < 1'_R)$ $(\$+, 0_R, 1_R)$
$\sim_R$	$\vdash \text{ConstSpec } (\lambda \sim'_R \bullet \forall x \bullet x + \sim'_R x = 0_R) \sim$
$-_R$	$\vdash \forall x y \bullet x - y = x + \sim y$
<b>N<sub>R</sub></b>	$\vdash 0. = 0_R \wedge (\forall m \bullet \text{NR} (m + 1) = \text{NR} m + 1_R)$
<b>Abs<sub>R</sub></b>	$\vdash \forall x \bullet \text{Abs } x = (\text{if } 0. \leq x \text{ then } x \text{ else } \sim x)$
<b>*<sub>R</sub></b>	$\vdash \text{ConstSpec}$ $(\lambda *'_R$ $\bullet (\forall x y z$ $\bullet *'_R (*'_R x y) z = *'_R x (*'_R y z))$ $\wedge (\forall x \bullet *'_R x 1. = x)$ $\wedge (\forall x y z$ $\bullet *'_R x (y + z) = *'_R x y + *'_R x z)$ $\wedge (\forall x y \bullet *'_R x y = *'_R y x)$ $\wedge (\forall x y \bullet 0. < x \wedge 0. < y \Rightarrow 0. < *'_R x y))$
	\$*
$/_R$	$\vdash \text{ConstSpec}$ $(\lambda /'_R$ $\bullet (\forall y z \bullet \neg z = 0. \Rightarrow /'_R (y * z) z = y)$ $\wedge (\forall x y z$ $\bullet \neg z = 0. \Rightarrow /'_R (x * y) z = x * /'_R y z))$
	\$/\$

---

---

$\frac{\text{N}}{-1}$	$\vdash \forall m n \bullet m / n = \text{NR } m / \text{NR } n$
$\hat{\text{N}}$	$\vdash \forall x \bullet x^{-1} = 1. / x$
$\text{ZR}$	$\vdash (\forall x \bullet x \wedge 0 = 1.) \wedge (\forall x m \bullet x \wedge (m + 1) = x * x \wedge m)$
	$\vdash \text{ConstSpec}$
	$(\lambda \text{ZR}'$
	• $\text{ZR}' (\text{NZ } 0) = 0.$
	$\wedge \text{ZR}' (\text{NZ } 1) = 1.$
	$\wedge (\forall i j$
	• $\text{ZR}' (i + j) = \text{ZR}' i + \text{ZR}' j))$
	$\text{ZR}$
$\hat{\text{Z}}$	$\vdash \text{ConstSpec}$
	$(\lambda \hat{\text{Z}}$
	• $\forall x m$
	• $\hat{\text{Z}} x (\text{NZ } m) = x \wedge m$
	$\wedge \hat{\text{Z}} x (\sim (\text{NZ } (m + 1)))$
	$= (x \wedge (m + 1))^{-1}$
	$\hat{\text{Z}}$
<b>Float</b>	$\vdash \forall m p e \bullet \text{Float } m p e = \text{NR } m * 10. \wedge (e + \sim p)$
<b>Max<sub>R</sub></b>	$\vdash \text{ConstSpec}$
	$(\lambda \text{Max}'_R$
	• $(\forall x \bullet \text{Max}'_R [x] = x)$
	$\wedge (\forall x y L$
	• $\text{Max}'_R (\text{Cons } x (\text{Cons } y L))$
	$= (\text{if } x < \text{Max}'_R (\text{Cons } y L)$
	$\text{then } \text{Max}'_R (\text{Cons } y L)$
	$\text{else } x))$
	$\text{Max}_R$
<b>Min<sub>R</sub></b>	$\vdash \text{ConstSpec}$
	$(\lambda \text{Min}'_R$
	• $(\forall x \bullet \text{Min}'_R [x] = x)$
	$\wedge (\forall x y L$
	• $\text{Min}'_R (\text{Cons } x (\text{Cons } y L))$
	$= (\text{if } x > \text{Min}'_R (\text{Cons } y L)$
	$\text{then } \text{Min}'_R (\text{Cons } y L)$
	$\text{else } x))$
	$\text{Min}_R$

## 56.7 Theorems

### dy\_less\_order\_lemmas\_thm

$\vdash \text{StrictLinearOrder } (\text{Universe}, \text{\$dy\_less})$   
 $\wedge \text{UnboundedBelow } (\text{Universe}, \text{\$dy\_less})$   
 $\wedge \text{UnboundedAbove } (\text{Universe}, \text{\$dy\_less})$   
 $\wedge \text{Universe DenseIn } (\text{Universe}, \text{\$dy\_less})$

### is\_R\_rep\_consistent\_thm

$\vdash \exists a \bullet \text{Is\_R\_Rep } a$

### <<sub>R</sub>-consistent

$\vdash \text{Consistent}$

---

$(\lambda <'_R$   
 $\bullet$  *StrictLinearOrder* (*Universe*,  $<'_R$ )  
 $\wedge$  *UnboundedBelow* (*Universe*,  $<'_R$ )  
 $\wedge$  *UnboundedAbove* (*Universe*,  $<'_R$ )  
 $\wedge$  *Complete* (*Universe*,  $<'_R$ )  
 $\wedge$  ( $\exists \iota$   
 $\bullet$  ( $\forall a b \bullet <'_R (\iota a) (\iota b) \Leftrightarrow a \text{ dy\_less } b$ )  
 $\wedge \{x | \exists a \bullet \iota a = x\}$   
*DenseIn* (*Universe*,  $<'_R$ )))

**$\mathbb{R}$ \_unbounded\_below\_thm**  
 $\vdash \forall x \bullet \exists y \bullet y < x$

**$\mathbb{R}$ \_unbounded\_above\_thm**  
 $\vdash \forall x \bullet \exists y \bullet x < y$

**$\mathbb{R}$ \_less\_irrefl\_thm**  
 $\vdash \forall x \bullet \neg x < x$

**$\mathbb{R}$ \_less\_antisym\_thm**  
 $\vdash \forall x y \bullet \neg (x < y \wedge y < x)$

**$\mathbb{R}$ \_less\_trans\_thm**  
 $\vdash \forall x y z \bullet x < y \wedge y < z \Rightarrow x < z$

**$\mathbb{R}$ \_less\_cases\_thm**  
 $\vdash \forall x y \bullet x < y \vee x = y \vee y < x$

**$\mathbb{R}$ \_≤\_cases\_thm**  
 $\vdash \forall x y \bullet x \leq y \vee y \leq x$

**$\mathbb{R}$ \_≤\_less\_cases\_thm**  
 $\vdash \forall x y \bullet x \leq y \vee y < x$

**$\mathbb{R}$ \_eq\_≤\_thm**  $\vdash \forall x y \bullet x = y \Leftrightarrow x \leq y \wedge y \leq x$

**$\mathbb{R}$ \_≤\_antisym\_thm**  
 $\vdash \forall x y \bullet x \leq y \wedge y \leq x \Rightarrow x = y$

**$\mathbb{R}$ \_less\_≤\_trans\_thm**  
 $\vdash \forall x y z \bullet x < y \wedge y \leq z \Rightarrow x < z$

**$\mathbb{R}$ \_≤\_less\_trans\_thm**  
 $\vdash \forall x y z \bullet x \leq y \wedge y < z \Rightarrow x < z$

**$\mathbb{R}$ \_≤\_refl\_thm**  $\vdash \forall x \bullet x \leq x$

**$\mathbb{R}$ \_≤\_trans\_thm**  
 $\vdash \forall x y z \bullet x \leq y \wedge y \leq z \Rightarrow x \leq z$

**$\mathbb{R}$ \_≤\_¬\_less\_thm**  
 $\vdash \forall x y \bullet x \leq y \Leftrightarrow \neg y < x$

**$\mathbb{R}$ \_¬\_≤\_less\_thm**  
 $\vdash \forall x y \bullet \neg x \leq y \Leftrightarrow y < x$

**$\mathbb{R}$ \_less\_¬\_eq\_thm**  
 $\vdash \forall x y \bullet x < y \Rightarrow \neg x = y$

**$\mathbb{R}$ \_less\_dense\_thm**  
 $\vdash \forall x y \bullet x < y \Rightarrow (\exists z \bullet x < z \wedge z < y)$

**$\mathbb{R}$ \_complete\_thm**  
 $\vdash \forall A$   
 $\bullet \neg A = \{\} \wedge (\exists b \bullet \forall x \bullet x \in A \Rightarrow x \leq b)$   
 $\Rightarrow (\exists s$   
 $\bullet (\forall x \bullet x \in A \Rightarrow x \leq s)$

---

$$\wedge (\forall b \bullet (\forall x \bullet x \in A \Rightarrow x \leq b) \Rightarrow s \leq b))$$

**Sup\_consistent**

$$\begin{aligned} &\vdash \text{Consistent} \\ &(\lambda \text{Sup}' \\ &\bullet \forall A \\ &\bullet \neg A = \{\} \wedge (\exists b \bullet \forall x \bullet x \in A \Rightarrow x \leq b) \\ &\quad \Rightarrow (\forall x \bullet x \in A \Rightarrow x \leq \text{Sup}' A) \\ &\quad \wedge (\forall b \\ &\quad \bullet (\forall x \bullet x \in A \Rightarrow x \leq b) \Rightarrow \text{Sup}' A \leq b)) \end{aligned}$$

**R\_sup\_thm**

$$\begin{aligned} &\vdash \forall A a \\ &\bullet \neg A = \{\} \wedge (\forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \Rightarrow (\forall x \bullet x \in A \Rightarrow x \leq \text{Sup} A) \\ &\quad \wedge (\forall b \bullet (\forall x \bullet x \in A \Rightarrow x \leq b) \Rightarrow \text{Sup} A \leq b) \end{aligned}$$

**R\_less\_sup\_thm**

$$\begin{aligned} &\vdash \forall A \\ &\bullet \neg A = \{\} \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \Rightarrow (\forall x \bullet x < \text{Sup} A \Leftrightarrow (\exists y \bullet y \in A \wedge x < y)) \end{aligned}$$

**R\_less\_sup\_bc\_thm**

$$\begin{aligned} &\vdash \forall A x \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \wedge (\exists y \bullet y \in A \wedge x < y) \\ &\quad \Rightarrow x < \text{Sup} A \end{aligned}$$

**R\_≤\_sup\_thm**

$$\begin{aligned} &\vdash \forall A a \\ &\bullet \neg A = \{\} \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \Rightarrow (\forall x \\ &\quad \bullet x \leq \text{Sup} A \\ &\quad \Leftrightarrow (\forall y \bullet (\forall z \bullet z \in A \Rightarrow z \leq y) \Rightarrow x \leq y)) \end{aligned}$$

**R\_≤\_sup\_bc\_thm**

$$\begin{aligned} &\vdash \forall A a x \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \wedge (\forall y \bullet (\forall z \bullet z \in A \Rightarrow z \leq y) \Rightarrow x \leq y) \\ &\quad \Rightarrow x \leq \text{Sup} A \end{aligned}$$

**R\_∈\_≤\_sup\_bc\_thm**

$$\vdash \forall A x \bullet x \in A \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \Rightarrow x \leq \text{Sup} A$$

**R\_⊆\_sup\_thm**

$$\begin{aligned} &\vdash \forall A B \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \wedge \neg B = \{\} \\ &\quad \wedge (\exists b \bullet \forall y \bullet y \in B \Rightarrow y \leq b) \\ &\quad \wedge A \subseteq B \\ &\quad \Rightarrow \text{Sup} A \leq \text{Sup} B \end{aligned}$$

**R\_sup\_≤\_bc\_thm**

$$\begin{aligned} &\vdash \forall A a x \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \wedge (\forall y \bullet y \in A \Rightarrow y \leq x) \end{aligned}$$

$$\Rightarrow \text{Sup } A \leq x$$

 **$\mathbb{R}$ \_sup\_less\_bc\_thm**

$$\begin{aligned} &\vdash \forall A \ x \ z \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\exists a \bullet \forall x \bullet x \in A \Rightarrow x \leq a) \\ &\quad \wedge (\forall y \bullet y \in A \Rightarrow y \leq x) \\ &\quad \wedge x < z \\ &\Rightarrow \text{Sup } A < z \end{aligned}$$

 **$\mathbb{R}$ \_sup\_eq\_bc\_thm**

$$\begin{aligned} &\vdash \forall A \ a \ s \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\forall x \bullet x \in A \Rightarrow x \leq s) \\ &\quad \wedge (\forall x \bullet (\forall y \bullet y \in A \Rightarrow y \leq x) \Rightarrow s \leq x) \\ &\Rightarrow \text{Sup } A = s \end{aligned}$$

 **$\mathbb{R}$ \_eq\_sup\_bc\_thm**

$$\begin{aligned} &\vdash \forall A \ a \ s \\ &\bullet \neg A = \{\} \\ &\quad \wedge (\forall x \bullet x \in A \Rightarrow x \leq s) \\ &\quad \wedge (\forall x \bullet (\forall y \bullet y \in A \Rightarrow y \leq x) \Rightarrow s \leq x) \\ &\Rightarrow s = \text{Sup } A \end{aligned}$$

 **$\mathbb{R}$ \_less\_sup\_∈\_thm**

$$\begin{aligned} &\vdash \forall A \ a \\ &\bullet \neg A = \{\} \wedge (\forall x \bullet x \in A \Rightarrow x \leq a) \wedge \neg \text{Sup } A \in A \\ &\Rightarrow (\forall x \\ &\quad \bullet x < \text{Sup } A \Rightarrow (\exists y \bullet x < y \wedge y < \text{Sup } A \wedge y \in A)) \end{aligned}$$

**+ $\mathbb{R}$ \_consistent****0 $\mathbb{R}$ \_consistent****1 $\mathbb{R}$ \_consistent**

$$\begin{aligned} &\vdash \text{Consistent} \\ &\quad (\lambda (+'_R, 0'_R, 1'_R) \\ &\quad \bullet (\forall x \ y \ z \\ &\quad \quad \bullet +'_R (+'_R x \ y) \ z = +'_R x (+'_R y \ z)) \\ &\quad \quad \wedge (\forall x \ y \bullet +'_R x \ y = +'_R y \ x) \\ &\quad \quad \wedge (\forall x \bullet +'_R x \ 0'_R = x) \\ &\quad \quad \wedge (\forall x \bullet \exists y \bullet +'_R x \ y = 0'_R) \\ &\quad \quad \wedge (\forall x \ y \ z \bullet y < z \Rightarrow +'_R x \ y < +'_R x \ z) \\ &\quad \quad \wedge 0'_R < 1'_R) \end{aligned}$$

 **$\sim\mathbb{R}$ \_consistent**

$$\vdash \text{Consistent } (\lambda \sim'_R \bullet \forall x \bullet x + \sim'_R x = 0_R)$$

 **$\mathbb{R}$ \_plus\_assoc\_thm**

$$\vdash \forall x \ y \ z \bullet (x + y) + z = x + y + z$$

 **$\mathbb{R}$ \_plus\_comm\_thm**

$$\vdash \forall x \ y \bullet x + y = y + x$$

 **$\mathbb{R}$ \_plus\_unit\_thm**

$$\vdash \forall x \bullet x + 0_R = x$$

 **$\mathbb{R}$ \_plus\_mono\_thm**

$$\vdash \forall x \ y \ z \bullet y < z \Rightarrow x + y < x + z$$

 **$\mathbb{R}$ \_plus\_assoc\_thm1**

$$\vdash \forall x y z \bullet x + y + z = (x + y) + z$$

**$\mathbb{R}$ \_plus\_mono\_thm1**

$$\vdash \forall x y z \bullet y < z \Rightarrow y + x < z + x$$

**$\mathbb{R}$ \_plus\_mono\_thm2**

$$\vdash \forall x y s t \bullet x < y \wedge s < t \Rightarrow x + s < y + t$$

**$\mathbb{R}$ \_plus\_0\_thm**  $\vdash \forall x \bullet x + 0. = x \wedge 0. + x = x$

**$\mathbb{R}$ \_0\_1\_thm**  $\vdash 0_R = 0. \wedge 1_R = 1.$

**$\mathbb{R}$ \_plus\_order\_thm**

$$\begin{aligned} \vdash \forall x y z \\ \bullet y + x = x + y \\ \wedge (x + y) + z = x + y + z \\ \wedge y + x + z = x + y + z \end{aligned}$$

**$\mathbb{R}$ \_plus\_minus\_thm**

$$\vdash \forall x \bullet x + \sim x = 0. \wedge \sim x + x = 0.$$

**$\mathbb{R}$ \_eq\_thm**  $\vdash \forall x y \bullet x = y \Leftrightarrow x + \sim y = 0.$

**$\mathbb{NR}$ \_plus\_homomorphism\_thm**

$$\vdash \forall m n \bullet \mathbb{NR} (m + n) = \mathbb{NR} m + \mathbb{NR} n$$

**$\mathbb{R}$ \_minus\_clauses**

$$\begin{aligned} \vdash \forall x y \\ \bullet \sim (\sim x) = x \\ \wedge x + \sim x = 0. \\ \wedge \sim x + x = 0. \\ \wedge \sim (x + y) = \sim x + \sim y \\ \wedge \sim 0. = 0. \end{aligned}$$

**$\mathbb{R}$ \_minus\_eq\_thm**

$$\vdash \forall x y \bullet \sim x = \sim y \Leftrightarrow x = y$$

**$\mathbb{NR}$ \_0\_less\_thm**

$$\vdash \forall m \bullet 0. < \mathbb{NR} (m + 1)$$

**$\mathbb{NR}$ \_one\_one\_thm**

$$\vdash \forall m n \bullet \mathbb{NR} m = \mathbb{NR} n \Leftrightarrow m = n$$

**$\mathbb{R}$ \_plus\_clauses**

$$\begin{aligned} \vdash \forall x y z \\ \bullet (x + z = y + z \Leftrightarrow x = y) \\ \wedge (z + x = y + z \Leftrightarrow x = y) \\ \wedge (x + z = z + y \Leftrightarrow x = y) \\ \wedge (z + x = z + y \Leftrightarrow x = y) \\ \wedge (x + z = z \Leftrightarrow x = 0.) \\ \wedge (z + x = z \Leftrightarrow x = 0.) \\ \wedge (z = z + y \Leftrightarrow y = 0.) \\ \wedge (z = y + z \Leftrightarrow y = 0.) \\ \wedge x + 0. = x \\ \wedge 0. + x = x \\ \wedge \neg 1. = 0. \\ \wedge \neg 0. = 1. \end{aligned}$$

**$\mathbb{R}$ \_less\_less\_0\_thm**

$$\vdash \forall x y \bullet x < y \Leftrightarrow x + \sim y < 0.$$

**$\mathbb{R}$ \_less\_clauses**

$$\vdash \forall x y z$$

- $(x + z < y + z \Leftrightarrow x < y)$
- $\wedge (z + x < y + z \Leftrightarrow x < y)$
- $\wedge (x + z < z + y \Leftrightarrow x < y)$
- $\wedge (z + x < z + y \Leftrightarrow x < y)$
- $\wedge (x + z < z \Leftrightarrow x < 0.)$
- $\wedge (z + x < z \Leftrightarrow x < 0.)$
- $\wedge (x < z + x \Leftrightarrow 0. < z)$
- $\wedge (x < x + z \Leftrightarrow 0. < z)$
- $\wedge \neg x < x$
- $\wedge 0. < 1.$
- $\wedge \neg 1. < 0.$

 **$\mathbb{R}_{\text{less\_0\_less\_thm}}$** 

$$\vdash \forall x y \bullet x < y \Leftrightarrow 0. < y + \sim x$$

 **$\mathbb{R}_{\leq}$ -clauses**

$$\vdash \forall x y z$$

- $(x + z \leq y + z \Leftrightarrow x \leq y)$
- $\wedge (z + x \leq y + z \Leftrightarrow x \leq y)$
- $\wedge (x + z \leq z + y \Leftrightarrow x \leq y)$
- $\wedge (z + x \leq z + y \Leftrightarrow x \leq y)$
- $\wedge (x + z \leq z \Leftrightarrow x \leq 0.)$
- $\wedge (z + x \leq z \Leftrightarrow x \leq 0.)$
- $\wedge (x \leq z + x \Leftrightarrow 0. \leq z)$
- $\wedge (x \leq x + z \Leftrightarrow 0. \leq z)$
- $\wedge x \leq x$
- $\wedge 0. \leq 1.$
- $\wedge \neg 1. \leq 0.$

 **$\mathbb{R}_{\leq \leq 0}$ -thm**  $\vdash \forall x y \bullet x \leq y \Leftrightarrow x + \sim y \leq 0.$  **$\mathbb{R}_{\leq 0 \leq}$ -thm**  $\vdash \forall x y \bullet x \leq y \Leftrightarrow 0. \leq y + \sim x$  **$\mathbb{NR}_{\text{less\_thm}}$**   $\vdash \forall m n \bullet \mathbb{NR} m < \mathbb{NR} n \Leftrightarrow m < n$  **$\mathbb{R}_{\text{less\_strong\_dense\_thm}}$** 

$$\vdash \forall x y \bullet x < y \Rightarrow (\exists d \bullet 0. < d \wedge x + d < y)$$

 **$\mathbb{NR}_{\leq}$ -thm**  $\vdash \forall m n \bullet \mathbb{NR} m \leq \mathbb{NR} n \Leftrightarrow m \leq n$  **$\mathbb{R}_{\text{sup\_plus\_thm}}$** 

$$\vdash \forall A a x$$

- $\neg A = \{\}$   $\wedge (\forall x \bullet x \in A \Rightarrow x \leq a)$
- $\Rightarrow \text{Sup } A + x = \text{Sup } \{t \mid \exists a \bullet a \in A \wedge t < a + x\}$

 **$\mathbb{R}_{\text{sup\_plus\_sup\_thm}}$** 

$$\vdash \forall A a B b$$

- $\neg A = \{\}$
- $\wedge (\forall x \bullet x \in A \Rightarrow x \leq a)$
- $\wedge \neg B = \{\}$
- $\wedge (\forall y \bullet y \in B \Rightarrow y \leq b)$
- $\Rightarrow \text{Sup } A + \text{Sup } B$
- $= \text{Sup } \{t \mid \exists a b \bullet a \in A \wedge b \in B \wedge t < a + b\}$

 **$\mathbb{R}_{\text{delta\_induction\_thm}}$** 

$$\vdash \forall x p$$

- $(\exists d$
- $0. < d$
- $\wedge (\exists e$

$$\begin{aligned} & \bullet d < e \wedge (\forall t \bullet x < t \wedge t < x + e \Rightarrow p t) \\ & \wedge (\forall s \bullet x < s \wedge p s \Rightarrow p (s + d)) \\ \Rightarrow & (\forall y \bullet x < y \Rightarrow p y) \end{aligned}$$

 **$\mathbb{R}$ \_ord\_pres\_strict\_thm**

$$\begin{aligned} & \vdash \forall f \\ & \bullet (\forall x y \bullet x < y \Rightarrow f x < f y) \\ \Rightarrow & (\forall x y \bullet f x < f y \Rightarrow x < y) \end{aligned}$$

 **$\mathbb{R}$ \_add\_hom\_0\_thm**

$$\begin{aligned} & \vdash \forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \Rightarrow (\forall x \bullet f 0. = 0.) \end{aligned}$$

 **$\mathbb{R}$ \_add\_hom\_minus\_thm**

$$\begin{aligned} & \vdash \forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \\ \Rightarrow & (\forall x \bullet f (\sim x) = \sim (f x)) \end{aligned}$$

 **$\mathbb{R}$ \_add\_hom\_extension\_thm**

$$\begin{aligned} & \vdash \forall f \\ & \bullet (\forall x y \bullet 0. \leq x \wedge 0. \leq y \Rightarrow f (x + y) = f x + f y) \\ & \wedge (\forall x \bullet 0. \leq x \Rightarrow f (\sim x) = \sim (f x)) \\ \Rightarrow & (\forall x y \bullet f (x + y) = f x + f y) \end{aligned}$$

 **$\mathbb{R}$ \_monoid\_delta\_dense\_thm**

$$\begin{aligned} & \vdash \forall G d \\ & \bullet 0. \in G \\ & \wedge (\forall g h \bullet g \in G \wedge h \in G \Rightarrow g + h \in G) \\ & \wedge d \in G \\ & \wedge 0. < d \\ \Rightarrow & (\forall x \\ & \bullet 0. < x \Rightarrow (\exists g \bullet g \in G \wedge g \leq x \wedge x < g + d)) \end{aligned}$$

 **$\mathbb{R}$ \_monoid\_dense\_thm**

$$\begin{aligned} & \vdash \forall G \\ & \bullet 0. \in G \\ & \wedge (\forall g h \bullet g \in G \wedge h \in G \Rightarrow g + h \in G) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow (\exists g \bullet g \in G \wedge 0. < g \wedge g < x)) \\ \Rightarrow & (\forall x y \\ & \bullet 0. < x \wedge x < y \Rightarrow (\exists g \bullet g \in G \wedge x < g \wedge g < y)) \end{aligned}$$

 **$\mathbb{R}$ \_subgroup\_dense\_thm**

$$\begin{aligned} & \vdash \forall G \\ & \bullet 0. \in G \\ & \wedge (\forall g h \bullet g \in G \wedge h \in G \Rightarrow g + h \in G) \\ & \wedge (\forall g \bullet g \in G \Rightarrow \sim g \in G) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow (\exists g \bullet g \in G \wedge 0. < g \wedge g < x)) \\ \Rightarrow & (\forall x y \bullet x < y \Rightarrow (\exists g \bullet g \in G \wedge x < g \wedge g < y)) \end{aligned}$$

 **$\mathbb{R}$ \_semigroup\_dense\_thm**

$$\begin{aligned} & \vdash \forall G \\ & \bullet (\forall g h \bullet g \in G \wedge h \in G \Rightarrow g + h \in G) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow (\exists g \bullet g \in G \wedge 0. < g \wedge g < x)) \\ \Rightarrow & (\forall x y \\ & \bullet 0. < x \wedge x < y \Rightarrow (\exists g \bullet g \in G \wedge x < g \wedge g < y)) \end{aligned}$$

 **$\mathbb{R}$ \_add\_hom\_image\_group\_thm**

$$\begin{aligned}
& \vdash \forall f I \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge I = \{b \mid \exists a \bullet b = f a\} \\
& \quad \Rightarrow 0. \in I \\
& \quad \wedge (\forall g h \bullet g \in I \wedge h \in I \Rightarrow g + h \in I) \\
& \quad \wedge (\forall g \bullet g \in I \Rightarrow \sim g \in I)
\end{aligned}$$

 **$\mathbb{R}$ \_add\_hom\_kernel\_group\_thm**

$$\begin{aligned}
& \vdash \forall f K \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \wedge K = \{a \mid f a = 0.\} \\
& \quad \Rightarrow 0. \in K \\
& \quad \wedge (\forall g h \bullet g \in K \wedge h \in K \Rightarrow g + h \in K) \\
& \quad \wedge (\forall g \bullet g \in K \Rightarrow \sim g \in K)
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_thm**

$$\begin{aligned}
& \vdash \forall f \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \quad \Rightarrow (\forall x y \bullet x < y \Rightarrow f x < f y)
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_strict\_thm**

$$\begin{aligned}
& \vdash \forall f \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \quad \Rightarrow (\forall x y \bullet f x < f y \Rightarrow x < y)
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_one\_one\_thm**

$$\begin{aligned}
& \vdash \forall f \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \quad \Rightarrow \text{OneOne } f
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_dense\_image\_thm**

$$\begin{aligned}
& \vdash \forall f e \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \quad \wedge 0. < e \\
& \quad \Rightarrow (\exists d \bullet 0. < d \wedge f d < e)
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_onto\_thm**

$$\begin{aligned}
& \vdash \forall f \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \quad \Rightarrow \text{Onto } f
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_inverse\_add\_hom\_thm**

$$\begin{aligned}
& \vdash \forall f g \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \quad \wedge (\forall x \bullet g(f x) = x) \\
& \quad \wedge (\forall x \bullet f(g x) = x) \\
& \quad \Rightarrow (\forall x y \bullet g(x + y) = g x + g y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < g x)
\end{aligned}$$

 **$\mathbb{R}$ \_opah\_inverse\_thm**

$$\begin{aligned}
& \vdash \forall f \\
& \bullet (\forall x y \bullet f(x + y) = f x + f y)
\end{aligned}$$

$$\begin{aligned}
& \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\
& \Rightarrow (\exists g \\
& \bullet (\forall x \bullet g (f x) = x) \\
& \quad \wedge (\forall x \bullet f (g x) = x) \\
& \quad \wedge (\forall x y \bullet g (x + y) = g x + g y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < g x))
\end{aligned}$$

 **$\mathbb{R}$ \_copah\_id\_thm**

$$\begin{aligned}
& \vdash \exists \iota \\
& \bullet (\forall x \bullet \iota x = x) \\
& \quad \wedge (\forall x y \bullet \iota (x + y) = \iota x + \iota y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \iota x) \\
& \quad \wedge (\forall f \\
& \quad \bullet (\forall x y \bullet f (x + y) = f x + f y) \\
& \quad \Rightarrow (\forall x \bullet \iota (f x) = f (\iota x)))
\end{aligned}$$

 **$\mathbb{R}$ \_copah\_double\_thm**

$$\begin{aligned}
& \vdash \exists \alpha \\
& \bullet (\forall x \bullet \alpha x = x + x) \\
& \quad \wedge (\forall x y \bullet \alpha (x + y) = \alpha x + \alpha y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \alpha x) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow x < \alpha x) \\
& \quad \wedge (\forall f \\
& \quad \bullet (\forall x y \bullet f (x + y) = f x + f y) \\
& \quad \Rightarrow (\forall x \bullet \alpha (f x) = f (\alpha x)))
\end{aligned}$$

 **$\mathbb{R}$ \_copah\_halve\_thm**

$$\begin{aligned}
& \vdash \exists \beta \\
& \bullet (\forall x \bullet \beta x + \beta x = x) \\
& \quad \wedge (\forall x y \bullet \beta (x + y) = \beta x + \beta y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \beta x) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow \beta x < x) \\
& \quad \wedge (\forall f \\
& \quad \bullet (\forall x y \bullet f (x + y) = f x + f y) \\
& \quad \Rightarrow (\forall x \bullet \beta (f x) = f (\beta x)))
\end{aligned}$$

 **$\mathbb{R}$ \_copah\_comp\_thm**

$$\begin{aligned}
& \vdash \forall \alpha \beta \\
& \bullet (\forall x y \bullet \alpha (x + y) = \alpha x + \alpha y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \alpha x) \\
& \quad \wedge (\forall f \\
& \quad \bullet (\forall x y \bullet f (x + y) = f x + f y) \\
& \quad \Rightarrow (\forall x \bullet \alpha (f x) = f (\alpha x))) \\
& \quad \wedge (\forall x y \bullet \beta (x + y) = \beta x + \beta y) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \beta x) \\
& \quad \wedge (\forall f \\
& \quad \bullet (\forall x y \bullet f (x + y) = f x + f y) \\
& \quad \Rightarrow (\forall x \bullet \beta (f x) = f (\beta x))) \\
& \Rightarrow (\exists \gamma \\
& \bullet (\forall x \bullet \gamma x = \alpha (\beta x)) \\
& \quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \gamma x) \\
& \quad \wedge (\forall x y \bullet \gamma (x + y) = \gamma x + \gamma y))
\end{aligned}$$

$$\begin{aligned} & \wedge (\forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \\ & \Rightarrow (\forall x \bullet \gamma (f x) = f (\gamma x))) \end{aligned}$$

 **$\mathbb{R}$ \_copah\_sum\_thm**

$$\begin{aligned} & \vdash \forall \alpha \beta \\ & \bullet (\forall x y \bullet \alpha (x + y) = \alpha x + \alpha y) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \alpha x) \\ & \wedge (\forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \\ & \Rightarrow (\forall x \bullet \alpha (f x) = f (\alpha x))) \\ & \wedge (\forall x y \bullet \beta (x + y) = \beta x + \beta y) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \beta x) \\ & \wedge (\forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \\ & \Rightarrow (\forall x \bullet \beta (f x) = f (\beta x))) \\ & \Rightarrow (\exists \gamma \\ & \bullet (\forall x \bullet \gamma x = \alpha x + \beta x) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \gamma x) \\ & \wedge (\forall x y \bullet \gamma (x + y) = \gamma x + \gamma y) \\ & \wedge (\forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \\ & \Rightarrow (\forall x \bullet \gamma (f x) = f (\gamma x)))) \end{aligned}$$

 **$\mathbb{R}$ \_halve\_closed\_dense\_thm**

$$\begin{aligned} & \vdash \forall A e \\ & \bullet 0. < e \\ & \wedge e \in A \\ & \wedge (\forall y \bullet y \in A \Rightarrow (\exists z \bullet z \in A \wedge z + z = y)) \\ & \Rightarrow (\forall d \bullet 0. < d \Rightarrow (\exists a \bullet a \in A \wedge 0. < a \wedge a < d)) \end{aligned}$$

 **$\mathbb{R}$ \_copah\_dense\_thm**

$$\begin{aligned} & \vdash \forall d x y \\ & \bullet 0. < d \wedge 0. < x \wedge x < y \\ & \Rightarrow (\exists \gamma \\ & \bullet (\forall x y \bullet \gamma (x + y) = \gamma x + \gamma y) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \gamma x) \\ & \wedge (\forall f \\ & \bullet (\forall x y \bullet f (x + y) = f x + f y) \\ & \Rightarrow (\forall x \bullet \gamma (f x) = f (\gamma x))) \\ & \wedge x < \gamma d \\ & \wedge \gamma d < y) \end{aligned}$$

 **$\mathbb{R}$ \_opah\_extension\_thm1**

$$\begin{aligned} & \vdash \forall f \\ & \bullet (\forall x y \bullet 0. < x \wedge 0. < y \Rightarrow f (x + y) = f x + f y) \\ & \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\ & \Rightarrow (\exists \phi \\ & \bullet \phi 0. = 0. \\ & \wedge (\forall x \bullet 0. < x \Rightarrow \phi x = f x) \\ & \wedge (\forall x y \\ & \bullet 0. \leq x \wedge 0. \leq y \Rightarrow \phi (x + y) = \phi x + \phi y)) \end{aligned}$$

**$\mathbb{R}$ \_opah\_extension\_thm2**

$$\begin{aligned} &\vdash \forall f \\ &\bullet f \ 0. = 0. \\ &\quad \wedge (\forall x \ y \\ &\quad \bullet 0. \leq x \wedge 0. \leq y \Rightarrow f (x + y) = f x + f y) \\ &\Rightarrow (\exists \psi \\ &\bullet (\forall x \bullet 0. \leq x \Rightarrow \psi x = f x) \\ &\quad \wedge (\forall x \ y \bullet \psi (x + y) = \psi x + \psi y)) \end{aligned}$$

 **$\mathbb{R}$ \_opah\_extension\_thm**

$$\begin{aligned} &\vdash \forall f \\ &\bullet (\forall x \ y \bullet 0. < x \wedge 0. < y \Rightarrow f (x + y) = f x + f y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\ &\Rightarrow (\exists \phi \\ &\bullet (\forall x \bullet 0. < x \Rightarrow \phi x = f x) \\ &\quad \wedge (\forall x \ y \bullet \phi (x + y) = \phi x + \phi y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < \phi x)) \end{aligned}$$

 **$\mathbb{R}$ \_opah\_order\_thm**

$$\begin{aligned} &\vdash \forall f \ g \ d \\ &\bullet (\forall x \ y \bullet f (x + y) = f x + f y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\ &\quad \wedge (\forall x \ y \bullet g (x + y) = g x + g y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < g x) \\ &\quad \wedge 0. < d \\ &\quad \wedge f d < g d \\ &\Rightarrow (\forall x \bullet 0. < x \Rightarrow f x < g x) \end{aligned}$$

 **$\mathbb{R}$ \_opah\_eq\_thm**

$$\begin{aligned} &\vdash \forall f \ g \ d \\ &\bullet (\forall x \ y \bullet f (x + y) = f x + f y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\ &\quad \wedge (\forall x \ y \bullet g (x + y) = g x + g y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < g x) \\ &\quad \wedge 0. < d \\ &\quad \wedge f d = g d \\ &\Rightarrow f = g \end{aligned}$$

 **$\mathbb{R}$ \_opah\_complete\_thm**

$$\begin{aligned} &\vdash \forall d \ e \\ &\bullet 0. < d \wedge 0. < e \\ &\Rightarrow (\exists f \\ &\bullet (\forall x \ y \bullet f (x + y) = f x + f y) \\ &\quad \wedge (\forall x \bullet 0. < x \Rightarrow 0. < f x) \\ &\quad \wedge f d = e) \end{aligned}$$

**\* $\mathbb{R}$ \_consistent**

$$\begin{aligned} &\vdash \text{Consistent} \\ &(\lambda \ *'_R \\ &\bullet (\forall x \ y \ z \\ &\quad \bullet *'_R (*'_R x y) z = *'_R x (*'_R y z)) \\ &\quad \wedge (\forall x \bullet *'_R x 1. = x) \\ &\quad \wedge (\forall x \ y \ z \end{aligned}$$

$$\begin{aligned} & \bullet \text{ }'_R x (y + z) = \text{ }'_R x y + \text{ }'_R x z \\ & \wedge (\forall x y \bullet \text{ }'_R x y = \text{ }'_R y x) \\ & \wedge (\forall x y \bullet 0. < x \wedge 0. < y \Rightarrow 0. < \text{ }'_R x y) \end{aligned}$$

 **$\mathbb{R}$ \_times\_assoc\_thm**

$$\vdash \forall x y z \bullet (x * y) * z = x * y * z$$

 **$\mathbb{R}$ \_times\_comm\_thm**

$$\vdash \forall x y \bullet x * y = y * x$$

 **$\mathbb{R}$ \_times\_unit\_thm**

$$\vdash \forall x \bullet x * 1. = x$$

 **$\mathbb{R}$ \_0\_less\_0\_less\_times\_thm**

$$\vdash \forall x y \bullet 0. < x \wedge 0. < y \Rightarrow 0. < x * y$$

 **$\mathbb{R}$ \_times\_assoc\_thm1**

$$\vdash \forall x y z \bullet x * y * z = (x * y) * z$$

 **$\mathbb{R}$ \_times\_plus\_distrib\_thm**

$$\begin{aligned} & \vdash \forall x y z \\ & \bullet x * (y + z) = x * y + x * z \\ & \wedge (x + y) * z = x * z + y * z \end{aligned}$$

 **$\mathbb{R}$ \_times\_order\_thm**

$$\begin{aligned} & \vdash \forall x y z \\ & \bullet y * x = x * y \\ & \wedge (x * y) * z = x * y * z \\ & \wedge y * x * z = x * y * z \end{aligned}$$

 **$\mathbb{R}$ \_times\_0\_thm**

$$\vdash \forall x \bullet x * 0. = 0. \wedge 0. * x = 0.$$

 **$\mathbb{R}$ \_times\_1\_thm**

$$\vdash \forall x \bullet x * 1. = x \wedge 1. * x = x$$

 **$\mathbb{NR}$ \_times\_homomorphism\_thm**

$$\vdash \forall m n \bullet \mathbb{NR} (m * n) = \mathbb{NR} m * \mathbb{NR} n$$

 **$\mathbb{R}$ \_times\_minus\_thm**

$$\begin{aligned} & \vdash \forall x y \\ & \bullet \sim x * y = \sim (x * y) \\ & \wedge x * \sim y = \sim (x * y) \\ & \wedge \sim x * \sim y = x * y \end{aligned}$$

 **$/\mathbb{R}$ -consistent**

$$\begin{aligned} & \vdash \textit{Consistent} \\ & (\lambda \text{ }'_R \\ & \bullet (\forall y z \bullet \neg z = 0. \Rightarrow \text{ }'_R (y * z) z = y) \\ & \wedge (\forall x y z \\ & \bullet \neg z = 0. \Rightarrow \text{ }'_R (x * y) z = x * \text{ }'_R y z)) \end{aligned}$$

 **$\mathbb{R}$ \_over\_times\_recip\_thm**

$$\vdash \forall z \bullet \neg z = 0. \Rightarrow (\forall x \bullet x / z = x * z^{-1})$$

 **$\mathbb{R}$ \_times\_recip\_thm**

$$\vdash \forall z \bullet \neg z = 0. \Rightarrow z * z^{-1} = 1.$$

 **$\mathbb{R}$ \_eq\_recip\_thm**

$$\vdash \forall z \bullet \neg z = 0. \Rightarrow (\forall y \bullet y = z \Leftrightarrow y * z^{-1} = 1.)$$

 **$\mathbb{R}$ \_times\_cancel\_thm**

$$\vdash \forall x y z \bullet \neg z = 0. \Rightarrow (x * z = y * z \Leftrightarrow x = y)$$

 **$\mathbb{R}$ \_times\_eq\_0\_thm**

$$\vdash \forall x y \bullet x * y = 0. \Leftrightarrow x = 0. \vee y = 0.$$

 **$\mathbb{R}$ \_times\_clauses**

$$\vdash \forall x$$

$$\bullet 0. * x = 0. \wedge x * 0. = 0. \wedge x * 1. = x \wedge 1. * x = x$$

 **$\mathbb{R}$ \_times\_mono\_⇔\_thm**

$$\vdash \forall x \bullet 0. < x \Rightarrow (\forall y z \bullet y < z \Leftrightarrow x * y < x * z)$$

 **$\mathbb{R}$ \_times\_mono\_thm**

$$\vdash \forall x y z \bullet 0. < x \wedge y < z \Rightarrow x * y < x * z$$

 **$\mathbb{R}$ \_0\_≤\_0\_≤\_times\_thm**

$$\vdash \forall x y \bullet 0. \leq x \wedge 0. \leq y \Rightarrow 0. \leq x * y$$

 **$\mathbb{R}$ \_¬\_recip\_0\_thm**

$$\vdash \forall z \bullet \neg z = 0. \Rightarrow \neg z^{-1} = 0.$$

 **$\mathbb{R}$ \_recip\_clauses**

$$\vdash (1.^{-1} = 1.$$

$$\wedge (\forall w$$

$$\bullet \neg w = 0.$$

$$\Rightarrow w^{-1}^{-1} = w$$

$$\wedge w * w^{-1} = 1.$$

$$\wedge w^{-1} * w = 1.))$$

$$\wedge (\forall w z$$

$$\bullet \neg w = 0. \wedge \neg z = 0.$$

$$\Rightarrow (w * z)^{-1} = w^{-1} * z^{-1})$$

 **$\mathbb{R}$ \_cross\_mult\_eq\_thm**

$$\vdash \forall w z$$

$$\bullet \neg w = 0. \wedge \neg z = 0.$$

$$\Rightarrow (\forall x y \bullet x / w = y / z \Leftrightarrow x * z = w * y)$$

 **$\mathbb{R}$ \_less\_¬\_eq\_0\_thm**

$$\vdash \forall z \bullet 0. < z \Rightarrow \neg z = 0.$$

 **$\mathbb{R}$ \_0\_less\_0\_less\_recip\_thm**

$$\vdash \forall z \bullet 0. < z \Rightarrow 0. < z^{-1}$$

 **$\mathbb{R}$ \_cross\_mult\_less\_thm**

$$\vdash \forall w z$$

$$\bullet 0. < w \wedge 0. < z$$

$$\Rightarrow (\forall x y \bullet x / w < y / z \Leftrightarrow x * z < w * y)$$

 **$\mathbb{R}$ \_over\_cancel\_eq\_thm**

$$\vdash \forall w z$$

$$\bullet \neg w = 0. \wedge \neg z = 0.$$

$$\Rightarrow (\forall x \bullet (x * z) / (w * z) = x / w)$$

 **$\mathbb{R}$ \_over\_plus\_over\_thm**

$$\vdash \forall x y u v$$

$$\bullet \neg u = 0. \wedge \neg v = 0.$$

$$\Rightarrow x / u + y / v = (x * v + y * u) / (u * v)$$

 **$\mathbb{R}$ \_0\_over\_thm**  $\vdash \forall z \bullet \neg z = 0. \Rightarrow 0. / z = 0.$  **$\mathbb{R}$ \_over\_1\_thm**  $\vdash \forall x \bullet x / 1. = x$  **$\mathbb{NR}$ \_plus\_homomorphism\_thm1**

$$\vdash \forall m n \bullet \mathbb{NR} m + \mathbb{NR} n = \mathbb{NR} (m + n)$$

 **$\mathbb{NR}$ \_times\_homomorphism\_thm1**

$$\vdash \forall m n \bullet \mathbb{NR} m * \mathbb{NR} n = \mathbb{NR} (m * n)$$

**$\mathbb{R}$ \_frac\_cross\_mult\_eq\_thm**

$$\begin{aligned} &\vdash \forall m n i j \\ &\bullet i / (m + 1) = j / (n + 1) \\ &\Leftrightarrow i * (n + 1) = j * (m + 1) \end{aligned}$$

 **$\mathbb{R}$ \_frac\_cancel\_eq\_thm**

$$\begin{aligned} &\vdash \forall i m n \\ &\bullet (i * (n + 1)) / ((m + 1) * (n + 1)) = i / (m + 1) \end{aligned}$$

 **$\mathbb{R}$ \_frac\_0\_thm**  $\vdash \forall m \bullet 0 / (m + 1) = 0.$  **$\mathbb{R}$ \_frac\_N\_thm**  $\vdash \forall i \bullet i / 1 = \mathbb{N}R i$  **$\mathbb{R}$ \_frac\_plus\_frac\_thm**

$$\begin{aligned} &\vdash \forall i j k m n \\ &\bullet i / (m + 1) + j / (n + 1) \\ &\quad = (i * (n + 1) + j * (m + 1)) \\ &\quad \quad / ((m + 1) * (n + 1)) \end{aligned}$$

 **$\mathbb{R}$ \_frac\_minus\_frac\_thm**

$$\begin{aligned} &\vdash \forall i j k m n \\ &\bullet j * (m + 1) \leq i * (n + 1) \\ &\quad \Rightarrow i / (m + 1) + \sim (j / (n + 1)) \\ &\quad = (i * (n + 1) - j * (m + 1)) \\ &\quad \quad / ((m + 1) * (n + 1)) \end{aligned}$$

 **$\mathbb{R}$ \_frac\_minus\_frac\_thm1**

$$\begin{aligned} &\vdash \forall i j m n \\ &\bullet i * (n + 1) \leq j * (m + 1) \\ &\quad \Rightarrow i / (m + 1) + \sim (j / (n + 1)) \\ &\quad = \sim \\ &\quad \quad ((j * (m + 1) - i * (n + 1)) \\ &\quad \quad \quad / ((m + 1) * (n + 1))) \end{aligned}$$

 **$\mathbb{R}$ \_over\_times\_over\_thm**

$$\begin{aligned} &\vdash \forall x y u v \\ &\bullet \neg u = 0. \wedge \neg v = 0. \\ &\quad \Rightarrow x / u * y / v = (x * y) / (u * v) \end{aligned}$$

 **$\mathbb{R}$ \_frac\_times\_frac\_thm**

$$\begin{aligned} &\vdash \forall i j m n \\ &\bullet i / (m + 1) * j / (n + 1) \\ &\quad = (i * j) / ((m + 1) * (n + 1)) \end{aligned}$$

 **$\mathbb{R}$ \_over\_recip\_thm**

$$\vdash \forall u v \bullet \neg u = 0. \wedge \neg v = 0. \Rightarrow (u / v)^{-1} = v / u$$

 **$\mathbb{R}$ \_frac\_recip\_thm**

$$\vdash \forall m n \bullet ((m + 1) / (n + 1))^{-1} = (n + 1) / (m + 1)$$

 **$\mathbb{R}$ \_minus\_recip\_thm**

$$\vdash \forall z \bullet \neg z = 0. \Rightarrow \sim z^{-1} = \sim (z^{-1})$$

 **$\mathbb{R}$ \_over\_eq\_0\_thm**

$$\vdash \forall u v \bullet \neg u = 0. \wedge \neg v = 0. \Rightarrow \neg u / v = 0.$$

 **$\mathbb{R}$ \_over\_over\_over\_thm**

$$\begin{aligned} &\vdash \forall x y u v \\ &\bullet \neg u = 0. \wedge \neg v = 0. \wedge \neg y = 0. \\ &\quad \Rightarrow x / u / (y / v) = (x * v) / (u * y) \end{aligned}$$

 **$\mathbb{R}$ \_frac\_less\_frac\_thm**

$$\begin{aligned} &\vdash \forall i j m n \\ &\bullet i / (m + 1) < j / (n + 1) \\ &\Leftrightarrow i * (n + 1) < j * (m + 1) \end{aligned}$$

 **$\mathbb{R}$ \_minus\_frac\_less\_frac\_thm**

$$\vdash \forall i j m n \bullet \sim (i / (m + 1)) < j / (n + 1) \Leftrightarrow 0 < i + j$$

 **$\mathbb{R}$ \_frac\_less\_minus\_frac\_thm**

$$\vdash \forall i j m n \bullet \neg i / (m + 1) < \sim (j / (n + 1))$$

 **$\mathbb{R}$ \_0\_≤\_frac\_thm**

$$\vdash \forall i m \bullet 0. \leq i / (m + 1)$$

 **$\mathbb{R}$ \_abs\_frac\_thm**

$$\vdash \forall i m \bullet \text{Abs } (i / (m + 1)) = i / (m + 1)$$

 **$\mathbb{R}$ \_abs\_minus\_thm**

$$\vdash \forall x \bullet \text{Abs } (\sim x) = \text{Abs } x$$

**Max<sub>R</sub>-consistent**

$$\begin{aligned} &\vdash \text{Consistent} \\ &(\lambda \text{Max}'_R \\ &\bullet (\forall x \bullet \text{Max}'_R [x] = x) \\ &\quad \wedge (\forall x y L \\ &\quad \bullet \text{Max}'_R (\text{Cons } x (\text{Cons } y L)) \\ &\quad = (\text{if } x < \text{Max}'_R (\text{Cons } y L) \\ &\quad \text{then } \text{Max}'_R (\text{Cons } y L) \\ &\quad \text{else } x))) \end{aligned}$$

**Min<sub>R</sub>-consistent**

$$\begin{aligned} &\vdash \text{Consistent} \\ &(\lambda \text{Min}'_R \\ &\bullet (\forall x \bullet \text{Min}'_R [x] = x) \\ &\quad \wedge (\forall x y L \\ &\quad \bullet \text{Min}'_R (\text{Cons } x (\text{Cons } y L)) \\ &\quad = (\text{if } x > \text{Min}'_R (\text{Cons } y L) \\ &\quad \text{then } \text{Min}'_R (\text{Cons } y L) \\ &\quad \text{else } x))) \end{aligned}$$

 **$\mathbb{R}$ \_max\_cons\_thm**

$$\begin{aligned} &\vdash \forall x L \\ &\bullet \text{Max}_R (\text{Cons } x L) \\ &= (\text{if } L = [] \\ &\quad \text{then } x \\ &\quad \text{else if } x < \text{Max}_R L \\ &\quad \text{then } \text{Max}_R L \\ &\quad \text{else } x) \end{aligned}$$

 **$\mathbb{R}$ \_max\_conv\_thm**

$$\begin{aligned} &\vdash \forall x y L \\ &\bullet \text{Max}_R (\text{Cons } x (\text{Cons } y L)) \\ &= (\text{if } x < y \\ &\quad \text{then } \text{Max}_R (\text{Cons } y L) \\ &\quad \text{else } \text{Max}_R (\text{Cons } x L)) \end{aligned}$$

 **$\mathbb{R}$ \_min\_cons\_thm**

$$\begin{aligned} &\vdash \forall x L \\ &\bullet \text{Min}_R (\text{Cons } x L) \end{aligned}$$

$$= (\text{if } L = [] \\ \text{then } x \\ \text{else if } \text{Min}_R L < x \\ \text{then } \text{Min}_R L \\ \text{else } x)$$
 **$\mathbb{R}$ \_min\_conv\_thm**

$$\vdash \forall x y L \\ \bullet \text{Min}_R (\text{Cons } x (\text{Cons } y L)) \\ = (\text{if } x < y \\ \text{then } \text{Min}_R (\text{Cons } x L) \\ \text{else } \text{Min}_R (\text{Cons } y L))$$
 **$\hat{\mathbb{Z}}$ \_consistent**

$$\vdash \text{Consistent} \\ (\lambda \hat{\mathbb{Z}} \\ \bullet \forall x m \\ \bullet \hat{\mathbb{Z}} x (\text{NZ } m) = x \hat{\mathbb{Z}} m \\ \wedge \hat{\mathbb{Z}} x (\sim (\text{NZ } (m + 1))) \\ = (x \hat{\mathbb{Z}} (m + 1))^{-1})$$
 **$\mathbb{Z}\mathbb{R}$ \_consistent**

$$\vdash \text{Consistent} \\ (\lambda \$\mathbb{Z}\mathbb{R}' \\ \bullet \$\mathbb{Z}\mathbb{R}' (\text{NZ } 0) = 0. \\ \wedge \$\mathbb{Z}\mathbb{R}' (\text{NZ } 1) = 1. \\ \wedge (\forall i j \\ \bullet \$\mathbb{Z}\mathbb{R}' (i + j) = \$\mathbb{Z}\mathbb{R}' i + \$\mathbb{Z}\mathbb{R}' j))$$
 **$\mathbb{Z}\mathbb{R}$ \_plus\_homomorphism\_thm**

$$\vdash \forall i j \bullet \mathbb{Z}\mathbb{R} (i + j) = \mathbb{Z}\mathbb{R} i + \mathbb{Z}\mathbb{R} j$$

$$\mathbb{Z}\mathbb{R}\text{-minus\_thm} \vdash \forall i \bullet \mathbb{Z}\mathbb{R} (\sim i) = \sim (\mathbb{Z}\mathbb{R} i)$$

$$\mathbb{Z}\mathbb{R}\text{-NZ\_thm} \vdash \forall m \bullet \mathbb{Z}\mathbb{R} (\text{NZ } m) = \text{NR } m \wedge \mathbb{Z}\mathbb{R} (\sim (\text{NZ } m)) = \sim (\text{NR } m)$$
 **$\mathbb{Z}\mathbb{R}$ \_times\_homomorphism\_thm**

$$\vdash \forall i j \bullet \mathbb{Z}\mathbb{R} (i * j) = \mathbb{Z}\mathbb{R} i * \mathbb{Z}\mathbb{R} j$$

## 57 THE THEORY $\mathbb{N}$

### 57.1 Parents

*pair*

### 57.2 Children

*list*

### 57.3 Constants

<b>Is_N_Rep</b>	$IND \rightarrow Bool$
<b>Zero</b>	$Worth$
<b>Suc</b>	$Worth \rightarrow Worth$
<b>\$+</b>	$Worth \rightarrow Worth \rightarrow Worth$
<b>\$≤</b>	$Worth \rightarrow Worth \rightarrow Bool$
<b>\$≥</b>	$Worth \rightarrow Worth \rightarrow Bool$
<b>\$&lt;</b>	$Worth \rightarrow Worth \rightarrow Bool$
<b>\$&gt;</b>	$Worth \rightarrow Worth \rightarrow Bool$
<b>\$*</b>	$Worth \rightarrow Worth \rightarrow Worth$
<b>\$Mod</b>	$Worth \rightarrow Worth \rightarrow Worth$
<b>\$Div</b>	$Worth \rightarrow Worth \rightarrow Worth$
<b>\$-</b>	$Worth \rightarrow Worth \rightarrow Worth$

### 57.4 Types

**Worth**

### 57.5 Fixity

*Left Infix 305:*

—

*Left Infix 315:*

**Div Mod**

*Right Infix 210:*

< > ≤ ≥

*Right Infix 300:*

+

*Right Infix 310:*

\*

### 57.6 Definitions

**Is\_N\_Rep**

**is\_n\_rep\_def**  $\vdash \exists \text{ zero suc}$

- $(Is\_N\_Rep \text{ zero}$ 
  - $\wedge (\forall n \bullet Is\_N\_Rep \ n \Rightarrow Is\_N\_Rep \ (suc \ n)))$
  - $\wedge (\forall n \bullet Is\_N\_Rep \ n \Rightarrow \neg \text{ suc } \ n = \text{ zero})$
  - $\wedge \text{ OneOne } \text{ suc}$
  - $\wedge (\forall p$ 
    - $p \ \text{zero} \wedge (\forall m \bullet p \ m \Rightarrow p \ (suc \ m))$
    - $\Rightarrow (\forall n \bullet Is\_N\_Rep \ n \Rightarrow p \ n))$

**N**

**N\_def**  $\vdash \exists f \bullet \text{TypeDefn } Is\_N\_Rep \ f$

**Zero**

**Suc**

---

**zero\_suc\_def**  $\vdash (\forall n \bullet \neg \text{Suc } n = \text{Zero})$   
 $\wedge \text{OneOne } \text{Suc}$   
 $\wedge (\forall p$   
 $\bullet p \text{ Zero} \wedge (\forall m \bullet p \ m \Rightarrow p \ (\text{Suc } m)) \Rightarrow (\forall n \bullet p \ n))$

+

**plus\_def**  $\vdash \forall m \ n$   
 $\bullet 0 + n = n$   
 $\wedge (m + 1) + n = (m + n) + 1$   
 $\wedge \text{Suc } m = m + 1$

$\leq$

**$\leq$ \_def**  $\vdash \forall m \ n \bullet m \leq n \Leftrightarrow (\exists i \bullet m + i = n)$

$\geq$

**$\geq$ \_def**  $\vdash \forall m \ n \bullet m \geq n \Leftrightarrow n \leq m$

<

**less\_def**  $\vdash \forall m \ n \bullet m < n \Leftrightarrow m + 1 \leq n$

>

**greater\_def**  $\vdash \forall m \ n \bullet m > n \Leftrightarrow n < m$

\*

**times\_def**  $\vdash \forall m \ n \bullet 0 * n = 0 \wedge (m + 1) * n = m * n + n$

**Mod**

**mod\_def**  $\vdash \forall m \ n$   
 $\bullet 0 < n$   
 $\Rightarrow 0 \text{ Mod } n = 0$   
 $\wedge (m + 1) \text{ Mod } n$   
 $= (\text{if } m \text{ Mod } n + 1 < n$   
 $\text{then } m \text{ Mod } n + 1$   
 $\text{else } 0)$

**Div**

**div\_def**  $\vdash \forall m \ n$   
 $\bullet 0 < n$   
 $\Rightarrow 0 \text{ Div } n = 0$   
 $\wedge (m + 1) \text{ Div } n$   
 $= (\text{if } m \text{ Mod } n + 1 < n$   
 $\text{then } m \text{ Div } n$   
 $\text{else } m \text{ Div } n + 1)$

-

**minus\_def**  $\vdash \forall m \ n \bullet (m + n) - n = m$

## 57.7 Theorems

### induction\_thm

 $\vdash \forall p \bullet p \ 0 \wedge (\forall m \bullet p \ m \Rightarrow p \ (m + 1)) \Rightarrow (\forall n \bullet p \ n)$ 

### $\neg$ \_plus1\_thm

 $\vdash \forall n \bullet \neg n + 1 = 0$ 

### one\_one\_plus1\_thm

 $\vdash \forall x1 \ x2 \bullet x1 + 1 = x2 + 1 \Rightarrow x1 = x2$ 

### prim\_rec\_thm

 $\vdash \forall z \ s \bullet \exists_1 f \bullet f \ 0 = z \wedge (\forall n \bullet f \ (n + 1) = s \ (f \ n) \ n)$ 

### plus\_assoc\_thm

 $\vdash \forall i \ m \ n \bullet (i + m) + n = i + m + n$

**plus\_assoc\_thm1**

$$\vdash \forall i m n \bullet i + m + n = (i + m) + n$$

**plus\_comm\_thm**

$$\vdash \forall m n \bullet m + n = n + m$$

**plus\_order\_thm**

$$\begin{aligned} &\vdash \forall i m n \\ &\bullet m + i = i + m \\ &\quad \wedge (i + m) + n = i + m + n \\ &\quad \wedge m + i + n = i + m + n \end{aligned}$$

**plus\_clauses**

$$\begin{aligned} &\vdash \forall m n i \\ &\bullet (m + i = n + i \Leftrightarrow m = n) \\ &\quad \wedge (i + m = n + i \Leftrightarrow m = n) \\ &\quad \wedge (m + i = i + n \Leftrightarrow m = n) \\ &\quad \wedge (i + m = i + n \Leftrightarrow m = n) \\ &\quad \wedge (m + i = i \Leftrightarrow m = 0) \\ &\quad \wedge (i + m = i \Leftrightarrow m = 0) \\ &\quad \wedge (i = i + n \Leftrightarrow n = 0) \\ &\quad \wedge (i = n + i \Leftrightarrow n = 0) \\ &\quad \wedge (m + i = 0 \Leftrightarrow m = 0 \wedge i = 0) \\ &\quad \wedge (0 = m + i \Leftrightarrow m = 0 \wedge i = 0) \\ &\quad \wedge (m + 0 = m \wedge 0 + m = m) \\ &\quad \wedge \neg 1 = 0 \\ &\quad \wedge \neg 0 = 1 \end{aligned}$$

**≤\_trans\_thm**

$$\vdash \forall m i n \bullet m \leq i \wedge i \leq n \Rightarrow m \leq n$$

**less\_trans\_thm**

$$\vdash \forall m i n \bullet m < i \wedge i < n \Rightarrow m < n$$

**≤\_clauses**

$$\begin{aligned} &\vdash \forall m n i \\ &\bullet (m + i \leq n + i \Leftrightarrow m \leq n) \\ &\quad \wedge (i + m \leq n + i \Leftrightarrow m \leq n) \\ &\quad \wedge (m + i \leq i + n \Leftrightarrow m \leq n) \\ &\quad \wedge (i + m \leq i + n \Leftrightarrow m \leq n) \\ &\quad \wedge (m + i \leq i \Leftrightarrow m = 0) \\ &\quad \wedge (i + m \leq i \Leftrightarrow m = 0) \\ &\quad \wedge (m + i \leq 0 \Leftrightarrow m = 0 \wedge i = 0) \\ &\quad \wedge (m \leq 0 \Leftrightarrow m = 0) \\ &\quad \wedge m \leq m + i \\ &\quad \wedge m \leq i + m \\ &\quad \wedge m \leq m \\ &\quad \wedge 0 \leq m \\ &\quad \wedge \neg 1 \leq 0 \end{aligned}$$

**less\_clauses**

$$\begin{aligned} &\vdash \forall m n i \\ &\bullet (m + i < n + i \Leftrightarrow m < n) \\ &\quad \wedge (i + m < n + i \Leftrightarrow m < n) \\ &\quad \wedge (m + i < i + n \Leftrightarrow m < n) \\ &\quad \wedge (i + m < i + n \Leftrightarrow m < n) \\ &\quad \wedge (m < m + i \Leftrightarrow 0 < i) \\ &\quad \wedge (m < i + m \Leftrightarrow 0 < i) \\ &\quad \wedge \neg m + i < m \end{aligned}$$

---

	$\wedge \neg m + i < i$
	$\wedge \neg m < 0$
	$\wedge \neg m < m$
	$\wedge 0 < m + 1$
	$\wedge 0 < 1 + m$
	$\wedge 0 < 1$
<b><math>\mathbb{N}</math>_cases_thm</b>	$\vdash \forall m \bullet m = 0 \vee (\exists i \bullet m = i + 1)$
<b><math>\leq</math>_cases_thm</b>	$\vdash \forall m n \bullet m \leq n \vee n \leq m$
<b><math>\leq</math>_plus1_thm</b>	$\vdash \forall m n \bullet m \leq n + 1 \Leftrightarrow m = n + 1 \vee m \leq n$
<b>plus1_<math>\leq</math>_thm</b>	$\vdash \forall m n \bullet m + 1 \leq n \Leftrightarrow m \leq n \wedge \neg m = n$
<b><math>\neg</math>_plus1_<math>\leq</math>_thm</b>	$\vdash \forall m n \bullet \neg m + 1 \leq n \Leftrightarrow n \leq m$
<b>less_cases_thm</b>	$\vdash \forall m n \bullet m < n \vee m = n \vee n < m$
<b><math>\neg</math>_less_plus1_thm</b>	$\vdash \forall m n \bullet \neg m < n + 1 \Leftrightarrow n < m$
<b>less_plus1_thm</b>	$\vdash \forall m n \bullet m < n + 1 \Leftrightarrow m = n \vee m < n$
<b>plus1_less_thm</b>	$\vdash \forall m n \bullet m + 1 < n \Leftrightarrow m < n \wedge \neg m + 1 = n$
<b><math>\leq</math>_antisym_thm</b>	$\vdash \forall m n \bullet m \leq n \wedge n \leq m \Leftrightarrow m = n$
<b>less_irrefl_thm</b>	$\vdash \forall m n \bullet \neg (m < n \wedge n < m)$
<b>cov_induction_thm</b>	$\vdash \forall p \bullet (\forall n \bullet (\forall m \bullet m < n \Rightarrow p m) \Rightarrow p n) \Rightarrow (\forall n \bullet p n)$
<b>less_well_order_thm</b>	$\vdash \forall p \bullet (\exists i \bullet p i) \Leftrightarrow (\exists m \bullet p m \wedge (\forall i \bullet p i \Rightarrow \neg i < m))$
<b><math>\neg</math>_less_thm</b>	$\vdash \forall m n \bullet \neg m < n \Leftrightarrow n \leq m$
<b><math>\neg</math>_≤_thm</b>	$\vdash \forall m n \bullet \neg m \leq n \Leftrightarrow n < m$
<b><math>\leq</math>_well_order_thm</b>	$\vdash \forall p \bullet (\exists i \bullet p i) \Leftrightarrow (\exists m \bullet p m \wedge (\forall i \bullet p i \Rightarrow m \leq i))$
<b><math>\leq</math>_least_upper_bound_thm</b>	$\vdash \forall p$ $\bullet (\exists i \bullet p i) \wedge (\exists n \bullet \forall j \bullet p j \Rightarrow j \leq n)$ $\Leftrightarrow (\exists m \bullet p m \wedge (\forall j \bullet p j \Rightarrow j \leq m))$
<b>minimum_<math>\neg</math>_thm</b>	$\vdash \forall p b$ $\bullet p 0 \wedge \neg p b$ $\Rightarrow (\exists m \bullet (\forall n \bullet n \leq m \Rightarrow p n) \wedge \neg p (m + 1))$
<b>times_comm_thm</b>	$\vdash \forall m n \bullet m * n = n * m$
<b>times_assoc_thm</b>	$\vdash \forall i m n \bullet (i * m) * n = i * m * n$
<b>times_plus_distrib_thm</b>	$\vdash \forall i m n$ $\bullet (i + m) * n = i * n + m * n$ $\wedge i * (m + n) = i * m + i * n$

---

**times\_clauses**

$$\vdash \forall m \bullet m * 0 = 0 \wedge 0 * m = 0 \wedge m * 1 = m \wedge 1 * m = m$$

$$\text{mod\_less\_thm} \quad \vdash \forall m n \bullet 0 < n \Rightarrow m \text{ Mod } n < n$$

$$\text{div\_mod\_thm} \quad \vdash \forall m n \bullet 0 < n \Rightarrow m = m \text{ Div } n * n + m \text{ Mod } n$$

**div\_mod\_unique\_thm**

$$\vdash \forall m n d r$$

$$\bullet r < n \Rightarrow m = d * n + r \Rightarrow d = m \text{ Div } n \wedge r = m \text{ Mod } n$$

**minus\_clauses**

$$\vdash \forall m n$$

$$\bullet m - m = 0$$

$$\wedge m - 0 = m$$

$$\wedge (m + n) - n = m$$

$$\wedge (m + n) - m = n$$

**58 THE THEORY  $\mathbb{Z}$** **58.1 Parents***sets***58.2 Children***dyadic***58.3 Constants**

$$\text{Is\_Z\_Rep} \quad (\mathbb{N} \times \mathbb{N}) \text{ SET} \rightarrow \text{BOOL}$$

$$\text{NZ} \quad \mathbb{N} \rightarrow \mathbb{Z}$$

$$\text{\$}\sim\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\text{\$}+\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\text{\$}-\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\text{\$}\ast\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\text{\$}\leq\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$$

$$\text{\$}<\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$$

$$\text{\$}\geq\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$$

$$\text{\$}>\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$$

$$\text{\$}Abs\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\text{\$}Mod\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\text{\$}Div\mathbf{z} \quad \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$$

## 58.4 Aliases

$+$	$\$+_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$
$-$	$\$-_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$
$\sim$	$\$\sim_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z}$
$*$	$\$*_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$
$\leq$	$\$\leq_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$
$<$	$\$<_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$
$\geq$	$\$\geq_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$
$>$	$\$>_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{BOOL}$
<b>Abs</b>	$\$Abs_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z}$
<b>Div</b>	$\$Div_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$
<b>Mod</b>	$\$Mod_{\mathbb{Z}} : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z}$

## 58.5 Types

 $\mathbb{Z}$ 

## 58.6 Fixity

*Left Infix 305:* $-_{\mathbb{Z}}$ *Left Infix 315:***Div<sub>z</sub> Mod<sub>z</sub>***Right Infix 210:* $<_{\mathbb{Z}} \quad >_{\mathbb{Z}} \quad \leq_{\mathbb{Z}} \quad \geq_{\mathbb{Z}}$ *Right Infix 300:* $+_{\mathbb{Z}}$ *Right Infix 310:* $*_{\mathbb{Z}}$ *Prefix 350:***Abs<sub>z</sub> ~<sub>z</sub>**

## 58.7 Definitions

**Is\_ℤ\_Rep**  $\vdash \forall a \bullet \text{Is\_}\mathbb{Z}\text{-Rep } a \Leftrightarrow (\exists m \ n \bullet a = \{(x, y) \mid m + y = n + x\})$  $\mathbb{Z}$ **ℤ\_def**  $\vdash \exists f \bullet \text{TypeDefn Is\_}\mathbb{Z}\text{-Rep } f$  $+_{\mathbb{Z}}$  $\sim_{\mathbb{Z}}$ **NZ** $\vdash \text{ConstSpec}$  $(\lambda (+'_{\mathbb{Z}}, \sim'_{\mathbb{Z}}, \$\text{NZ}') \bullet$  $\bullet (\forall i \ j \ k$  $\bullet +'_{\mathbb{Z}} (+'_{\mathbb{Z}} i \ j) \ k = +'_{\mathbb{Z}} i (+'_{\mathbb{Z}} j \ k)$  $\wedge +'_{\mathbb{Z}} i \ j = +'_{\mathbb{Z}} j \ i$  $\wedge +'_{\mathbb{Z}} i (\sim'_{\mathbb{Z}} i) = \$\text{NZ}' \ 0$  $\wedge +'_{\mathbb{Z}} i (\$\text{NZ}' \ 0) = i)$  $\wedge (\forall m \ n$

---

	<ul style="list-style-type: none"> <li>• <math>+'_Z (\\$NZ' m) (\\$NZ' n)</math></li> <li style="padding-left: 20px;"><math>= \\$NZ' (m + n)</math></li> <li><math>\wedge \text{OneOne } \\$NZ'</math></li> <li><math>\wedge (\forall i</math></li> <li style="padding-left: 20px;">• <math>\exists m \bullet i = \\$NZ' m \vee i = \sim'_Z (\\$NZ' m))</math></li> </ul>
	( $\$, \sim, NZ$ )
$-_Z$	$\vdash \forall i j \bullet i - j = i + \sim j$
$*_Z$	$\vdash \text{ConstSpec}$ $(\lambda *_Z$ <ul style="list-style-type: none"> <li>• <math>\forall i j k</math></li> <li style="padding-left: 20px;">• <math>*'_Z i (j + k) = *_Z i j + *_Z i k</math></li> <li style="padding-left: 40px;"><math>\wedge *_Z i (NZ 1) = i</math></li> </ul>
	$\$*$
$\leq_Z$	$\vdash \forall i j \bullet i \leq j \Leftrightarrow (\exists m \bullet i + NZ m = j)$
$<_Z$	$\vdash \forall i j \bullet i < j \Leftrightarrow i + NZ 1 \leq j$
$\geq_Z$	$\vdash \forall i j \bullet i \geq j \Leftrightarrow j \leq i$
$>_Z$	$\vdash \forall i j \bullet i > j \Leftrightarrow j < i$
<b>Abs<sub>Z</sub></b>	$\vdash \forall i \bullet \text{Abs } i = (\text{if } NZ 0 \leq i \text{ then } i \text{ else } \sim i)$
<b>Div<sub>Z</sub></b>	
<b>Mod<sub>Z</sub></b>	$\vdash \text{ConstSpec}$ $(\lambda (\text{Div}'_Z, \text{Mod}'_Z)$ <ul style="list-style-type: none"> <li>• <math>\forall i j</math></li> <li style="padding-left: 20px;">• <math>\neg j = NZ 0</math></li> <li style="padding-left: 40px;"><math>\Rightarrow i = \text{Div}'_Z i j * j + \text{Mod}'_Z i j</math></li> <li style="padding-left: 60px;"><math>\wedge NZ 0 \leq \text{Mod}'_Z i j</math></li> <li style="padding-left: 60px;"><math>\wedge \text{Mod}'_Z i j &lt; \text{Abs } j</math></li> </ul>
	( $\$Div, \$Mod$ )

## 58.8 Theorems

### is\_Z\_rep\_consistent\_thm

 $\vdash \exists a \bullet \text{Is-Z-Rep } a$ 
 $+_Z$ -consistent

 $\sim_Z$ -consistent

 $NZ$ -consistent

 $\vdash \text{Consistent}$ 
 $(\lambda (+'_Z, \sim'_Z, \$NZ')$ 

- $(\forall i j k$ 
  - $+'_Z (+'_Z i j) k = +'_Z i (+'_Z j k)$
  - $\wedge +'_Z i j = +'_Z j i$
  - $\wedge +'_Z i (\sim'_Z i) = \$NZ' 0$
  - $\wedge +'_Z i (\$NZ' 0) = i$
- $\wedge (\forall m n$ 
  - $+'_Z (\$NZ' m) (\$NZ' n)$
  - $= \$NZ' (m + n)$
- $\wedge \text{OneOne } \$NZ'$
- $\wedge (\forall i$ 
  - $\exists m \bullet i = \$NZ' m \vee i = \sim'_Z (\$NZ' m))$

**Z\_plus\_comm\_thm**

$$\vdash \forall i j \bullet i + j = j + i$$

**Z\_plus\_assoc\_thm**

$$\vdash \forall i j k \bullet (i + j) + k = i + j + k$$

**Z\_plus\_assoc\_thm1**

$$\vdash \forall i j k \bullet i + j + k = (i + j) + k$$

**Z\_plus\_order\_thm**

$$\begin{aligned} \vdash \forall i j k \\ \bullet j + i = i + j \\ \wedge (i + j) + k = i + j + k \\ \wedge j + i + k = i + j + k \end{aligned}$$

**Z\_cases\_thm**  $\vdash \forall i \bullet \exists m \bullet i = \text{NZ } m \vee i = \sim (\text{NZ } m)$ **Z\_plus0\_thm**  $\vdash \forall i \bullet i + \text{NZ } 0 = i \wedge \text{NZ } 0 + i = i$ **Z\_plus\_minus\_thm**

$$\vdash \forall i \bullet i + \sim i = \text{NZ } 0 \wedge \sim i + i = \text{NZ } 0$$

**Z\_eq\_thm**

$$\vdash \forall i j \bullet i = j \Leftrightarrow i + \sim j = \text{NZ } 0$$

**NZ\_plus\_homomorphism\_thm**

$$\vdash \forall m n \bullet \text{NZ } (m + n) = \text{NZ } m + \text{NZ } n$$

**Z\_minus\_clauses**

$$\begin{aligned} \vdash \forall i j \\ \bullet \sim (\sim i) = i \\ \wedge i + \sim i = \text{NZ } 0 \\ \wedge \sim i + i = \text{NZ } 0 \\ \wedge \sim (i + j) = \sim i + \sim j \\ \wedge \sim (\text{NZ } 0) = \text{NZ } 0 \end{aligned}$$

**Z\_cases\_thm1**  $\vdash \forall i \bullet \exists m \bullet i = \text{NZ } m \vee i = \sim (\text{NZ } (m + 1))$ **Z\_induction\_thm**

$$\begin{aligned} \vdash \forall p \\ \bullet p (\text{NZ } 1) \\ \wedge (\forall i \bullet p i \Rightarrow p (\sim i)) \\ \wedge (\forall i j \bullet p i \wedge p j \Rightarrow p (i + j)) \\ \Rightarrow (\forall i \bullet p i) \end{aligned}$$

**NZ\_one\_one\_thm**

$$\vdash \forall m n \bullet \text{NZ } m = \text{NZ } n \Leftrightarrow m = n$$

**Z\_plus\_clauses**

$$\begin{aligned} \vdash \forall i j k \\ \bullet (i + k = j + k \Leftrightarrow i = j) \\ \wedge (k + i = j + k \Leftrightarrow i = j) \\ \wedge (i + k = k + j \Leftrightarrow i = j) \\ \wedge (k + i = k + j \Leftrightarrow i = j) \\ \wedge (i + k = k \Leftrightarrow i = \text{NZ } 0) \\ \wedge (k + i = k \Leftrightarrow i = \text{NZ } 0) \\ \wedge (k = k + j \Leftrightarrow j = \text{NZ } 0) \\ \wedge (k = j + k \Leftrightarrow j = \text{NZ } 0) \\ \wedge i + \text{NZ } 0 = i \\ \wedge \text{NZ } 0 + i = i \\ \wedge \neg \text{NZ } 1 = \text{NZ } 0 \\ \wedge \neg \text{NZ } 0 = \text{NZ } 1 \end{aligned}$$

---

**$\mathbb{Z}_{\leq} \leq 0$ \_thm**  $\vdash \forall i j \bullet i \leq j \Leftrightarrow i + \sim j \leq \text{NZ } 0$

**$\mathbb{Z}_{\text{minus}} \leq$ \_thm**

$\vdash \forall i j \bullet \sim i \leq \sim j \Leftrightarrow j \leq i$

**$\mathbb{Z}_{\leq} \text{minus}$ \_thm**

$\vdash \forall i j \bullet i \leq j \Leftrightarrow \sim j \leq \sim i$

**$\mathbb{Z}_{\leq}$ \_clauses**

$\vdash \forall i j k$

- $(i + k \leq j + k \Leftrightarrow i \leq j)$
- $\wedge (k + i \leq j + k \Leftrightarrow i \leq j)$
- $\wedge (i + k \leq k + j \Leftrightarrow i \leq j)$
- $\wedge (k + i \leq k + j \Leftrightarrow i \leq j)$
- $\wedge (i + k \leq k \Leftrightarrow i \leq \text{NZ } 0)$
- $\wedge (k + i \leq k \Leftrightarrow i \leq \text{NZ } 0)$
- $\wedge (k \leq k + j \Leftrightarrow \text{NZ } 0 \leq j)$
- $\wedge (k \leq j + k \Leftrightarrow \text{NZ } 0 \leq j)$
- $\wedge i \leq i$
- $\wedge \neg \text{NZ } 1 \leq \text{NZ } 0$
- $\wedge \text{NZ } 0 \leq \text{NZ } 1$

**$\text{NZ}_{\leq}$ \_thm**  $\vdash \forall m n \bullet \text{NZ } m \leq \text{NZ } n \Leftrightarrow m \leq n$

**\* $\mathbb{Z}$ \_consistent**

$\vdash \text{Consistent}$

$(\lambda *'_Z$

•  $\forall i j k$

•  $*'_Z i (j + k) = *'_Z i j + *'_Z i k$

•  $\wedge *'_Z i (\text{NZ } 1) = i)$

**$\text{NZ}_{\text{times}}$ \_homomorphism\_thm**

$\vdash \forall m n \bullet \text{NZ } (m * n) = \text{NZ } m * \text{NZ } n$

**$\mathbb{Z}_{\text{times}}$ \_minus\_thm**

$\vdash \forall i j$

•  $\sim i * j = \sim (i * j)$

•  $\wedge i * \sim j = \sim (i * j)$

•  $\wedge \sim i * \sim j = i * j$

**$\mathbb{Z}_{\text{times}}$ \_comm\_thm**

$\vdash \forall i j \bullet i * j = j * i$

**$\mathbb{Z}_{\text{times}}$ \_assoc\_thm**

$\vdash \forall i j k \bullet (i * j) * k = i * j * k$

**$\text{Div}_Z$ \_consistent**

**$\text{Mod}_Z$ \_consistent**

$\vdash \text{Consistent}$

$(\lambda (\text{Div}'_Z, \text{Mod}'_Z)$

•  $\forall i j$

•  $\neg j = \text{NZ } 0$

•  $\Rightarrow i = \text{Div}'_Z i j * j + \text{Mod}'_Z i j$

•  $\wedge \text{NZ } 0 \leq \text{Mod}'_Z i j$

•  $\wedge \text{Mod}'_Z i j < \text{Abs } j)$

**$\text{NZ}_{\text{plus}}$ \_homomorphism\_thm1**

$\vdash \forall m n \bullet \text{NZ } m + \text{NZ } n = \text{NZ } (m + n)$

**$\mathbb{Z}_{\text{N}}$ \_induction\_thm**

$\vdash \forall p$

---

$\bullet p (\text{NZ } 0) \wedge (\forall i \bullet \text{NZ } 0 \leq i \wedge p i \Rightarrow p (i + \text{NZ } 1))$   
 $\Rightarrow (\forall m \bullet \text{NZ } 0 \leq m \Rightarrow p m)$

**Z\_N\_plus\_thm**  $\vdash \forall i j \bullet \text{NZ } 0 \leq i \wedge \text{NZ } 0 \leq j \Rightarrow \text{NZ } 0 \leq i + j$

**Z\_N\_plus1\_thm**  
 $\vdash \forall i \bullet \text{NZ } 0 \leq i \Rightarrow \text{NZ } 0 \leq i + \text{NZ } 1$

**Z\_minus\_thm**  $\vdash \forall i j$   
 $\bullet \sim (\sim i) = i$   
 $\wedge i + \sim i = \text{NZ } 0$   
 $\wedge \sim i + i = \text{NZ } 0$   
 $\wedge \sim (i + j) = \sim i + \sim j$   
 $\wedge \sim (\text{NZ } 0) = \text{NZ } 0$

**Z\_N\_cases\_thm**  
 $\vdash \forall i$   
 $\bullet \text{NZ } 0 \leq i$   
 $\Rightarrow i = \text{NZ } 0 \vee (\exists j \bullet \text{NZ } 0 \leq j \wedge i = j + \text{NZ } 1)$

**Z\_N\_not\_minus\_thm**  
 $\vdash \forall i \bullet \text{NZ } 0 \leq i \Rightarrow i = \text{NZ } 0 \vee \neg \text{NZ } 0 \leq \sim i$

**Z\_not\_N\_thm**  $\vdash \forall i \bullet \neg \text{NZ } 0 \leq i \Rightarrow \text{NZ } 0 \leq \sim i$

**Z\_plus\_eq\_thm**  
 $\vdash \forall i j k \bullet i + j = k \Leftrightarrow i = k + \sim j$

**Z\_N\_not\_plus1\_thm**  
 $\vdash \forall i \bullet \text{NZ } 0 \leq i \Rightarrow \neg i + \text{NZ } 1 = \text{NZ } 0$

**Z\_times\_assoc\_thm1**  
 $\vdash \forall i j k \bullet i * j * k = (i * j) * k$

**Z\_times\_order\_thm**  
 $\vdash \forall i j k$   
 $\bullet j * i = i * j$   
 $\wedge (i * j) * k = i * j * k$   
 $\wedge j * i * k = i * j * k$

**NZ\_times\_homomorphism\_thm1**  
 $\vdash \forall m n \bullet \text{NZ } m * \text{NZ } n = \text{NZ } (m * n)$

**Z\_times1\_thm**  $\vdash \forall i \bullet i * \text{NZ } 1 = i \wedge \text{NZ } 1 * i = i$

**Z\_times\_plus\_distrib\_thm**  
 $\vdash \forall i j k$   
 $\bullet i * (j + k) = i * j + i * k$   
 $\wedge (i + j) * k = i * k + j * k$

**Z\_times0\_thm**  $\vdash \forall i \bullet \text{NZ } 0 * i = \text{NZ } 0 \wedge i * \text{NZ } 0 = \text{NZ } 0$

**Z\_eq\_thm1**  $\vdash \forall i j \bullet i = j \Leftrightarrow \sim i + j = \text{NZ } 0$

**Z\_times\_eq\_0\_thm**  
 $\vdash \forall i j \bullet i * j = \text{NZ } 0 \Leftrightarrow i = \text{NZ } 0 \vee j = \text{NZ } 0$

**Z\_times\_clauses**  
 $\vdash \forall i j$   
 $\bullet \text{NZ } 0 * i = \text{NZ } 0$   
 $\wedge i * \text{NZ } 0 = \text{NZ } 0$   
 $\wedge i * \text{NZ } 1 = i$   
 $\wedge \text{NZ } 1 * i = i$

**Z\_N\_times\_thm**  
 $\vdash \forall i j \bullet \text{NZ } 0 \leq i \wedge \text{NZ } 0 \leq j \Rightarrow \text{NZ } 0 \leq i * j$

---

**$\mathbb{Z}_{\leq}$ \_trans\_thm**

$$\vdash \forall i j k \bullet i \leq j \wedge j \leq k \Rightarrow i \leq k$$

 **$\mathbb{Z}_{\leq}$ \_cases\_thm**

$$\vdash \forall i j \bullet i \leq j \vee j \leq i$$

 **$\mathbb{Z}_{\leq}$ \_refl\_thm**

$$\vdash \forall i \bullet i \leq i$$

 **$\mathbb{Z}_{\leq}$ \_0\_thm1**

$$\vdash \forall i j \bullet i \leq j \Leftrightarrow \text{NZ } 0 \leq j + \sim i$$

 **$\mathbb{Z}_{\leq}$ \_antisym\_thm**

$$\vdash \forall i j \bullet i \leq j \wedge j \leq i \Rightarrow i = j$$

 **$\mathbb{Z}_{\text{less}}$ \_trans\_thm**

$$\vdash \forall i j k \bullet i < j \wedge j < k \Rightarrow i < k$$

 **$\mathbb{Z}_{\text{less}}$ \_irrefl\_thm**

$$\vdash \forall i j \bullet \neg (i < j \wedge j < i)$$

 **$\mathbb{Z}_{\text{less}}$ \_cases\_thm**

$$\vdash \forall i j \bullet i < j \vee i = j \vee j < i$$

 **$\text{NZ}_{\text{less}}$ \_thm**

$$\vdash \forall m n \bullet \text{NZ } m < \text{NZ } n \Leftrightarrow m < n$$

 **$\mathbb{Z}_{\text{less}}$ \_less\_0\_thm**

$$\vdash \forall i j \bullet i < j \Leftrightarrow i + \sim j < \text{NZ } 0$$

 **$\mathbb{Z}_{\text{less}}$ \_less\_0\_thm1**

$$\vdash \forall i j \bullet i < j \Leftrightarrow \text{NZ } 0 < j + \sim i$$

 **$\mathbb{Z}_{\text{minus}}$ \_less\_thm**

$$\vdash \forall i j \bullet \sim i < \sim j \Leftrightarrow j < i$$

 **$\mathbb{Z}_{\neg}$ \_less\_thm**

$$\vdash \forall i j \bullet \neg i < j \Leftrightarrow j \leq i$$

 **$\mathbb{Z}_{\neg}$ \_less\_thm**

$$\vdash \forall i j \bullet \neg i \leq j \Leftrightarrow j < i$$

 **$\mathbb{Z}_{\leq}$ \_less\_eq\_thm**

$$\vdash \forall i j \bullet i \leq j \Leftrightarrow i < j \vee i = j$$

 **$\mathbb{Z}_{\text{less}}$ \_less\_trans\_thm**

$$\vdash \forall i j k \bullet i < j \wedge j \leq k \Rightarrow i < k$$

 **$\mathbb{Z}_{\leq}$ \_less\_trans\_thm**

$$\vdash \forall i j k \bullet i \leq j \wedge j < k \Rightarrow i < k$$

 **$\mathbb{Z}_{\text{minus}}$ \_NZ\_less\_thm**

$$\vdash \forall i m \bullet i + \sim (\text{NZ } m) \leq i$$

 **$\mathbb{Z}_{\leq}$ \_plus\_NZ\_thm**

$$\vdash \forall i m \bullet i \leq i + \text{NZ } m$$

 **$\mathbb{Z}_{\in}$ \_NZ\_thm**

$$\vdash \forall i \bullet \text{NZ } 0 \leq i \Leftrightarrow (\exists m \bullet i = \text{NZ } m)$$

 **$\mathbb{Z}_{\text{less}}$ \_clauses**

$$\begin{aligned} &\vdash \forall i j k \\ &\bullet (i + k < j + k \Leftrightarrow i < j) \\ &\quad \wedge (k + i < j + k \Leftrightarrow i < j) \\ &\quad \wedge (i + k < k + j \Leftrightarrow i < j) \\ &\quad \wedge (k + i < k + j \Leftrightarrow i < j) \\ &\quad \wedge (i + k < k \Leftrightarrow i < \text{NZ } 0) \\ &\quad \wedge (k + i < k \Leftrightarrow i < \text{NZ } 0) \\ &\quad \wedge (i < k + i \Leftrightarrow \text{NZ } 0 < k) \\ &\quad \wedge (i < i + k \Leftrightarrow \text{NZ } 0 < k) \\ &\quad \wedge \neg i < i \\ &\quad \wedge \text{NZ } 0 < \text{NZ } 1 \\ &\quad \wedge \neg \text{NZ } 1 < \text{NZ } 0 \end{aligned}$$

 **$\mathbb{Z}_{\text{N}}$ \_abs\_thm**

$$\vdash \forall m \bullet \text{Abs } (\text{NZ } m) = \text{NZ } m \wedge \text{Abs } (\sim (\text{NZ } m)) = \text{NZ } m$$

---

**$\mathbb{Z}$ \_abs\_thm**      $\vdash \forall i \bullet \text{NZ } 0 \leq i \Rightarrow \text{Abs } i = i \wedge \text{Abs } (\sim i) = i$   
 **$\mathbb{Z}$ \_abs\_N\_thm**    $\vdash \forall i \bullet \text{NZ } 0 \leq \text{Abs } i$   
 **$\mathbb{Z}$ \_abs\_eq\_0\_thm**  
                    $\vdash \forall i \bullet \text{Abs } i = \text{NZ } 0 \Leftrightarrow i = \text{NZ } 0$   
 **$\mathbb{Z}$ \_abs\_minus\_thm**  
                    $\vdash \forall i \bullet \text{Abs } (\sim i) = \text{Abs } i$   
 **$\mathbb{Z}$ \_N\_abs\_minus\_thm**  
                    $\vdash \forall i \ j$   
                   •  $\text{NZ } 0 \leq i \wedge \text{NZ } 0 \leq j \wedge j \leq i \Rightarrow \text{Abs } (i + \sim j) \leq i$   
 **$\mathbb{Z}$ \_abs\_times\_thm**  
                    $\vdash \forall i \ j \bullet \text{Abs } (i * j) = \text{Abs } i * \text{Abs } j$   
 **$\mathbb{Z}$ \_abs\_plus\_thm**  
                    $\vdash \forall i \ j \bullet \text{Abs } (i + j) \leq \text{Abs } i + \text{Abs } j$   
 **$\mathbb{Z}$ \_div\_mod\_unique\_lemma1**  
                    $\vdash \forall i \ j \bullet \text{NZ } 0 \leq i \wedge \text{NZ } 0 \leq j \wedge i * j < j \Rightarrow i = \text{NZ } 0$   
 **$\mathbb{Z}$ \_div\_mod\_unique\_lemma2**  
                    $\vdash \forall j \ d \ r$   
                   •  $\neg j = \text{NZ } 0$   
                    $\Rightarrow d * j + r = \text{NZ } 0 \wedge \text{NZ } 0 \leq r \wedge r < \text{Abs } j$   
                    $\Rightarrow d = \text{NZ } 0 \wedge r = \text{NZ } 0$   
 **$\mathbb{Z}$ \_div\_mod\_unique\_lemma3**  
                    $\vdash \forall i \ j \ d \ r \ D \ R$   
                   •  $\neg j = \text{NZ } 0$   
                    $\Rightarrow D * j + R = d * j + r$   
                    $\wedge \text{NZ } 0 \leq r$   
                    $\wedge r \leq R$   
                    $\wedge R < \text{Abs } j$   
                    $\Rightarrow D = d \wedge R = r$   
 **$\mathbb{Z}$ \_div\_mod\_unique\_thm**  
                    $\vdash \forall i \ j \ d \ r$   
                   •  $\neg j = \text{NZ } 0$   
                    $\Rightarrow (i = d * j + r \wedge \text{NZ } 0 \leq r \wedge r < \text{Abs } j$   
                    $\Leftrightarrow d = i \text{ Div } j \wedge r = i \text{ Mod } j)$   
 **$\mathbb{Z}$ \_≤\_induction\_thm**  
                    $\vdash \forall j \ p$   
                   •  $p \ j \wedge (\forall i \bullet j \leq i \wedge p \ i \Rightarrow p \ (i + \text{NZ } 1))$   
                    $\Rightarrow (\forall i \bullet j \leq i \Rightarrow p \ i)$   
 **$\mathbb{Z}$ \_cov\_induction\_thm**  
                    $\vdash \forall j \ p$   
                   •  $(\forall i \bullet j \leq i \wedge (\forall k \bullet j \leq k \wedge k < i \Rightarrow p \ k) \Rightarrow p \ i)$   
                    $\Rightarrow (\forall i \bullet j \leq i \Rightarrow p \ i)$   
 **$\mathbb{Z}$ \_fun\_∃\_thm**    $\vdash \forall f \ g \ z$   
                   •  $(\forall x \bullet g \ (f \ x) = x) \wedge (\forall y \bullet f \ (g \ y) = y)$   
                    $\Rightarrow (\exists_1 \ h$   
                   •  $h \ (\text{NZ } 0) = z$   
                    $\wedge (\forall i \bullet h \ (i + \text{NZ } 1) = f \ (h \ i))$   
                    $\wedge (\forall i \bullet h \ (i - \text{NZ } 1) = g \ (h \ i))$

---

## 59 INDEX

**	70	- <sub>R</sub>	294
**	71	- <sub>R</sub>	295
* <sub>1</sub>	63	- <sub>R</sub>	296
* <sub>1</sub>	71	- <sub>Z</sub>	317
* <sub>1</sub>	77	- <sub>Z</sub>	318
* <sub>2</sub>	63	- <sub>Z</sub>	319
* <sub>2</sub>	71	-	295
* <sub>2</sub>	77	-	313
* <sub>3</sub>	63	-	314
* <sub>3</sub>	71	-	318
* <sub>3</sub>	77	..	274
* <sub>4</sub>	63	..	275
* <sub>4</sub>	71	/ <sub>N</sub>	294
* <sub>4</sub>	77	/ <sub>N</sub>	295
* <sub>5</sub>	63	/ <sub>N</sub>	297
* <sub>5</sub>	71	/ <sub>R-consistent</sub>	308
* <sub>5</sub>	77	/ <sub>R</sub>	294
* <sub>6</sub>	63	/ <sub>R</sub>	295
* <sub>6</sub>	71	/ <sub>R</sub>	296
* <sub>6</sub>	77	/	295
* <sub>7</sub>	63	0 <sub>R-consistent</sub>	300
* <sub>7</sub>	71	0 <sub>R</sub>	294
* <sub>7</sub>	77	0 <sub>R</sub>	296
* <sub>R-consistent</sub>	307	1 <sub>R-consistent</sub>	300
* <sub>R</sub>	294	1 <sub>R</sub>	294
* <sub>R</sub>	295	1 <sub>R</sub>	296
* <sub>R</sub>	296	<<<	268
* <sub>Z-consistent</sub>	321	<<	268
* <sub>Z</sub>	317	< <sub>R-consistent</sub>	297
* <sub>Z</sub>	318	< <sub>R</sub>	294
* <sub>Z</sub>	319	< <sub>R</sub>	295
*	69	< <sub>Z</sub>	317
*	70	< <sub>Z</sub>	318
*	295	< <sub>Z</sub>	319
*	313	<	295
*	314	<	313
*	318	<	314
+ <sub>R-consistent</sub>	300	<	318
+ <sub>R</sub>	294	=	263
+ <sub>R</sub>	295	> <sub>R</sub>	294
+ <sub>R</sub>	296	> <sub>R</sub>	295
+ <sub>Z-consistent</sub>	319	> <sub>R</sub>	296
+ <sub>Z</sub>	317	> <sub>Z</sub>	317
+ <sub>Z</sub>	318	> <sub>Z</sub>	318
+	281	> <sub>Z</sub>	319
+	295	>	295
+	313	>	313
+	314	>	314
+	318	>	318
,	272	@	253
,	273	@	258

@	259	And	63
AbsChar	10	And	73
absState_consistent	40	ANSWER	118
absState_consistent	116	Antisymmetric	255
absState <sub>t</sub> _consistent	117	Antisym	267
absState <sub>t</sub>	113	Antisym	268
absState <sub>t</sub>	115	append_assoc_thm	290
absState	28	append_cancel_thm	290
absState	29	append_def	260
abs_char_rep_char_def	10	append_empty_thm	290
Abs <sub>R</sub>	294	Append	258
Abs <sub>R</sub>	296	Append	260
Abs <sub>Z</sub>	317	applyAnd	63
Abs <sub>Z</sub>	318	applyAnd	74
Abs <sub>Z</sub>	319	applyEqual	63
Abs	295	applyEqual	76
Abs	318	applyNot	63
accessDenied	16	applyNot	73
accessDenied	21	applyOr	63
Act <sub>t</sub> _lemma	190	applyOr	75
Act <sub>t</sub>	131	applyPlus	63
Act <sub>t</sub>	135	applyPlus	75
AllTuples_lemma1	222	apply	63
AllTuples_lemma2	228	apply	76
AllTuples_OK <sub>c</sub> _lemma	228	Arbitrary	264
AllTuples_OK <sub>d</sub> _lemma	228	Architecture_Secure	215
AllTuples	194	Architecture_Secure	216
AllTuples	203	at2	136
all_binop	65	at2	143
all_binop	84	at3	136
all_columns	66	at3	143
all_columns	83	at4	136
all_data_columns <sub>local</sub>	156	at4	143
all_data_columns <sub>local</sub>	177	at_at_eq_thm	254
all_false	67	At_consistent	254
all_false	91	at_thm1	285
all_tuples	65	at_thm	252
all_tuples	84	At	253
ambiguousColumn	16	auxapply	66
ambiguousColumn	21	auxapply	86
ambiguousEvaluate	16	BEHAVIOURS	14
ambiguousEvaluate	21	behaviours	35
ambiguousHaving	16	behaviours	36
ambiguousHaving	21	Behaviours	235
ambiguousName	153	Behaviours	238
ambiguousName	160	BinOpAnd_OK <sub>c</sub> _lemma	214
ambiguousUpdate	16	BinOpAnd_OK <sub>d</sub> _lemma	212
ambiguousUpdate	21	BinOpAnd	192
Am	35	BinOpAnd	196
andBools	66	BinOpOr_OK <sub>c</sub> _lemma	214
andBools	86	BinOpOr_OK <sub>d</sub> _lemma	212
andb	66	BinOpOr	192
andb	71	BinOpOr	196
andb	87	BinOp_OK <sub>c</sub> _lemma	214

<i>BinOp_OK<sub>d</sub>-lemma</i> . . . . .	213	<i>CC-uniform</i> . . . . .	24
<i>binop_type</i> . . . . .	156	<i>CC-unique</i> . . . . .	18
<i>binop_type</i> . . . . .	177	<i>CC-unique</i> . . . . .	24
<i>binop</i> . . . . .	65	<i>Chains</i> . . . . .	255
<i>binop</i> . . . . .	84	<i>changeSpec</i> . . . . .	28
<i>BinOp</i> . . . . .	192	<i>changeSpec</i> . . . . .	31
<i>BinOp</i> . . . . .	196	<i>charsType</i> . . . . .	67
<i>bin_rel_ext_clauses</i> . . . . .	8	<i>charsType</i> . . . . .	88
<i>bin_rel_insert_thm</i> . . . . .	9	<i>chars</i> . . . . .	19
<i>bin_rel_∅_universe_thm</i> . . . . .	8	<i>char_def</i> . . . . .	10
<i>booleanType</i> . . . . .	67	<i>CHAR</i> . . . . .	10
<i>booleanType</i> . . . . .	88	<i>Char</i> . . . . .	10
<i>boolean</i> . . . . .	19	<i>Char</i> . . . . .	20
<i>BoolItem_OneOne-lemma</i> . . . . .	209	<i>checkComplete</i> . . . . .	64
<i>BoolItem</i> . . . . .	192	<i>checkComplete</i> . . . . .	81
<i>BoolItem</i> . . . . .	195	<i>checkFieldClasses</i> . . . . .	66
<i>BoolVal</i> . . . . .	17	<i>checkFieldClasses</i> . . . . .	88
<i>BoolVal</i> . . . . .	22	<i>checkGroup</i> . . . . .	66
<i>bool_cases_axiom</i> . . . . .	256	<i>checkGroup</i> . . . . .	87
<i>Bool</i> . . . . .	20	<i>checkIntegrity</i> . . . . .	66
<i>Bool</i> . . . . .	263	<i>checkIntegrity</i> . . . . .	88
<i>BoundedObs</i> . . . . .	232	<i>CheckList</i> . . . . .	192
<i>BoundedObs</i> . . . . .	233	<i>CheckList</i> . . . . .	197
<i>BoundInfo</i> . . . . .	142	<i>checkNulls</i> . . . . .	67
<i>CaseC</i> . . . . .	193	<i>checkNulls</i> . . . . .	90
<i>CaseC</i> . . . . .	197	<i>CheckTest</i> . . . . .	192
<i>CaseValue</i> . . . . .	193	<i>CheckTest</i> . . . . .	197
<i>CaseValue</i> . . . . .	198	<i>checkType</i> . . . . .	67
<i>CaseValValue</i> . . . . .	192	<i>checkType</i> . . . . .	90
<i>CaseValValue</i> . . . . .	197	<i>checkUniform</i> . . . . .	66
<i>CaseVal-lemma</i> . . . . .	211	<i>checkUniform</i> . . . . .	88
<i>CaseVal_OK<sub>c</sub>-lemma</i> . . . . .	214	<i>checkUniqueness</i> . . . . .	66
<i>CaseVal_OK<sub>d</sub>-lemma</i> . . . . .	213	<i>checkUniqueness</i> . . . . .	87
<i>CaseVal</i> . . . . .	193	<i>check_boolean</i> . . . . .	156
<i>CaseVal</i> . . . . .	197	<i>check_boolean</i> . . . . .	177
<i>Case_OK<sub>c</sub>-lemma</i> . . . . .	214	<i>check_enum</i> . . . . .	154
<i>Case_OK<sub>d</sub>-lemma</i> . . . . .	213	<i>check_enum</i> . . . . .	160
<i>case</i> . . . . .	136	<i>check_fixed</i> . . . . .	154
<i>CASE</i> . . . . .	137	<i>check_fixed</i> . . . . .	160
<i>CASE</i> . . . . .	143	<i>check_floating</i> . . . . .	154
<i>case</i> . . . . .	143	<i>check_floating</i> . . . . .	160
<i>Case</i> . . . . .	193	<i>check_interval</i> . . . . .	154
<i>Case</i> . . . . .	198	<i>check_interval</i> . . . . .	160
<i>CC_classLimited</i> . . . . .	18	<i>check_time</i> . . . . .	154
<i>CC_classLimited</i> . . . . .	24	<i>check_time</i> . . . . .	160
<i>CC_exist</i> . . . . .	18	<i>check_type_conversion</i> . . . . .	156
<i>CC_exist</i> . . . . .	24	<i>check_type_conversion</i> . . . . .	177
<i>CC_primary</i> . . . . .	18	<i>check_where_complete1</i> . . . . .	67
<i>CC_primary</i> . . . . .	24	<i>check_where_complete1</i> . . . . .	91
<i>CC_referential</i> . . . . .	18	<i>check_where_complete</i> . . . . .	64
<i>CC_referential</i> . . . . .	24	<i>check_where_complete</i> . . . . .	81
<i>CC_secondary</i> . . . . .	18	<i>Choose_consistent</i> . . . . .	280
<i>CC_secondary</i> . . . . .	24	<i>Choose</i> . . . . .	280
<i>CC-uniform</i> . . . . .	18	<i>ci_index</i> . . . . .	139

---

<i>ci_index</i> . . . . .	149	<i>cleanRow_lemma</i> . . . . .	41
<i>ci_lwb</i> . . . . .	139	<i>cleanRow_updateRow_lemma1</i> . . . . .	59
<i>ci_lwb</i> . . . . .	149	<i>cleanRow_updateRow_lemma</i> . . . . .	54
<i>ci_null_allowed</i> . . . . .	139	<i>cleanRow</i> . . . . .	28
<i>ci_null_allowed</i> . . . . .	149	<i>cleanRow</i> . . . . .	30
<i>ci_uniform</i> . . . . .	139	<i>cleanTable_deleteRows_lemma</i> . . . . .	58
<i>ci_uniform</i> . . . . .	149	<i>cleanTable_insertQuery_lemma</i> . . . . .	44
<i>ci_unique</i> . . . . .	139	<i>cleanTable_insertRows_lemma</i> . . . . .	56
<i>ci_unique</i> . . . . .	149	<i>cleanTable_lemma</i> . . . . .	41
<i>classChange</i> . . . . .	16	<i>cleanTable_updateQuery_lemma</i> . . . . .	48
<i>classChange</i> . . . . .	21	<i>cleanTable_updateRows_lemma</i> . . . . .	60
<i>Classification_OK<sub>c</sub>-lemma</i> . . . . .	214	<i>cleanTable</i> . . . . .	28
<i>Classification_OK<sub>a</sub>-lemma</i> . . . . .	211	<i>cleanTable</i> . . . . .	30
<i>classification</i> . . . . .	64	<i>clearance</i> . . . . .	65
<i>classification</i> . . . . .	84	<i>clearance</i> . . . . .	84
<i>Classification</i> . . . . .	193	<i>client_clearance</i> . . . . .	153
<i>Classification</i> . . . . .	200	<i>client_clearance</i> . . . . .	160
<i>Classified_value</i> . . . . .	69	<i>CodeVal</i> . . . . .	17
<i>Classified_value</i> . . . . .	71	<i>CodeVal</i> . . . . .	22
<i>Classified_value</i> . . . . .	102	<i>Code</i> . . . . .	19
<i>classify_default</i> . . . . .	65	<i>ColConS</i> . . . . .	18
<i>classify_default</i> . . . . .	84	<i>ColConS</i> . . . . .	25
<i>classify</i> . . . . .	65	<i>ColCon</i> . . . . .	19
<i>classify</i> . . . . .	84	<i>ColCon</i> . . . . .	24
<i>ClassItem</i> . . . . .	193	<i>colDefaults_lemma</i> . . . . .	61
<i>ClassItem</i> . . . . .	199	<i>colDefaults</i> . . . . .	29
<i>ClassName</i> . . . . .	124	<i>colDefaults</i> . . . . .	31
<i>ClassName</i> . . . . .	125	<i>ColNeeds</i> . . . . .	124
<i>classType</i> . . . . .	67	<i>ColNeeds</i> . . . . .	126
<i>classType</i> . . . . .	89	<i>colposns</i> . . . . .	66
<i>ClassUpdate</i> . . . . .	17	<i>colposns</i> . . . . .	86
<i>ClassUpdate</i> . . . . .	23	<i>colsInGroup</i> . . . . .	66
<i>ClassVal</i> . . . . .	17	<i>colsInGroup</i> . . . . .	86
<i>ClassVal</i> . . . . .	22	<i>colspecs</i> . . . . .	66
<i>class_bottom</i> . . . . .	113	<i>colspecs</i> . . . . .	87
<i>class_bottom</i> . . . . .	114	<i>ColSpec<sub>a</sub></i> . . . . .	131
<i>class_column</i> . . . . .	156	<i>ColSpec<sub>a</sub></i> . . . . .	133
<i>class_column</i> . . . . .	177	<i>ColSpec</i> . . . . .	19
<i>class_of_item</i> . . . . .	121	<i>ColSpec</i> . . . . .	23
<i>class_of_item</i> . . . . .	122	<i>ColType</i> . . . . .	142
<i>Class</i> . . . . .	14	<i>ColumnSpecification</i> . . . . .	142
<i>class</i> . . . . .	19	<i>column_data_test</i> . . . . .	156
<i>cleanColCons_fun_thm</i> . . . . .	42	<i>column_data_test</i> . . . . .	178
<i>cleanColCons_lemma</i> . . . . .	41	<i>col_exp</i> . . . . .	156
<i>cleanColCons</i> . . . . .	28	<i>col_exp</i> . . . . .	178
<i>cleanColCons</i> . . . . .	29	<i>Col_name</i> . . . . .	67
<i>cleanDirectory_lemma</i> . . . . .	42	<i>Col_name</i> . . . . .	70
<i>cleanDirectory</i> . . . . .	28	<i>Col_name</i> . . . . .	90
<i>cleanDirectory</i> . . . . .	30	<i>Col_spec</i> . . . . .	66
<i>cleanDir_fun_thm</i> . . . . .	42	<i>Col_spec</i> . . . . .	70
<i>cleanRows_errors_or_vals_lemma</i> . . . . .	47	<i>Col_spec</i> . . . . .	84
<i>cleanRows_size_lemma</i> . . . . .	46	<i>col_target</i> . . . . .	156
<i>cleanRows</i> . . . . .	28	<i>col_target</i> . . . . .	179
<i>cleanRows</i> . . . . .	30	<i>Col</i> . . . . .	71

---

<i>combine_def</i> .....	260	<i>Contents</i> .....	199
<i>Combine</i> .....	258	<i>contextual_data</i> .....	153
<i>Combine</i> .....	260	<i>contextual_data</i> .....	160
<i>CombI</i> .....	11	<i>convert_tableSpecification_backup</i> .....	157
<i>CombK</i> .....	11	<i>convert_tableSpecification_backup</i> .....	179
<i>CombS</i> .....	11	<i>convert_colspec</i> .....	156
<i>comb_i_def</i> .....	11	<i>convert_colspec</i> .....	177
<i>comb_k_def</i> .....	11	<i>convert_ssltype</i> .....	156
<i>comb_s_def</i> .....	11	<i>convert_ssltype</i> .....	179
<i>comma_def</i> .....	273	<i>convert_swordtype</i> .....	157
<i>CommonValue_OK<sub>c</sub>-lemma</i> .....	215	<i>convert_swordtype</i> .....	180
<i>CommonValue_OK<sub>d</sub>-lemma</i> .....	213	<i>convert_tableSpecifcation</i> .....	157
<i>CommonValue</i> .....	194	<i>convert_tableSpecifcation</i> .....	179
<i>CommonValue</i> .....	202	<i>convert_tablespec</i> .....	156
<i>comparability_thm</i> .....	255	<i>convert_tablespec</i> .....	179
<i>complement_clauses</i> .....	278	<i>convert_type</i> .....	157
<i>complement_def</i> .....	277	<i>convert_type</i> .....	180
<i>Complete</i> .....	267	<i>convert</i> .....	69
<i>Complete</i> .....	269	<i>convert</i> .....	107
<i>composite</i> .....	242	<i>Correct_Compile_OkSTP</i> .....	216
<i>composite</i> .....	244	<i>Correct_Compile_OkSTP</i> .....	217
<i>ComputeAnd_lemma</i> .....	211	<i>Correct_Compile_STP_secure_E</i> .....	216
<i>ComputeAnd</i> .....	192	<i>Correct_Compile_STP_secure_E</i> .....	217
<i>ComputeAnd</i> .....	196	<i>Correct_Compile</i> .....	131
<i>ComputeOr_lemma</i> .....	212	<i>Correct_Compile</i> .....	136
<i>ComputeOr</i> .....	192	<i>correlate_from</i> .....	65
<i>ComputeOr</i> .....	196	<i>correlate_from</i> .....	84
<i>ConditionE</i> .....	131	<i>CountAll_OK<sub>c</sub>-lemma</i> .....	214
<i>ConditionE</i> .....	135	<i>CountAll_OK<sub>d</sub>-lemma</i> .....	211
<i>cond_def</i> .....	264	<i>CountAll</i> .....	193
<i>cond_thm</i> .....	265	<i>CountAll</i> .....	199
<i>Cond</i> .....	264	<i>CountDistinct_OK<sub>c</sub>-lemma</i> .....	215
<i>conjunct1_fun_lemma</i> .....	50	<i>CountDistinct_OK<sub>d</sub>-lemma</i> .....	213
<i>conjunct1_lemma1</i> .....	51	<i>CountDistinct</i> .....	193
<i>conjunct1_lemma2</i> .....	51	<i>CountDistinct</i> .....	198
<i>conjunct1</i> .....	55	<i>CountNonNull_OK<sub>c</sub>-lemma</i> .....	215
<i>conjunct2</i> .....	61	<i>CountNonNull_OK<sub>d</sub>-lemma</i> .....	213
<i>conjunct3</i> .....	48	<i>CountNonNull</i> .....	193
<i>conjunct4</i> .....	44	<i>CountNonNull</i> .....	198
<i>constant_value<sub>data</sub></i> .....	157	<i>count_all</i> .....	65
<i>constant_value<sub>data</sub></i> .....	187	<i>count_all</i> .....	84
<i>constant_value<sub>type</sub></i> .....	156	<i>cov_induction_thm</i> .....	316
<i>constant_value<sub>type</sub></i> .....	179	<i>CS_consGroup</i> .....	17
<i>ConstraintInfo</i> .....	142	<i>CS_consGroup</i> .....	23
<i>ConstraintInfo</i> .....	149	<i>CS_default</i> .....	17
<i>cons</i> .....	66	<i>CS_default</i> .....	23
<i>cons</i> .....	87	<i>CS_dinaryType</i> .....	17
<i>Cons</i> .....	258	<i>CS_dinaryType</i> .....	23
<i>Cons</i> .....	260	<i>CS_ide</i> .....	17
<i>Contents_OK<sub>c</sub>-lemma</i> .....	214	<i>CS_ide</i> .....	23
<i>Contents_OK<sub>d</sub>-lemma</i> .....	211	<i>CS_max</i> .....	17
<i>contents</i> .....	65	<i>CS_max</i> .....	23
<i>contents</i> .....	84	<i>CS_min</i> .....	17
<i>Contents</i> .....	193	<i>CS_min</i> .....	23

---

<i>CS_nullType</i> .....	17	<i>Dat_item</i> .....	17
<i>CS_nullType</i> .....	23	<i>Dat_item</i> .....	23
<i>CS_posn</i> .....	17	<i>DBMS_TYPE</i> .....	118
<i>CS_posn</i> .....	23	<i>DCS_max</i> .....	130
<i>CS_sterlingType</i> .....	17	<i>DCS_max</i> .....	132
<i>CS_sterlingType</i> .....	23	<i>DCS_min</i> .....	130
<i>current_time</i> .....	65	<i>DCS_min</i> .....	132
<i>current_time</i> .....	84	<i>DCS_name</i> .....	130
<i>curry_def</i> .....	273	<i>DCS_name</i> .....	132
<i>Curry</i> .....	272	<i>default_directory</i> .....	153
<i>Curry</i> .....	273	<i>default_directory</i> .....	160
<i>cuts_complete_thm</i> .....	269	<i>DeleteEffect</i> .....	19
<i>cuts_strict_linear_order_thm</i> .....	269	<i>DeleteEffect</i> .....	26
<i>cuts_strict_partial_order_thm</i> .....	269	<i>deleteQuery_lemma</i> .....	53
<i>cuts_trich_thm</i> .....	269	<i>deleteQuery</i> .....	29
<i>cuts_unbounded_above_thm</i> .....	270	<i>deleteQuery</i> .....	32
<i>cuts_unbounded_below_thm</i> .....	270	<i>deleteRows_lemma</i> .....	53
<i>Cuts</i> .....	267	<i>Delete</i> .....	20
<i>Cuts</i> .....	269	<i>delete</i> .....	66
<i>c_anon<sub>s</sub></i> .....	138	<i>delete</i> .....	83
<i>c_anon<sub>s</sub></i> .....	147	<i>DenoteConstant_OK<sub>c</sub>_lemma</i> .....	214
<i>c_anon<sub>tc</sub></i> .....	139	<i>DenoteConstant_OK<sub>d</sub>_lemma</i> .....	211
<i>c_anon<sub>tc</sub></i> .....	148	<i>DenoteConstant</i> .....	192
<i>c_anon<sub>tn</sub></i> .....	140	<i>DenoteConstant</i> .....	195
<i>c_anon<sub>tn</sub></i> .....	150	<i>denote_class</i> .....	64
<i>c_anon<sub>t</sub></i> .....	138	<i>denote_class</i> .....	84
<i>c_anon<sub>t</sub></i> .....	147	<i>denote_code</i> .....	64
<i>c_anyType</i> .....	137	<i>denote_code</i> .....	84
<i>c_anyType</i> .....	145	<i>denote_col_name</i> .....	65
<i>c_booleanType</i> .....	137	<i>denote_col_name</i> .....	84
<i>c_booleanType</i> .....	145	<i>denote_col_spec</i> .....	65
<i>c_classType</i> .....	137	<i>denote_col_spec</i> .....	84
<i>c_classType</i> .....	145	<i>denote_false</i> .....	64
<i>c_codeType</i> .....	137	<i>denote_false</i> .....	84
<i>c_codeType</i> .....	145	<i>denote_float</i> .....	64
<i>c_constant_null</i> .....	140	<i>denote_float</i> .....	84
<i>c_constant_null</i> .....	149	<i>denote_integer</i> .....	64
<i>c_monoleanType</i> .....	137	<i>denote_integer</i> .....	84
<i>c_monoleanType</i> .....	145	<i>denote_interval</i> .....	64
<i>c_none<sub>t</sub></i> .....	138	<i>denote_interval</i> .....	84
<i>c_none<sub>t</sub></i> .....	147	<i>denote_name</i> .....	156
<i>c_nullType</i> .....	138	<i>denote_name</i> .....	178
<i>c_nullType</i> .....	145	<i>denote_null</i> .....	64
<i>dataList</i> .....	66	<i>denote_null</i> .....	84
<i>dataList</i> .....	87	<i>denote_string</i> .....	64
<i>DataS</i> .....	17	<i>denote_string</i> .....	84
<i>DataS</i> .....	23	<i>denote_table_spec</i> .....	65
<i>DataUpdate</i> .....	17	<i>denote_table_spec</i> .....	84
<i>DataUpdate</i> .....	23	<i>denote_time</i> .....	64
<i>data_components</i> .....	40	<i>denote_time</i> .....	84
<i>Data</i> .....	19	<i>denote_true</i> .....	64
<i>Data</i> .....	23	<i>denote_true</i> .....	84
<i>Dat_class</i> .....	17	<i>denote_void</i> .....	64
<i>Dat_class</i> .....	23	<i>denote_void</i> .....	84

---

---

<i>denote<sub>classExp</sub></i> .....	157	<i>destUpdate_consistent</i> .....	43
<i>denote<sub>classExp</sub></i> .....	180	<i>destUpdate</i> .....	19
<i>DenseIn</i> .....	267	<i>destUpdate</i> .....	27
<i>DenseIn</i> .....	268	<i>destValuedItem</i> .....	62
<i>dense_complete_subset_thm</i> .....	270	<i>destValuedItem</i> .....	71
<i>dense_superset_thm</i> .....	270	<i>destVal_consistent</i> .....	43
<i>dense_universe_thm</i> .....	270	<i>destVal_consistent</i> .....	191
<i>Dense</i> .....	267	<i>destVal_fun_thm</i> .....	50
<i>Dense</i> .....	268	<i>destVal</i> .....	19
<i>DerColSpec</i> .....	132	<i>destVal</i> .....	26
<i>DerTableRow</i> .....	132	<i>dest_absolute</i> .....	137
<i>DerTableSpec</i> .....	132	<i>dest_absolute</i> .....	144
<i>DerTable</i> .....	132	<i>dest_and</i> .....	140
<i>DerTable</i> .....	133	<i>dest_and</i> .....	150
<i>destBoolVal</i> .....	62	<i>dest_anonymous_column</i> .....	139
<i>destBoolVal</i> .....	71	<i>dest_anonymous_column</i> .....	149
<i>destClassVal</i> .....	62	<i>dest_column</i> .....	140
<i>destClassVal</i> .....	72	<i>dest_column</i> .....	149
<i>destClass_consistent</i> .....	49	<i>dest_constant_class</i> .....	140
<i>destClass</i> .....	17	<i>dest_constant_class</i> .....	149
<i>destClass</i> .....	23	<i>dest_constant<sub>ec</sub></i> .....	140
<i>destCodeVal</i> .....	62	<i>dest_constant<sub>ec</sub></i> .....	150
<i>destCodeVal</i> .....	71	<i>dest_constant<sub>tc</sub></i> .....	139
<i>destData_consistent</i> .....	49	<i>dest_constant<sub>tc</sub></i> .....	148
<i>destData</i> .....	17	<i>dest_constant</i> .....	138
<i>destData</i> .....	23	<i>dest_constant</i> .....	147
<i>destDelete_consistent</i> .....	43	<i>dest_default</i> .....	137
<i>destDelete</i> .....	19	<i>dest_default</i> .....	144
<i>destDelete</i> .....	27	<i>dest_enumType</i> .....	137
<i>destError_consistent</i> .....	43	<i>dest_enumType</i> .....	145
<i>destError_consistent</i> .....	191	<i>dest_fixedType</i> .....	137
<i>destError</i> .....	19	<i>dest_fixedType</i> .....	145
<i>destError</i> .....	26	<i>dest_intervalType</i> .....	137
<i>destFloatVal</i> .....	62	<i>dest_intervalType</i> .....	145
<i>destFloatVal</i> .....	71	<i>dest_local_identifier</i> .....	140
<i>destInsert_consistent</i> .....	43	<i>dest_local_identifier</i> .....	149
<i>destInsert</i> .....	19	<i>dest_name<sub>s</sub></i> .....	138
<i>destInsert</i> .....	27	<i>dest_name<sub>s</sub></i> .....	147
<i>destIntervalVal</i> .....	62	<i>dest_name<sub>tc</sub></i> .....	139
<i>destIntervalVal</i> .....	71	<i>dest_name<sub>tc</sub></i> .....	148
<i>destIntVal</i> .....	62	<i>dest_name<sub>tn</sub></i> .....	140
<i>destIntVal</i> .....	71	<i>dest_name<sub>tn</sub></i> .....	150
<i>destItem_consistent</i> .....	49	<i>dest_name<sub>t</sub></i> .....	138
<i>destItem</i> .....	17	<i>dest_name<sub>t</sub></i> .....	147
<i>destItem</i> .....	23	<i>dest_ors</i> .....	140
<i>destNullItem</i> .....	62	<i>dest_ors</i> .....	150
<i>destNullItem</i> .....	71	<i>dest_simple</i> .....	140
<i>destSelect_consistent</i> .....	43	<i>dest_simple</i> .....	150
<i>destSelect</i> .....	19	<i>dest_specific</i> .....	139
<i>destSelect</i> .....	27	<i>dest_specific</i> .....	149
<i>destStringVal</i> .....	62	<i>dest_stringType</i> .....	137
<i>destStringVal</i> .....	71	<i>dest_stringType</i> .....	145
<i>destTimeVal</i> .....	62	<i>dest_timeType</i> .....	137
<i>destTimeVal</i> .....	71	<i>dest_timeType</i> .....	145

---

---

<i>dest_upb</i> . . . . .	138	<i>dominates</i> . . . . .	15
<i>dest_upb</i> . . . . .	147	<i>doms_null_lemma1</i> . . . . .	45
<i>dest_variable</i> . . . . .	140	<i>doms_null_lemma2</i> . . . . .	46
<i>dest_variable</i> . . . . .	150	<i>dom_id_ran_id_thm</i> . . . . .	257
<i>DinaryName</i> . . . . .	124	<i>dom_list_rel_thm</i> . . . . .	257
<i>DinaryName</i> . . . . .	125	<i>dom_null_thm</i> . . . . .	285
<i>dinary_columns</i> . . . . .	157	<i>dom_rel_combine_null_⊆_lemma</i> . . . . .	50
<i>dinary_columns</i> . . . . .	187	<i>dom_singleton_thm1</i> . . . . .	283
<i>dinary</i> . . . . .	17	<i>dom_singleton_thm</i> . . . . .	283
<i>dinary</i> . . . . .	21	<i>dom_thm</i> . . . . .	285
<i>DirectoryS</i> . . . . .	18	<i>dom_⋃_ran_lemma</i> . . . . .	45
<i>DirectoryS</i> . . . . .	26	<i>dom_⋃_greater_thm</i> . . . . .	284
<i>Directory</i> . . . . .	20	<i>dom_⋃_thm</i> . . . . .	258
<i>Directory</i> . . . . .	26	<i>Dom</i> . . . . .	5
<i>DirTables</i> . . . . .	131	<i>Dom</i> . . . . .	6
<i>DirTables</i> . . . . .	133	<i>dom</i> . . . . .	136
<i>dir_class_preserved_lemma</i> . . . . .	42	<i>dom</i> . . . . .	143
<i>Dir_class</i> . . . . .	18	<i>dot_dot_eq_thm</i> . . . . .	288
<i>Dir_class</i> . . . . .	26	<i>dot_dot_single_thm</i> . . . . .	287
<i>dir_components</i> . . . . .	40	<i>dot_dot_size_thm</i> . . . . .	257
<i>Dir_exist</i> . . . . .	18	<i>downGrade</i> . . . . .	16
<i>Dir_exist</i> . . . . .	26	<i>downGrade</i> . . . . .	21
<i>Dir_tables</i> . . . . .	18	<i>DownSets</i> . . . . .	267
<i>Dir_tables</i> . . . . .	26	<i>DownSets</i> . . . . .	269
<i>DistinctTuples</i> . . . . .	194	<i>downset_cut_thm</i> . . . . .	270
<i>DistinctTuples</i> . . . . .	203	<i>DownSet</i> . . . . .	267
<i>distinct_length_≤_thm</i> . . . . .	249	<i>DownSet</i> . . . . .	269
<i>distinct_single_thm</i> . . . . .	288	<i>down_sets_cuts_thm</i> . . . . .	269
<i>distinct_size_length_thm</i> . . . . .	250	<i>down_sets_dense_thm</i> . . . . .	270
<i>distinct_thm1</i> . . . . .	251	<i>down_sets_less_thm</i> . . . . .	270
<i>distinct_tuples</i> . . . . .	65	<i>down_⋂_comp_up_thm</i> . . . . .	234
<i>distinct_tuples</i> . . . . .	84	<i>DTR_cols</i> . . . . .	130
<i>distinct_⋀_thm</i> . . . . .	288	<i>DTR_cols</i> . . . . .	132
<i>distinct</i> . . . . .	64	<i>DTR_row_o_HideDerTableRow_lemma</i> . . . . .	209
<i>distinct</i> . . . . .	79	<i>DTR_row</i> . . . . .	130
<i>Distinct</i> . . . . .	274	<i>DTR_row</i> . . . . .	132
<i>Distinct</i> . . . . .	275	<i>DTR_where</i> . . . . .	130
<i>div_def</i> . . . . .	314	<i>DTR_where</i> . . . . .	132
<i>div_mod_thm</i> . . . . .	317	<i>DTS_colSpecs</i> . . . . .	130
<i>div_mod_unique_thm</i> . . . . .	317	<i>DTS_colSpecs</i> . . . . .	132
<i>Div_Z_consistent</i> . . . . .	321	<i>DTS_maxRow</i> . . . . .	130
<i>Div_Z</i> . . . . .	317	<i>DTS_maxRow</i> . . . . .	132
<i>Div_Z</i> . . . . .	318	<i>DTS_name</i> . . . . .	130
<i>Div_Z</i> . . . . .	319	<i>DTS_name</i> . . . . .	132
<i>Div</i> . . . . .	313	<i>DT_rows</i> . . . . .	131
<i>Div</i> . . . . .	314	<i>DT_rows</i> . . . . .	133
<i>Div</i> . . . . .	318	<i>DT_spec_HideDerTable_lemma</i> . . . . .	219
<i>dominates_consistent</i> . . . . .	27	<i>DT_spec</i> . . . . .	131
<i>dominates_lubl_map_hide_lemma</i> . . . . .	210	<i>DT_spec</i> . . . . .	133
<i>dominates_lub_lemma</i> . . . . .	190	<i>dummyVal</i> . . . . .	28
<i>dominates_w</i> . . . . .	63	<i>dummyVal</i> . . . . .	29
<i>dominates_w</i> . . . . .	71	<i>DYADIC</i> . . . . .	12
<i>dominates_w</i> . . . . .	73	<i>dy_arch_thm</i> . . . . .	13
<i>dominates</i> . . . . .	14	<i>dy_balance_thm1</i> . . . . .	13

---

<i>dy_balance_thm2</i> .....	13	<i>EM_SecureE_Lemma1</i> .....	191
<i>dy_exp_clauses</i> .....	13	<i>EM_SecureE_Lemma2</i> .....	191
<i>dy_exp</i> .....	11	<i>EM_SecureE_Lemma3</i> .....	191
<i>dy_exp</i> .....	12	<i>EM_SecureE_thm</i> .....	191
<i>dy_left_dense_thm</i> .....	13	<i>EM_SecureE</i> .....	216
<i>dy_less_antisym_thm</i> .....	12	<i>EM_SecureE</i> .....	217
<i>dy_less_dense_thm</i> .....	13	<i>EM<sub>1</sub></i> .....	131
<i>dy_less_irrefl_thm</i> .....	12	<i>EM<sub>1</sub></i> .....	136
<i>dy_less_order_lemmas_thm</i> .....	297	<i>EM</i> .....	131
<i>dy_less_trans_thm</i> .....	12	<i>EM</i> .....	133
<i>dy_less_trich_thm</i> .....	12	<i>engroup</i> .....	66
<i>dy_less</i> .....	11	<i>engroup</i> .....	86
<i>dy_less</i> .....	12	<i>enseq</i> .....	63
<i>dy_one</i> .....	11	<i>enseq</i> .....	79
<i>dy_one</i> .....	12	<i>enter_scope</i> .....	156
<i>dy_right_dense_thm</i> .....	13	<i>enter_scope</i> .....	176
<i>dy_times_assoc_thm</i> .....	12	<i>enter<sub>corrtable</sub></i> .....	155
<i>dy_times_comm_thm</i> .....	12	<i>enter<sub>corrtable</sub></i> .....	174
<i>dy_times_mono_thm1</i> .....	13	<i>enter<sub>identiferconstantclass</sub></i> .....	155
<i>dy_times_mono_thm2</i> .....	13	<i>enter<sub>identiferconstantclass</sub></i> .....	173
<i>dy_times_mono_thm</i> .....	13	<i>enter<sub>identifier</sub></i> .....	155
<i>dy_times_mono_↔_thm</i> .....	13	<i>enter<sub>identifier</sub></i> .....	172
<i>dy_times_order_thm</i> .....	12	<i>enter<sub>parameter</sub></i> .....	155
<i>dy_times_unit_clauses</i> .....	13	<i>enter<sub>parameter</sub></i> .....	174
<i>dy_times_unit_thm</i> .....	13	<i>enter<sub>table</sub></i> .....	156
<i>dy_times</i> .....	11	<i>enter<sub>table</sub></i> .....	175
<i>dy_times</i> .....	12	<i>enumerate_singleton_thm</i> .....	283
<i>Effect</i> .....	20	<i>enumerate_thm</i> .....	257
<i>elems_append_thm</i> .....	290	<i>enumerate_∪_thm</i> .....	258
<i>elems_combine_elems_thm</i> .....	240	<i>Enumerate</i> .....	274
<i>elems_combine_map_thm1</i> .....	240	<i>Enumerate</i> .....	275
<i>elems_combine_map_thm</i> .....	240	<i>Enum</i> .....	142
<i>elems_combine_swap_thm</i> .....	240	<i>Env</i> .....	70
<i>elems_combine_thm</i> .....	240	<i>Env</i> .....	72
<i>elems_map_thm</i> .....	252	<i>Equal</i> .....	63
<i>elems_map_thm</i> .....	291	<i>Equal</i> .....	73
<i>elems_set_comp_thm</i> .....	258	<i>Equivalence</i> .....	232
<i>elems_thm1</i> .....	251	<i>equiv_anti_mono_lift_rel_thm</i> .....	234
<i>elems_thm2</i> .....	251	<i>eq_rewrite_thm</i> .....	265
<i>elems_thm3</i> .....	251	<i>Errors</i> .....	20
<i>elems_∧_thm</i> .....	288	<i>error</i> .....	16
<i>elems_∨_thm</i> .....	249	<i>Error</i> .....	20
<i>Elms</i> .....	274	<i>error</i> .....	20
<i>Elms</i> .....	275	<i>EvalProjectData</i> .....	194
<i>elem</i> .....	66	<i>EvalProjectData</i> .....	201
<i>elem</i> .....	88	<i>EvalProject</i> .....	194
<i>emptyEnv</i> .....	64	<i>EvalProject</i> .....	201
<i>emptyEnv</i> .....	82	<i>evaluate</i> .....	65
<i>emptyTuple</i> .....	64	<i>evaluate</i> .....	84
<i>emptyTuple</i> .....	82	<i>Evaluate</i> .....	195
<i>emptyUnionList</i> .....	153	<i>Evaluate</i> .....	204
<i>emptyUnionList</i> .....	160	<i>ExceptionData</i> .....	63
<i>empty_finite_thm</i> .....	249	<i>ExceptionData</i> .....	73
<i>Empty</i> .....	277	<i>ExceptionValue</i> .....	137

<i>ExceptionValue</i> . . . . .	144	<i>fillTab</i> . . . . .	64
<i>ExceptionVal</i> . . . . .	17	<i>fillTab</i> . . . . .	80
<i>ExceptionVal</i> . . . . .	22	<i>fill_cols</i> . . . . .	121
<i>Exception</i> . . . . .	137	<i>fill_cols</i> . . . . .	122
<i>Exception</i> . . . . .	144	<i>fill_row</i> . . . . .	121
<i>ExistsTuples_OK_c-lemma</i> . . . . .	215	<i>fill_row</i> . . . . .	122
<i>ExistsTuples_OK_d-lemma</i> . . . . .	213	<i>fill_table</i> . . . . .	121
<i>ExistsTuples</i> . . . . .	193	<i>fill_table</i> . . . . .	122
<i>ExistsTuples</i> . . . . .	199	<i>FilterObj_consistent</i> . . . . .	240
<i>ExpClass</i> . . . . .	142	<i>FilterObj</i> . . . . .	235
<i>ExpType</i> . . . . .	142	<i>FilterObj</i> . . . . .	238
<i>extract_∧_single_ax</i> . . . . .	282	<i>filterRow</i> . . . . .	28
<i>extract_∧_single_lemma</i> . . . . .	56	<i>filterRow</i> . . . . .	29
<i>extract_parameter</i> . . . . .	155	<i>filter_cols</i> . . . . .	121
<i>extract_parameter</i> . . . . .	174	<i>filter_cols</i> . . . . .	123
<i>extract</i> . . . . .	64	<i>FILTER_PARS</i> . . . . .	121
<i>extract</i> . . . . .	82	<i>filter_select</i> . . . . .	121
<i>Extract</i> . . . . .	274	<i>filter_select</i> . . . . .	123
<i>Extract</i> . . . . .	275	<i>filter_table</i> . . . . .	121
<i>ext_thm</i> . . . . .	266	<i>filter_table</i> . . . . .	123
<i>E_group</i> . . . . .	62	<i>FILTER_TYPE</i> . . . . .	118
<i>E_group</i> . . . . .	72	<i>filter_where_row</i> . . . . .	121
<i>E_row</i> . . . . .	62	<i>filter_where_row</i> . . . . .	123
<i>E_row</i> . . . . .	72	<i>find_column</i> . . . . .	154
<i>factor0</i> . . . . .	242	<i>find_column</i> . . . . .	160
<i>factor0</i> . . . . .	243	<i>find_ident</i> . . . . .	154
<i>factor1</i> . . . . .	242	<i>find_ident</i> . . . . .	161
<i>factor1</i> . . . . .	243	<i>find</i> . . . . .	64
<i>factor2</i> . . . . .	242	<i>find</i> . . . . .	79
<i>factor2</i> . . . . .	243	<i>FinitaryRecType</i> . . . . .	289
<i>factor3</i> . . . . .	242	<i>finite_distinct_elems_thm</i> . . . . .	249
<i>factor3</i> . . . . .	243	<i>finite_function_thm</i> . . . . .	252
<i>FactoredMachine</i> . . . . .	243	<i>finite_induction_thm</i> . . . . .	249
<i>FactoredMachine</i> . . . . .	244	<i>finite_max_thm</i> . . . . .	252
<i>Factorisation</i> . . . . .	243	<i>finite_size_thm</i> . . . . .	252
<i>factors</i> . . . . .	242	<i>finite_⊆_well_founded_thm</i> . . . . .	250
<i>factors</i> . . . . .	244	<i>Finite</i> . . . . .	248
<i>factor_level</i> . . . . .	242	<i>fin_dom_thm</i> . . . . .	283
<i>factor_level</i> . . . . .	243	<i>fin_enumerate_ax</i> . . . . .	282
<i>factor_out</i> . . . . .	242	<i>fin_id_thm</i> . . . . .	283
<i>factor_out</i> . . . . .	244	<i>fin_inv_rel_thm</i> . . . . .	284
<i>Factor</i> . . . . .	243	<i>fin_lemma1</i> . . . . .	47
<i>false</i> . . . . .	69	<i>fin_lemma2</i> . . . . .	48
<i>fef036_main_lemma</i> . . . . .	231	<i>fin_list_rel_thm</i> . . . . .	286
<i>FE_SWORD_SYSTEM_secure</i> . . . . .	215	<i>fin_list_rel_▷_thm</i> . . . . .	286
<i>FE_SWORD_SYSTEM_secure</i> . . . . .	216	<i>fin_rec_type_induction_thm1</i> . . . . .	291
<i>FE_SWORD_SYSTEM</i> . . . . .	215	<i>fin_rec_type_induction_thm</i> . . . . .	291
<i>FE_SWORD_SYSTEM</i> . . . . .	216	<i>fin_rec_type_prim_rec_exists_thm</i> . . . . .	291
<i>FE_SWORD</i> . . . . .	117	<i>fin_rec_type_prim_rec_lemma1</i> . . . . .	291
<i>FE_SWORD</i> . . . . .	119	<i>fin_rec_type_prim_rec_thm</i> . . . . .	292
<i>fieldClassOutOfRange</i> . . . . .	16	<i>fin_rec_type_prim_rec_unique_thm</i> . . . . .	291
<i>fieldClassOutOfRange</i> . . . . .	21	<i>fin_rel_combine_thm</i> . . . . .	284
<i>fillCol</i> . . . . .	64	<i>fin_set_induction_thm</i> . . . . .	251
<i>fillCol</i> . . . . .	80	<i>fin_set_thm1</i> . . . . .	251

---

<i>fin_set_thm2</i> .....	251	<i>f_def</i> .....	262
<i>fin_set_thm3</i> .....	251	<i>F_exponent</i> .....	16
<i>fin_set_thm4</i> .....	251	<i>F_exponent</i> .....	20
<i>fin_set_thm5</i> .....	283	<i>F_mantissa</i> .....	16
<i>fin_squash_eq_ax</i> .....	282	<i>F_mantissa</i> .....	20
<i>fin_squash_thm</i> .....	284	<i>f_thm</i> .....	265
<i>fin_≤_thm</i> .....	283	<i>f<sub>1</sub></i> .....	124
<i>fin_▷_thm</i> .....	284	<i>f<sub>1</sub></i> .....	126
<i>Fixed</i> .....	142	<i>f<sub>2</sub></i> .....	124
<i>flat_append_thm</i> .....	290	<i>f<sub>2</sub></i> .....	127
<i>flat_empty_thm</i> .....	290	<i>f<sub>3</sub></i> .....	124
<i>Flat</i> .....	274	<i>f<sub>3</sub></i> .....	128
<i>Flat</i> .....	275	<i>F</i> .....	262
<i>floatingType</i> .....	67	<i>getColPosn</i> .....	64
<i>floatingType</i> .....	88	<i>getColPosn</i> .....	80
<i>floating</i> .....	19	<i>getData</i> .....	64
<i>FloatVal</i> .....	17	<i>getData</i> .....	80
<i>FloatVal</i> .....	22	<i>getDir</i> .....	64
<i>Float</i> .....	19	<i>getDir</i> .....	80
<i>Float</i> .....	20	<i>getTable</i> .....	28
<i>Float</i> .....	294	<i>getTable</i> .....	31
<i>Float</i> .....	297	<i>getTab</i> .....	64
<i>FlowSecureMachine</i> .....	235	<i>getTab</i> .....	80
<i>FlowSecureMachine</i> .....	238	<i>get<sub>table</sub>info</i> .....	156
<i>fold_def</i> .....	261	<i>get<sub>table</sub>info</i> .....	176
<i>fold</i> .....	136	<i>GiveData</i> .....	131
<i>fold</i> .....	143	<i>GiveData</i> .....	134
<i>Fold</i> .....	258	<i>giveError_consistent</i> .....	43
<i>Fold</i> .....	261	<i>giveError_consistent</i> .....	191
<i>From_name</i> .....	67	<i>giveError_eq_thm</i> .....	43
<i>From_name</i> .....	90	<i>giveError</i> .....	19
<i>from_spec<sub>enter</sub></i> .....	158	<i>giveError</i> .....	26
<i>from_spec<sub>enter</sub></i> .....	189	<i>giveVal_consistent</i> .....	43
<i>from_spec<sub>info</sub></i> .....	158	<i>giveVal_consistent</i> .....	191
<i>from_spec<sub>info</sub></i> .....	188	<i>giveVal_eq_thm</i> .....	43
<i>From_spec</i> .....	67	<i>giveVal</i> .....	19
<i>From_spec</i> .....	70	<i>giveVal</i> .....	26
<i>From_spec</i> .....	90	<i>gbl</i> .....	63
<i>from</i> .....	65	<i>gbl</i> .....	73
<i>from</i> .....	84	<i>gbl_consistent</i> .....	27
<i>Front</i> .....	274	<i>gbl</i> .....	14
<i>Front</i> .....	275	<i>gbl</i> .....	15
<i>fst_def</i> .....	273	<i>graph_at_thm</i> .....	254
<i>fst_snd_split_thm</i> .....	210	<i>Graph</i> .....	5
<i>Fst</i> .....	272	<i>Graph</i> .....	6
<i>Fst</i> .....	273	<i>greater_def</i> .....	314
<i>Functional</i> .....	6	<i>GroupA_cols_lemma1</i> .....	225
<i>Functional</i> .....	7	<i>GroupA_cols_lemma2</i> .....	225
<i>fun_destError_thm</i> .....	47	<i>GroupA_lemma</i> .....	225
<i>fun_if_thm</i> .....	209	<i>GroupA_OK<sub>c</sub>_lemma</i> .....	227
<i>fun_nth_map_lemma1</i> .....	219	<i>GroupA_OK<sub>d</sub>_lemma</i> .....	226
<i>fun_nth_map_lemma</i> .....	219	<i>GroupA</i> .....	218
<i>fun_rel_thm</i> .....	266	<i>GroupB_OK<sub>c</sub>_lemma</i> .....	227
<i>fun_updateRow_thm</i> .....	47	<i>GroupB_OK<sub>d</sub>_lemma</i> .....	226

---

---

<i>GroupB</i> . . . . .	218	<i>identical_obj_refl_thm</i> . . . . .	241
<i>GroupedResult</i> . . . . .	70	<i>identical_obj_rft_thm</i> . . . . .	241
<i>GroupedResult</i> . . . . .	72	<i>identical_obj_sym_thm</i> . . . . .	241
<i>Group_lemma1</i> . . . . .	223	<i>identical_obj_trans_thm</i> . . . . .	241
<i>Group_lemma2</i> . . . . .	223	<i>identity_witness_label_secure_thm</i> . . . . .	247
<i>Group_OK<sub>c</sub>_lemma</i> . . . . .	227	<i>identity_witness_thm</i> . . . . .	247
<i>Group_OK<sub>d</sub>_lemma</i> . . . . .	227	<i>identity_witness</i> . . . . .	242
<i>Group</i> . . . . .	194	<i>identity_witness</i> . . . . .	246
<i>Group</i> . . . . .	202	<i>Ide</i> . . . . .	16
<i>G_group</i> . . . . .	62	<i>Ide</i> . . . . .	20
<i>G_group</i> . . . . .	72	<i>id_cName</i> . . . . .	141
<i>G_res</i> . . . . .	62	<i>id_cName</i> . . . . .	152
<i>G_res</i> . . . . .	72	<i>id_dom_null_thm</i> . . . . .	285
<i>HasSupremum</i> . . . . .	267	<i>id_identName</i> . . . . .	141
<i>HasSupremum</i> . . . . .	268	<i>id_identName</i> . . . . .	152
<i>hd_def</i> . . . . .	260	<i>id_info</i> . . . . .	141
<i>Hd</i> . . . . .	258	<i>id_info</i> . . . . .	152
<i>Hd</i> . . . . .	260	<i>id_lub<sub>i d</sub></i> . . . . .	141
<i>Head</i> . . . . .	274	<i>id_lub<sub>i d</sub></i> . . . . .	152
<i>Hidden</i> . . . . .	28	<i>id_singleton_thm</i> . . . . .	283
<i>Hidden</i> . . . . .	29	<i>id_vName</i> . . . . .	141
<i>HideDerTableData_lemma1</i> . . . . .	223	<i>id_vName</i> . . . . .	152
<i>HideDerTableData_lemma</i> . . . . .	220	<i>id_∪_thm</i> . . . . .	283
<i>HideDerTableData</i> . . . . .	131	<i>Id</i> . . . . .	5
<i>HideDerTableData</i> . . . . .	135	<i>Id</i> . . . . .	6
<i>HideDerTableRow_Length_lemma</i> . . . . .	209	<i>if_rewrite_thm</i> . . . . .	266
<i>HideDerTableRow</i> . . . . .	131	<i>if_thm</i> . . . . .	265
<i>HideDerTableRow</i> . . . . .	135	<i>image_single_thm</i> . . . . .	287
<i>HideDerTable_flat_lemma</i> . . . . .	220	<i>image_∪_thm</i> . . . . .	286
<i>HideDerTable</i> . . . . .	131	<i>Image</i> . . . . .	5
<i>HideDerTable</i> . . . . .	135	<i>Image</i> . . . . .	6
<i>hideR_hide_lemma</i> . . . . .	40	<i>Image</i> . . . . .	7
<i>hideR_lemma</i> . . . . .	42	<i>Independent</i> . . . . .	232
<i>hideR</i> . . . . .	28	<i>IndexedEquiv</i> . . . . .	232
<i>hideR</i> . . . . .	30	<i>induced_order_antisym_thm</i> . . . . .	271
<i>hide_eq_lemma</i> . . . . .	44	<i>induced_order_complete_thm</i> . . . . .	271
<i>hide</i> . . . . .	28	<i>induced_order_dense_thm</i> . . . . .	271
<i>hide</i> . . . . .	30	<i>induced_order_irrefl_thm</i> . . . . .	271
<i>Hide</i> . . . . .	35	<i>induced_order_trans_thm</i> . . . . .	271
<i>H</i> . . . . .	218	<i>induced_order_trich_thm</i> . . . . .	271
<i>H</i> . . . . .	219	<i>induced_order_unbounded_above_thm</i> . . . . .	271
<i>IdeL</i> . . . . .	19	<i>induced_order_unbounded_below_thm</i> . . . . .	272
<i>IdeL</i> . . . . .	26	<i>induced_strict_linear_order_thm</i> . . . . .	271
<i>IdentDetail</i> . . . . .	142	<i>induced_strict_partial_order_thm</i> . . . . .	271
<i>IdentDetail</i> . . . . .	152	<i>induction_thm</i> . . . . .	314
<i>identicalObjs</i> . . . . .	235	<i>IND</i> . . . . .	263
<i>identicalObjs</i> . . . . .	237	<i>infinity_axiom</i> . . . . .	256
<i>identicalObj_consistent</i> . . . . .	239	<i>Influenced</i> . . . . .	232
<i>identicalObj</i> . . . . .	235	<i>Influenced</i> . . . . .	233
<i>identicalObj</i> . . . . .	236	<i>init_trans_state</i> . . . . .	154
<i>identical_obj_equiv_thm</i> . . . . .	241	<i>init_trans_state</i> . . . . .	160
<i>identical_obj_filter_obj_thm1</i> . . . . .	240	<i>Init</i> . . . . .	235
<i>identical_obj_filter_obj_thm2</i> . . . . .	240	<i>Init</i> . . . . .	237
<i>identical_obj_filter_obj_thm</i> . . . . .	240	<i>Injective</i> . . . . .	6

---

---

<i>Injective</i> .....	7	<i>InvRel</i> .....	6
<i>InjLists</i> .....	274	<i>inv_rel_singleton_thm</i> .....	284
<i>InjLists</i> .....	275	<i>inv_rel_thm</i> .....	8
<i>InL</i> .....	280	<i>inv_rel_∪_thm</i> .....	284
<i>InL</i> .....	281	<i>inv_rel_∈_arrow_thm</i> .....	254
<i>innermost</i> .....	154	<i>inv_rel_∅_null_thm</i> .....	285
<i>innermost</i> .....	162	<i>in_new_scope</i> .....	156
<i>inRange</i> .....	66	<i>in_new_scope</i> .....	176
<i>inRange</i> .....	88	<i>In</i> .....	274
<i>InR</i> .....	280	<i>In</i> .....	275
<i>InR</i> .....	281	<i>Irrefl</i> .....	267
<i>InsertEffect</i> .....	19	<i>Irrefl</i> .....	268
<i>InsertEffect</i> .....	26	<i>isBoolVal</i> .....	62
<i>insertQuery_lemma</i> .....	53	<i>isBoolVal</i> .....	72
<i>insertQuery</i> .....	29	<i>IsCharRep</i> .....	10
<i>insertQuery</i> .....	31	<i>isClassVal</i> .....	62
<i>insertRows_lemma</i> .....	52	<i>isClassVal</i> .....	72
<i>insert_def</i> .....	277	<i>isClass</i> .....	17
<i>Insert_list</i> .....	69	<i>isClass</i> .....	23
<i>Insert_list</i> .....	71	<i>isCleared</i> .....	64
<i>Insert_list</i> .....	103	<i>isCleared</i> .....	81
<i>insert_tuples</i> .....	65	<i>isCodeVal</i> .....	62
<i>insert_tuples</i> .....	84	<i>isCodeVal</i> .....	72
<i>Insert</i> .....	20	<i>isData</i> .....	17
<i>insert</i> .....	66	<i>isData</i> .....	23
<i>insert</i> .....	83	<i>isDelete</i> .....	19
<i>Insert</i> .....	276	<i>isDelete</i> .....	27
<i>Insert</i> .....	277	<i>isError_consistent</i> .....	43
<i>integerType</i> .....	67	<i>isError_consistent</i> .....	191
<i>integerType</i> .....	88	<i>isError_updateField_lemma</i> .....	44
<i>integer</i> .....	19	<i>isError_updateRow_lemma</i> .....	45
<i>Integer</i> .....	142	<i>isError_⇔_updateRow_lemma</i> .....	45
<i>internalError</i> .....	153	<i>isError</i> .....	19
<i>internalError</i> .....	159	<i>isError</i> .....	26
<i>InternalExpClass</i> .....	142	<i>isFloatVal</i> .....	62
<i>internal_value_class</i> .....	158	<i>isFloatVal</i> .....	72
<i>internal_value_class</i> .....	188	<i>isInsert</i> .....	19
<i>intervalType</i> .....	67	<i>isInsert</i> .....	27
<i>intervalType</i> .....	88	<i>isIntervalVal</i> .....	62
<i>IntervalVal</i> .....	17	<i>isIntervalVal</i> .....	72
<i>IntervalVal</i> .....	22	<i>isIntVal</i> .....	62
<i>Interval</i> .....	19	<i>isIntVal</i> .....	72
<i>interval</i> .....	19	<i>isItem</i> .....	17
<i>intPlus</i> .....	63	<i>isItem</i> .....	23
<i>intPlus</i> .....	71	<i>IsListRep</i> .....	258
<i>intPlus</i> .....	75	<i>IsListRep</i> .....	259
<i>IntVal</i> .....	17	<i>IsL</i> .....	280
<i>IntVal</i> .....	22	<i>IsL</i> .....	281
<i>Int</i> .....	19	<i>isNotCleared</i> .....	64
<i>Int</i> .....	20	<i>isNotCleared</i> .....	81
<i>inUpdates_lemma</i> .....	59	<i>isNullItem</i> .....	62
<i>invert</i> .....	136	<i>isNullItem</i> .....	71
<i>invert</i> .....	143	<i>IsOneRep</i> .....	266
<i>InvRel</i> .....	5	<i>IsOneRep</i> .....	267

---

---

<i>IsPairRep</i> .....	272	<i>ItemUpdate</i> .....	17
<i>IsPairRep</i> .....	273	<i>ItemUpdate</i> .....	23
<i>IsR</i> .....	280	<i>item_sterling</i> .....	113
<i>IsR</i> .....	281	<i>item_sterling</i> .....	114
<i>isSelect</i> .....	19	<i>Item</i> .....	20
<i>isSelect</i> .....	27	<i>iterate_consistent</i> .....	39
<i>IsSetRep</i> .....	276	<i>iterate_witness</i> .....	38
<i>IsSetRep</i> .....	277	<i>iterate</i> .....	35
<i>isState_deleteQuery_lemma</i> .....	112	<i>Iter</i> .....	248
<i>isState_insertQuery_lemma</i> .....	112	<i>Iter</i> .....	249
<i>isState_lemma3</i> .....	44	<i>Itf</i> .....	36
<i>isState_updateQuery_lemma</i> .....	112	<i>I_exponent</i> .....	16
<i>isState_updateRows_lemma</i> .....	112	<i>I_exponent</i> .....	20
<i>isState_updateRow_lemma</i> .....	112	<i>i_local_thm</i> .....	292
<i>isState<sub>t</sub></i> .....	113	<i>I_mantissa</i> .....	16
<i>isState<sub>t</sub></i> .....	115	<i>I_mantissa</i> .....	20
<i>isState</i> .....	19	<i>jay</i> .....	64
<i>isState</i> .....	26	<i>jay</i> .....	83
<i>isstate</i> .....	35	<i>JoinData</i> .....	193
<i>isStringVal</i> .....	62	<i>JoinData</i> .....	200
<i>isStringVal</i> .....	72	<i>JoinedRowExistence_OK<sub>c</sub>_lemma</i> .....	215
<i>IsSumRep</i> .....	280	<i>JoinedRowExistence_OK<sub>d</sub>_lemma</i> .....	214
<i>IsSumRep</i> .....	281	<i>JoinedRowExistence</i> .....	193
<i>istate<sub>f</sub></i> .....	117	<i>JoinedRowExistence</i> .....	200
<i>istate<sub>f</sub></i> .....	119	<i>JoinRows_lemma</i> .....	220
<i>isTimeVal</i> .....	62	<i>JoinRows</i> .....	193
<i>isTimeVal</i> .....	72	<i>JoinRows</i> .....	200
<i>isUpdate</i> .....	19	<i>JoinSpecs</i> .....	193
<i>isUpdate</i> .....	27	<i>JoinSpecs</i> .....	200
<i>isValuedItem</i> .....	62	<i>Join_lemma1</i> .....	220
<i>isValuedItem</i> .....	71	<i>Join_lemma2</i> .....	220
<i>isVal_consistent</i> .....	43	<i>Join_lemma3</i> .....	221
<i>isVal_consistent</i> .....	191	<i>Join_lemma4</i> .....	221
<i>isVal_updateRow_lemma</i> .....	45	<i>Join_lemma5</i> .....	221
<i>isVal</i> .....	19	<i>Join_lemma6</i> .....	222
<i>isVal</i> .....	26	<i>Join_OK<sub>d</sub>_lemma</i> .....	222
<i>is_char_rep_def</i> .....	10	<i>join</i> .....	64
<i>is_list_rep_def</i> .....	259	<i>join</i> .....	83
<i>is_one_rep_def</i> .....	267	<i>Join</i> .....	193
<i>is_pair_rep_def</i> .....	273	<i>Join</i> .....	200
<i>is_select</i> .....	131	<i>k_local_thm</i> .....	292
<i>is_select</i> .....	135	<i>label_secure_to</i> .....	242
<i>is_set_rep_def</i> .....	277	<i>label_secure_to</i> .....	245
<i>is_sum_rep_def</i> .....	281	<i>label_secure</i> .....	242
<i>is_N_rep_def</i> .....	313	<i>label_secure</i> .....	245
<i>Is_N_Rep</i> .....	313	<i>Last</i> .....	274
<i>is_ℝ_rep_consistent_thm</i> .....	297	<i>Last</i> .....	275
<i>Is_ℝ_Rep</i> .....	294	<i>lattice_bottom_consistent</i> .....	27
<i>Is_ℝ_Rep</i> .....	295	<i>lattice_bottom</i> .....	14
<i>is_ℤ_rep_consistent_thm</i> .....	319	<i>lattice_bottom</i> .....	15
<i>Is_ℤ_Rep</i> .....	317	<i>lattice_top_consistent</i> .....	27
<i>Is_ℤ_Rep</i> .....	318	<i>lattice_top</i> .....	14
<i>ItemBool</i> .....	192	<i>lattice_top</i> .....	15
<i>ItemBool</i> .....	195	<i>leaf_is_a_tree_thm</i> .....	293

---

---

<i>lemma1_2_thm</i> . . . . .	38	<i>lift_rel_same_lab_val_down_set_thm</i> . . . . .	234
<i>Lemma1</i> . . . . .	36	<i>lift_rel_same_lab_val_equiv_thm</i> . . . . .	234
<i>Lemma1</i> . . . . .	37	<i>Lift</i> . . . . .	137
<i>lemma2_thm</i> . . . . .	38	<i>Lift</i> . . . . .	144
<i>Lemma2</i> . . . . .	36	<i>ListAnd</i> . . . . .	192
<i>Lemma2</i> . . . . .	37	<i>ListAnd</i> . . . . .	196
<i>Lemma3</i> . . . . .	36	<i>ListNth</i> . . . . .	194
<i>Lemma3</i> . . . . .	37	<i>ListNth</i> . . . . .	202
<i>Lemma3</i> . . . . .	39	<i>ListOk</i> . . . . .	137
<i>Lemma4</i> . . . . .	36	<i>ListOk</i> . . . . .	144
<i>Lemma4</i> . . . . .	37	<i>ListOr</i> . . . . .	192
<i>Lemma4</i> . . . . .	39	<i>ListOr</i> . . . . .	196
<i>Lemma5</i> . . . . .	36	<i>ListRel</i> . . . . .	274
<i>Lemma5</i> . . . . .	37	<i>ListRel</i> . . . . .	275
<i>Lemma5</i> . . . . .	39	<i>Lists<sub>1</sub></i> . . . . .	274
<i>length_0_thm</i> . . . . .	210	<i>Lists<sub>1</sub></i> . . . . .	275
<i>length_0_thm</i> . . . . .	291	<i>Lists</i> . . . . .	274
<i>length_1_thm</i> . . . . .	210	<i>Lists</i> . . . . .	275
<i>length_append_thm</i> . . . . .	290	<i>list_cases_thm</i> . . . . .	261
<i>length_def</i> . . . . .	260	<i>list_clauses</i> . . . . .	261
<i>length_length_flat_thm</i> . . . . .	290	<i>list_def</i> . . . . .	260
<i>length_map_lemma</i> . . . . .	219	<i>list_induction_thm</i> . . . . .	261
<i>length_map_thm</i> . . . . .	252	<i>list_local_thm1</i> . . . . .	292
<i>length_set_def_thm</i> . . . . .	287	<i>list_local_thm</i> . . . . .	292
<i>length_single_thm</i> . . . . .	286	<i>list_prim_rec_thm</i> . . . . .	261
<i>length_thm</i> . . . . .	251	<i>list_rel_cons_thm</i> . . . . .	257
<i>length_∩_thm</i> . . . . .	287	<i>list_rel_fun_thm</i> . . . . .	285
<i>length_∧_one_thm</i> . . . . .	286	<i>list_rel_list_thm</i> . . . . .	287
<i>length_∧_thm</i> . . . . .	257	<i>list_rel_null_thm1</i> . . . . .	288
<i>length_⊥_less_thm</i> . . . . .	249	<i>list_rel_null_thm2</i> . . . . .	288
<i>length_⊥_≤_thm</i> . . . . .	249	<i>list_rel_null_thm</i> . . . . .	285
<i>Length</i> . . . . .	258	<i>list_rel_one_one_thm</i> . . . . .	288
<i>Length</i> . . . . .	260	<i>list_rel_singleton_thm</i> . . . . .	257
<i>less_cases_thm</i> . . . . .	316	<i>list_rel_thm</i> . . . . .	257
<i>less_clauses</i> . . . . .	315	<i>list_rel_∧_eq_thm</i> . . . . .	288
<i>less_def</i> . . . . .	314	<i>list_rel_∧_singleton_thm</i> . . . . .	257
<i>less_irrefl_thm</i> . . . . .	316	<i>list_rel_∧_thm</i> . . . . .	257
<i>less_plus1_thm</i> . . . . .	316	<i>list_rel_∧_▷_thm</i> . . . . .	286
<i>less_trans_thm</i> . . . . .	315	<i>list_rel_⊕_thm</i> . . . . .	287
<i>less_well_order_thm</i> . . . . .	316	<i>list_rel_▷_fun_thm</i> . . . . .	285
<i>let_def</i> . . . . .	264	<i>list_∪</i> . . . . .	63
<i>Let</i> . . . . .	264	<i>list_∪</i> . . . . .	71
<i>levelled_factorisation</i> . . . . .	242	<i>list_∪</i> . . . . .	77
<i>levelled_factorisation</i> . . . . .	244	<i>LIST</i> . . . . .	259
<i>LiftConstant</i> . . . . .	137	<i>LIST</i> . . . . .	260
<i>LiftConstant</i> . . . . .	144	<i>LocalFunctional</i> . . . . .	289
<i>LiftRel</i> . . . . .	232	<i>local_functional_thm</i> . . . . .	292
<i>lift_machine_consistent</i> . . . . .	239	<i>lookup_column_info_look</i> . . . . .	154
<i>lift_machine</i> . . . . .	235	<i>lookup_column_info_look</i> . . . . .	161
<i>lift_machine</i> . . . . .	237	<i>lookup_column_row_class_look</i> . . . . .	155
<i>lift_rel_indexed_equiv_thm</i> . . . . .	234	<i>lookup_column_row_class_look</i> . . . . .	168
<i>lift_rel_same_at_c_below_level_indexed_equiv_thm</i> . . . . .	247	<i>lookup_col_spec_class_look</i> . . . . .	154
<i>lift_rel_same_at_c_below_level_refl_thm</i> . . . . .	247	<i>lookup_col_spec_class_look</i> . . . . .	162
<i>lift_rel_same_ins_same_outs_down_set_thm</i> . . . . .	234	<i>lookup_col_spec_dinary_look</i> . . . . .	154

---

---

<i>lookup_col_spec_dinary_look</i> . . . . .	163	<i>lub_coltype</i> . . . . .	157
<i>lookup_col_spec_sterling_look</i> . . . . .	154	<i>lub_coltype</i> . . . . .	182
<i>lookup_col_spec_sterling_look</i> . . . . .	165	<i>lub_expclass</i> . . . . .	157
<i>lookup_table_detail_look</i> . . . . .	155	<i>lub_expclass</i> . . . . .	181
<i>lookup_table_detail_look</i> . . . . .	171	<i>lub_exp</i> . . . . .	157
<i>lookup_table_row_class_look</i> . . . . .	155	<i>lub_exp</i> . . . . .	182
<i>lookup_table_row_class_look</i> . . . . .	170	<i>lub_sqlcol</i> . . . . .	157
<i>lookup_col_specclass</i> . . . . .	154	<i>lub_sqlcol</i> . . . . .	182
<i>lookup_col_specclass</i> . . . . .	163	<i>lub_sqlname</i> . . . . .	157
<i>lookup_col_specdinary</i> . . . . .	154	<i>lub_sqlname</i> . . . . .	182
<i>lookup_col_specdinary</i> . . . . .	164	<i>lub_tableinfo</i> . . . . .	157
<i>lookup_col_specsterling</i> . . . . .	154	<i>lub_tableinfo</i> . . . . .	182
<i>lookup_col_specsterling</i> . . . . .	166	<i>lub_tsqlclassname</i> . . . . .	157
<i>lookup_columninfo</i> . . . . .	154	<i>lub_tsqlclassname</i> . . . . .	183
<i>lookup_columninfo</i> . . . . .	162	<i>lub_tsqlcol</i> . . . . .	157
<i>lookup_columnrowclass</i> . . . . .	155	<i>lub_tsqlcol</i> . . . . .	183
<i>lookup_columnrowclass</i> . . . . .	169	<i>lub_tsqlname</i> . . . . .	157
<i>lookup_localcolimplementation</i> . . . . .	154	<i>lub_tsqlname</i> . . . . .	183
<i>lookup_localcolimplementation</i> . . . . .	166	<i>lub_type</i> . . . . .	157
<i>lookup_localcolinfo</i> . . . . .	155	<i>lub_type</i> . . . . .	182
<i>lookup_localcolinfo</i> . . . . .	166	<i>lub_worth</i> . . . . .	157
<i>lookup_localcolspecclasses</i> . . . . .	155	<i>lub_worth</i> . . . . .	182
<i>lookup_localcolspecclasses</i> . . . . .	167	<i>lub</i> . . . . .	14
<i>lookup_localcolspecsterlings</i> . . . . .	155	<i>lub</i> . . . . .	15
<i>lookup_localcolspecsterlings</i> . . . . .	167	<i>Machine</i> . . . . .	236
<i>lookup_localrowclasses</i> . . . . .	155	<i>Machine</i> . . . . .	237
<i>lookup_localrowclasses</i> . . . . .	168	<i>machine</i> . . . . .	242
<i>lookup_localtableinfo</i> . . . . .	156	<i>machine</i> . . . . .	244
<i>lookup_localtableinfo</i> . . . . .	176	<i>main_thm1</i> . . . . .	39
<i>lookup_paramdata</i> . . . . .	156	<i>main_thm</i> . . . . .	38
<i>lookup_paramdata</i> . . . . .	176	<i>MakeGroups_lemma1</i> . . . . .	225
<i>lookup_tabledetail</i> . . . . .	155	<i>MakeGroups_lemma2</i> . . . . .	225
<i>lookup_tabledetail</i> . . . . .	171	<i>MakeGroups_lemma3</i> . . . . .	225
<i>lookup_table_rowclass</i> . . . . .	155	<i>MakeGroups</i> . . . . .	194
<i>lookup_table_rowclass</i> . . . . .	171	<i>MakeGroups</i> . . . . .	201
<i>lookup</i> . . . . .	64	<i>make_case</i> . . . . .	157
<i>lookup</i> . . . . .	81	<i>make_case</i> . . . . .	186
<i>lub_lemma</i> . . . . .	210	<i>make_col</i> . . . . .	159
<i>lub</i> . . . . .	63	<i>make_col</i> . . . . .	188
<i>lub</i> . . . . .	73	<i>make_data</i> . . . . .	64
<i>lub_consistent</i> . . . . .	27	<i>make_data</i> . . . . .	80
<i>lub_data</i> . . . . .	63	<i>make_dinary</i> . . . . .	159
<i>lub_data</i> . . . . .	73	<i>make_dinary</i> . . . . .	188
<i>lub_lattice_bottom_thm</i> . . . . .	210	<i>make_sterling</i> . . . . .	159
<i>lub_lemma</i> . . . . .	190	<i>make_sterling</i> . . . . .	188
<i>lub_wdata</i> . . . . .	63	<i>make_sv</i> . . . . .	157
<i>lub_wdata</i> . . . . .	73	<i>make_sv</i> . . . . .	184
<i>lub_wl</i> . . . . .	63	<i>map_cleanRow_lemma1</i> . . . . .	57
<i>lub_wl</i> . . . . .	73	<i>map_cleanRow_lemma2</i> . . . . .	58
<i>lub_w</i> . . . . .	63	<i>map_def</i> . . . . .	261
<i>lub_w</i> . . . . .	71	<i>map_distinct_thm</i> . . . . .	252
<i>lub_w</i> . . . . .	73	<i>map_HideDerTable_map_DT_spec_lemma</i> . . . . .	220
<i>lub_boundinfo</i> . . . . .	157	<i>map_hide_map_hide_lemma</i> . . . . .	210
<i>lub_boundinfo</i> . . . . .	180	<i>map_map_id_thm</i> . . . . .	290

---

---

<i>map_null_thm</i> .....	284	<i>MkDerTable</i> .....	133
<i>map_o_lemma</i> .....	190	<i>MkDirectory</i> .....	18
<i>map_∧_thm1</i> .....	287	<i>MkDirectory</i> .....	26
<i>map_∧_thm</i> .....	285	<i>MkEnv</i> .....	62
<i>Map</i> .....	258	<i>MkEnv</i> .....	72
<i>Map</i> .....	261	<i>MkFactoredMachine</i> .....	242
<i>maxBound</i> .....	154	<i>MkFactoredMachine</i> .....	244
<i>maxBound</i> .....	162	<i>MkFactorisation</i> .....	242
<i>MaximalElements</i> .....	255	<i>MkFactorisation</i> .....	243
<i>Maximal⊆</i> .....	280	<i>MkFloat</i> .....	16
<i>Max_consistent</i> .....	249	<i>MkFloat</i> .....	20
<i>max_∈_thm</i> .....	250	<i>MkGroupedResult</i> .....	62
<i>Max<sub>R</sub>-consistent</i> .....	311	<i>MkGroupedResult</i> .....	72
<i>Max<sub>R</sub></i> .....	294	<i>MkIdentDetail</i> .....	141
<i>Max<sub>R</sub></i> .....	297	<i>MkIdentDetail</i> .....	152
<i>Max</i> .....	248	<i>MkInt</i> .....	16
<i>MaybeResult</i> .....	71	<i>MkInt</i> .....	20
<i>maybe</i> .....	69	<i>MkMachine</i> .....	235
<i>Maybe</i> .....	71	<i>MkMachine</i> .....	237
<i>mayNotBeComplete</i> .....	16	<i>MkNullData<sub>t</sub></i> .....	124
<i>mayNotBeComplete</i> .....	21	<i>MkNullData<sub>t</sub></i> .....	125
<i>minimum_¬_thm</i> .....	316	<i>MkObj_consistent</i> .....	238
<i>minus_clauses</i> .....	317	<i>MkObj</i> .....	235
<i>minus_def</i> .....	314	<i>MkObj</i> .....	236
<i>min_clauses</i> .....	251	<i>MkParamInfo</i> .....	141
<i>Min_consistent</i> .....	249	<i>MkParamInfo</i> .....	152
<i>min_thm</i> .....	251	<i>MkReference</i> .....	18
<i>min_∈_thm</i> .....	250	<i>MkReference</i> .....	24
<i>min_≤_thm</i> .....	250	<i>MkReq</i> .....	235
<i>Min<sub>R</sub>-consistent</i> .....	311	<i>MkReq</i> .....	236
<i>Min<sub>R</sub></i> .....	294	<i>MkRow</i> .....	18
<i>Min<sub>R</sub></i> .....	297	<i>MkRow</i> .....	25
<i>Min</i> .....	248	<i>MkScope</i> .....	141
<i>MkClassVal</i> .....	124	<i>MkScope</i> .....	152
<i>MkClassVal</i> .....	125	<i>MkSsqlCol</i> .....	139
<i>MkColCon</i> .....	18	<i>MkSsqlCol</i> .....	147
<i>MkColCon</i> .....	24	<i>MkST_STACK</i> .....	141
<i>MkColSpec</i> .....	17	<i>MkST_STACK</i> .....	152
<i>MkColSpec</i> .....	23	<i>MkTableDetail</i> .....	141
<i>MkConstraintInfo</i> .....	139	<i>MkTableDetail</i> .....	151
<i>MkConstraintInfo</i> .....	149	<i>MkTableInfo</i> .....	139
<i>MkData<sub>t</sub></i> .....	124	<i>MkTableInfo</i> .....	148
<i>MkData<sub>t</sub></i> .....	125	<i>MkTableSpec</i> .....	18
<i>MkData</i> .....	17	<i>MkTableSpec</i> .....	25
<i>MkData</i> .....	23	<i>mkTf<sub>f</sub></i> .....	117
<i>MkDerColSpec</i> .....	130	<i>mkTf<sub>f</sub></i> .....	118
<i>MkDerColSpec</i> .....	132	<i>MkTf<sub>t</sub></i> .....	113
<i>MkDerTableRow_lemma</i> .....	190	<i>MkTf<sub>t</sub></i> .....	116
<i>MkDerTableRow</i> .....	130	<i>mkTf</i> .....	35
<i>MkDerTableRow</i> .....	132	<i>MkTRANS_STATE</i> .....	153
<i>MkDerTableSpec</i> .....	130	<i>MkTRANS_STATE</i> .....	160
<i>MkDerTableSpec</i> .....	132	<i>MkTree_consistent</i> .....	293
<i>MkDerTable_lemma</i> .....	190	<i>MkTree</i> .....	289
<i>MkDerTable</i> .....	131	<i>MkTree</i> .....	290

---

---

<i>MkTsqlCol</i> . . . . .	139	<i>mk_variable</i> . . . . .	150
<i>MkTsqlCol</i> . . . . .	148	<i>ML_BEHAVIOUR</i> . . . . .	236
<i>MkValuedItem</i> . . . . .	17	<i>ml_secure</i> . . . . .	232
<i>MkValuedItem</i> . . . . .	22	<i>ml_secure</i> . . . . .	233
<i>mk_absolute</i> . . . . .	137	<i>mod_def</i> . . . . .	314
<i>mk_absolute</i> . . . . .	144	<i>mod_less_thm</i> . . . . .	317
<i>mk_and</i> . . . . .	140	<i>Mod<sub>Z</sub>_consistent</i> . . . . .	321
<i>mk_and</i> . . . . .	150	<i>Mod<sub>Z</sub></i> . . . . .	317
<i>mk_anonymous_column</i> . . . . .	139	<i>Mod<sub>Z</sub></i> . . . . .	318
<i>mk_anonymous_column</i> . . . . .	149	<i>Mod<sub>Z</sub></i> . . . . .	319
<i>mk_column</i> . . . . .	140	<i>Mod</i> . . . . .	313
<i>mk_column</i> . . . . .	149	<i>Mod</i> . . . . .	314
<i>mk_constant_class</i> . . . . .	140	<i>Mod</i> . . . . .	318
<i>mk_constant_class</i> . . . . .	149	<i>monoleanType</i> . . . . .	67
<i>mk_constant<sub>ec</sub></i> . . . . .	140	<i>monoleanType</i> . . . . .	88
<i>mk_constant<sub>ec</sub></i> . . . . .	150	<i>monolean</i> . . . . .	19
<i>mk_constant<sub>tc</sub></i> . . . . .	139	<i>MonOp_OK<sub>c</sub>-lemma</i> . . . . .	214
<i>mk_constant<sub>tc</sub></i> . . . . .	148	<i>MonOp_OK<sub>d</sub>-lemma</i> . . . . .	211
<i>mk_constant</i> . . . . .	138	<i>monop_type</i> . . . . .	156
<i>mk_constant</i> . . . . .	147	<i>monop_type</i> . . . . .	177
<i>mk_default</i> . . . . .	137	<i>monop</i> . . . . .	65
<i>mk_default</i> . . . . .	144	<i>monop</i> . . . . .	84
<i>MK_DEST</i> . . . . .	142	<i>MonOp</i> . . . . .	192
<i>mk_enumType</i> . . . . .	137	<i>MonOp</i> . . . . .	195
<i>mk_enumType</i> . . . . .	145	<i>NatItem</i> . . . . .	193
<i>mk_fixedType</i> . . . . .	137	<i>NatItem</i> . . . . .	198
<i>mk_fixedType</i> . . . . .	145	<i>NestClosed</i> . . . . .	280
<i>mk_intervalType</i> . . . . .	137	<i>Nest</i> . . . . .	280
<i>mk_intervalType</i> . . . . .	145	<i>newData</i> . . . . .	63
<i>mk_local_identifier</i> . . . . .	140	<i>newData</i> . . . . .	73
<i>mk_local_identifier</i> . . . . .	149	<i>NextNum</i> . . . . .	124
<i>mk_name<sub>s</sub></i> . . . . .	138	<i>NextNum</i> . . . . .	127
<i>mk_name<sub>s</sub></i> . . . . .	147	<i>Next</i> . . . . .	235
<i>mk_name<sub>tc</sub></i> . . . . .	139	<i>Next</i> . . . . .	237
<i>mk_name<sub>tc</sub></i> . . . . .	148	<i>nil_cons_def</i> . . . . .	260
<i>mk_name<sub>tn</sub></i> . . . . .	140	<i>Nil</i> . . . . .	260
<i>mk_name<sub>tn</sub></i> . . . . .	150	<i>noErrors</i> . . . . .	64
<i>mk_name<sub>t</sub></i> . . . . .	138	<i>noErrors</i> . . . . .	80
<i>mk_name<sub>t</sub></i> . . . . .	147	<i>noNulls</i> . . . . .	16
<i>mk_ors</i> . . . . .	140	<i>noNulls</i> . . . . .	21
<i>mk_ors</i> . . . . .	150	<i>nonUniformValues</i> . . . . .	16
<i>mk_simple</i> . . . . .	140	<i>nonUniformValues</i> . . . . .	21
<i>mk_simple</i> . . . . .	150	<i>nonUniqueValues</i> . . . . .	16
<i>mk_specific</i> . . . . .	139	<i>nonUniqueValues</i> . . . . .	21
<i>mk_specific</i> . . . . .	149	<i>noScope</i> . . . . .	153
<i>mk_stringType</i> . . . . .	137	<i>noScope</i> . . . . .	160
<i>mk_stringType</i> . . . . .	145	<i>noSuchColumn</i> . . . . .	16
<i>mk_timeType</i> . . . . .	137	<i>noSuchColumn</i> . . . . .	21
<i>mk_timeType</i> . . . . .	145	<i>noSuchDirectory</i> . . . . .	16
<i>mk_tree_one_one_thm</i> . . . . .	238	<i>noSuchDirectory</i> . . . . .	21
<i>mk_tree_onto_thm</i> . . . . .	238	<i>noSuchParameter</i> . . . . .	153
<i>mk_upb</i> . . . . .	138	<i>noSuchParameter</i> . . . . .	160
<i>mk_upb</i> . . . . .	147	<i>noSuchTable</i> . . . . .	16
<i>mk_variable</i> . . . . .	140	<i>noSuchTable</i> . . . . .	21

---

---

<i>notCleared</i> .....	16	<i>OkSTP</i> .....	206
<i>notCleared</i> .....	20	<i>OkTableComputation</i> .....	195
<i>notDyadic</i> .....	153	<i>OkTableComputation</i> .....	206
<i>notDyadic</i> .....	160	<i>OkValue</i> .....	137
<i>notMonadic</i> .....	153	<i>OkValue</i> .....	144
<i>notMonadic</i> .....	160	<i>OK_TC<sub>c</sub></i> .....	195
<i>notSetFunction</i> .....	153	<i>OK_TC<sub>c</sub></i> .....	207
<i>notSetFunction</i> .....	160	<i>OK_TC<sub>d</sub>-lemma</i> .....	211
<i>notTriadic</i> .....	153	<i>OK_TC<sub>d</sub></i> .....	195
<i>notTriadic</i> .....	160	<i>OK_TC<sub>d</sub></i> .....	207
<i>notTrigger</i> .....	153	<i>ok_to_proceed</i> .....	117
<i>notTrigger</i> .....	160	<i>ok_to_proceed</i> .....	119
<i>not_lattice_top_thm</i> .....	233	<i>ok_vc_tc_lemmas</i> .....	229
<i>Not</i> .....	63	<i>OK_VC<sub>c</sub></i> .....	195
<i>Not</i> .....	73	<i>OK_VC<sub>c</sub></i> .....	207
<i>Nth_HideDerTableRow_lemma</i> .....	209	<i>OK_VC<sub>d</sub></i> .....	195
<i>nth_length_one_thm</i> .....	287	<i>OK_VC<sub>d</sub></i> .....	207
<i>nth_∧_thm1</i> .....	287	<i>Ok</i> .....	137
<i>nth_∧_thm</i> .....	287	<i>Ok</i> .....	144
<i>Nth</i> .....	274	<i>OneOne</i> .....	262
<i>Nth</i> .....	275	<i>one_col</i> .....	64
<i>NullData</i> .....	63	<i>one_col</i> .....	80
<i>NullData</i> .....	73	<i>one_def1</i> .....	267
<i>NullItemItem</i> .....	17	<i>one_def</i> .....	267
<i>NullItemItem</i> .....	23	<i>one_fns_thm</i> .....	267
<i>NullItem</i> .....	20	<i>one_one_def</i> .....	262
<i>nullUpdate</i> .....	69	<i>one_one_inv_thm</i> .....	283
<i>nullUpdate</i> .....	111	<i>one_one_left_inv_thm</i> .....	291
<i>nullValue</i> .....	16	<i>one_one_onto_inv_thm</i> .....	291
<i>nullValue</i> .....	21	<i>one_one_plus1_thm</i> .....	314
<i>null</i> .....	19	<i>one_one_thm</i> .....	266
<i>Num_to_Int</i> .....	67	<i>one_result</i> .....	64
<i>Num_to_Int</i> .....	91	<i>one_result</i> .....	80
<i>Num</i> .....	16	<i>One</i> .....	266
<i>Num</i> .....	20	<i>ONE</i> .....	267
<i>objectClass_consistent</i> .....	238	<i>One</i> .....	267
<i>objectClass</i> .....	235	<i>onlyInTriggers</i> .....	153
<i>objectClass</i> .....	236	<i>onlyInTriggers</i> .....	160
<i>objectContains_consistent</i> .....	239	<i>onto_def</i> .....	262
<i>objectContains</i> .....	235	<i>onto_right_inv_thm</i> .....	291
<i>objectContains</i> .....	236	<i>Onto</i> .....	262
<i>objectRefers_consistent</i> .....	239	<i>OPT</i> .....	142
<i>objectRefers</i> .....	235	<i>Op</i> .....	71
<i>objectRefers</i> .....	236	<i>Or</i> .....	63
<i>object_clauses</i> .....	239	<i>Or</i> .....	73
<i>obj_induction_thm</i> .....	240	<i>OTHERS</i> .....	137
<i>obj_prim_rec_thm</i> .....	239	<i>OTHERS</i> .....	144
<i>Obj</i> .....	236	<i>OutL</i> .....	280
<i>OBSERVATION</i> .....	232	<i>OutL</i> .....	281
<i>ObservedValue</i> .....	232	<i>outputFilter_secureE</i> .....	190
<i>ObservedValue</i> .....	233	<i>outputFilter</i> .....	121
<i>OkSTP_Secure_E_Lemma</i> .....	216	<i>outputFilter</i> .....	123
<i>OkSTP_Secure_E_Lemma</i> .....	217	<i>Output</i> .....	235
<i>OkSTP</i> .....	195	<i>Output</i> .....	237

---

---

<i>OutR</i> .....	280	<i>processSelect</i> .....	69
<i>OutR</i> .....	281	<i>processSelect</i> .....	110
<i>o_assoc_thm</i> .....	11	<i>processUpdate</i> .....	69
<i>o_def</i> .....	11	<i>processUpdate</i> .....	108
<i>o_fst_local_thm</i> .....	292	<i>Process<sub>t</sub></i> .....	113
<i>o_i_thm</i> .....	11	<i>Process</i> .....	35
<i>o_snd_local_thm</i> .....	292	<i>ProjectData_lemma1</i> .....	222
<i>o</i> .....	6	<i>ProjectData_lemma</i> .....	219
<i>o</i> .....	11	<i>ProjectData_OK<sub>d</sub>_lemma</i> .....	227
<i>pair_clauses</i> .....	273	<i>ProjectData</i> .....	193
<i>pair_def</i> .....	273	<i>ProjectData</i> .....	200
<i>pair_eq_thm1</i> .....	285	<i>ProjectSpec</i> .....	193
<i>pair_eq_thm</i> .....	252	<i>ProjectSpec</i> .....	200
<i>pair_ops_def</i> .....	273	<i>projectTuples</i> .....	66
<i>Pair</i> .....	272	<i>projectTuples</i> .....	85
<i>Pair</i> .....	273	<i>Project</i> .....	194
<i>parameterTable</i> .....	141	<i>Project</i> .....	201
<i>parameterTable</i> .....	152	<i>promote</i> .....	63
<i>ParamInfo</i> .....	142	<i>promote</i> .....	79
<i>ParamInfo</i> .....	152	<i>purge_above_level_consistent</i> .....	246
<i>pigeon_hole_thm</i> .....	250	<i>purge_above_level_thm</i> .....	246
<i>pi_clasf</i> .....	141	<i>purge_above_level</i> .....	242
<i>pi_clasf</i> .....	152	<i>purge_above_level</i> .....	246
<i>pi_name</i> .....	141	<i>PutInGroup</i> .....	194
<i>pi_name</i> .....	152	<i>PutInGroup</i> .....	201
<i>pi_val<sub>p</sub></i> .....	141	<i>query_class</i> .....	153
<i>pi_val<sub>p</sub></i> .....	152	<i>query_class</i> .....	160
<i>plus1_less_thm</i> .....	316	<i>query_constants_class</i> .....	153
<i>plus1_≤_thm</i> .....	316	<i>query_constants_class</i> .....	160
<i>plus_assoc_thm1</i> .....	315	<i>query_type</i> .....	44
<i>plus_assoc_thm</i> .....	314	<i>queryselectquery</i> .....	159
<i>plus_clauses</i> .....	315	<i>queryselectquery</i> .....	188
<i>plus_comm_thm</i> .....	315	<i>Query</i> .....	71
<i>plus_def</i> .....	314	<i>ran_thm</i> .....	286
<i>plus_order_thm</i> .....	315	<i>Ran</i> .....	5
<i>Plus</i> .....	63	<i>Ran</i> .....	6
<i>Plus</i> .....	73	<i>Reference</i> .....	19
<i>pp' ts_def</i> .....	264	<i>Reference</i> .....	24
<i>pp' TS</i> .....	264	<i>ReflexiveIn</i> .....	255
<i>Prefix</i> .....	274	<i>Reflexive</i> .....	5
<i>Prefix</i> .....	275	<i>Reflexive</i> .....	7
<i>priceless</i> .....	17	<i>RelCombine</i> .....	6
<i>priceless</i> .....	21	<i>RelCombine</i> .....	7
<i>prim_rec_thm</i> .....	314	<i>RelList_consistent</i> .....	288
<i>processDelete</i> .....	69	<i>RelList</i> .....	274
<i>processDelete</i> .....	106	<i>RelList</i> .....	275
<i>processInsert</i> .....	69	<i>rel_combine_fun_thm</i> .....	285
<i>processInsert</i> .....	104	<i>rel_combine_null_lemma</i> .....	49
<i>processIntegrity</i> .....	69	<i>rel_combine_null_thm1</i> .....	285
<i>processIntegrity</i> .....	103	<i>rel_combine_null_thm2</i> .....	285
<i>processQuery<sub>t</sub></i> .....	113	<i>rel_combine_null_thm</i> .....	285
<i>processQuery<sub>t</sub></i> .....	115	<i>rel_combine_one_lemma</i> .....	49
<i>processQuery</i> .....	69	<i>rel_combine_singleton_thm</i> .....	284
<i>processQuery</i> .....	111	<i>rel_combine_∪_thm1</i> .....	284

---

---

<i>rel_combine_∪_thm</i> . . . . .	284	<i>revealRow</i> . . . . .	28
<i>rel_ext_clauses</i> . . . . .	284	<i>revealRow</i> . . . . .	31
<i>rel_ext_thm</i> . . . . .	257	<i>rev_def</i> . . . . .	260
<i>rel_list_null_thm</i> . . . . .	288	<i>rev_list_induction_thm</i> . . . . .	257
<i>rel_list_⊕_ax1</i> . . . . .	282	<i>rev_rev_thm</i> . . . . .	257
<i>rel_list_⊕_ax2</i> . . . . .	283	<i>rev_∧_thm</i> . . . . .	257
<i>rel_thm</i> . . . . .	285	<i>Rev</i> . . . . .	258
<i>rel_∈_in_clauses</i> . . . . .	7	<i>Rev</i> . . . . .	260
<i>RemoveDuplicates</i> . . . . .	194	<i>rightNull</i> . . . . .	67
<i>RemoveDuplicates</i> . . . . .	203	<i>rightNull</i> . . . . .	90
<i>remove_constants</i> . . . . .	157	<i>rightType</i> . . . . .	67
<i>remove_constants</i> . . . . .	185	<i>rightType</i> . . . . .	89
<i>remove_nulls</i> . . . . .	157	<i>RiskInputs</i> . . . . .	131
<i>remove_nulls</i> . . . . .	186	<i>RiskInputs</i> . . . . .	135
<i>RepChar</i> . . . . .	10	<i>rowClassTooLow</i> . . . . .	16
<i>replaceData_fun_thm</i> . . . . .	42	<i>rowClassTooLow</i> . . . . .	21
<i>replaceData_lemma</i> . . . . .	40	<i>rowExistColCon</i> . . . . .	124
<i>replaceData_updateField_lemma</i> . . . . .	54	<i>rowExistColCon</i> . . . . .	126
<i>replaceData_updateField_lemma</i> . . . . .	58	<i>rowExistCol</i> . . . . .	124
<i>replaceData</i> . . . . .	28	<i>rowExistCol</i> . . . . .	126
<i>replaceData</i> . . . . .	29	<i>RowExistName</i> . . . . .	124
<i>replaceRows</i> . . . . .	28	<i>RowExistName</i> . . . . .	125
<i>replaceRows</i> . . . . .	31	<i>RowS</i> . . . . .	18
<i>reprState</i> . . . . .	124	<i>RowS</i> . . . . .	25
<i>reprState</i> . . . . .	130	<i>row_components</i> . . . . .	40
<i>Repr_colCon</i> . . . . .	124	<i>Row</i> . . . . .	20
<i>Repr_colCon</i> . . . . .	125	<i>Row</i> . . . . .	25
<i>repr_col</i> . . . . .	156	<i>R_data</i> . . . . .	18
<i>repr_col</i> . . . . .	177	<i>R_data</i> . . . . .	25
<i>repr_data</i> . . . . .	124	<i>R_exist</i> . . . . .	18
<i>repr_data</i> . . . . .	125	<i>R_exist</i> . . . . .	25
<i>repr_dir</i> . . . . .	124	<i>R_group</i> . . . . .	18
<i>repr_dir</i> . . . . .	130	<i>R_group</i> . . . . .	24
<i>repr_row</i> . . . . .	124	<i>R_o_R</i> . . . . .	5
<i>repr_row</i> . . . . .	129	<i>R_o_R</i> . . . . .	6
<i>repr_table</i> . . . . .	124	<i>R_table</i> . . . . .	18
<i>repr_table</i> . . . . .	129	<i>R_table</i> . . . . .	24
<i>repState_consistent</i> . . . . .	40	<i>R_∫-R</i> . . . . .	5
<i>repState_consistent</i> . . . . .	116	<i>R_∫-R</i> . . . . .	6
<i>repState<sub>t</sub>-consistent</i> . . . . .	116	<i>sameFilterInputs</i> . . . . .	235
<i>repState<sub>t</sub></i> . . . . .	113	<i>sameFilterInputs</i> . . . . .	238
<i>repState<sub>t</sub></i> . . . . .	115	<i>SameLabVal</i> . . . . .	232
<i>repState</i> . . . . .	28	<i>SameLabVal</i> . . . . .	233
<i>repState</i> . . . . .	29	<i>sameOutputs</i> . . . . .	235
<i>reqClearance</i> . . . . .	235	<i>sameOutputs</i> . . . . .	237
<i>reqClearance</i> . . . . .	236	<i>sameRequests</i> . . . . .	235
<i>reqSsql</i> . . . . .	235	<i>sameRequests</i> . . . . .	237
<i>reqSsql</i> . . . . .	236	<i>sameRequest</i> . . . . .	235
<i>Req</i> . . . . .	236	<i>sameRequest</i> . . . . .	237
<i>resultBool</i> . . . . .	64	<i>same_at_c_below_level_equiv_thm</i> . . . . .	247
<i>resultBool</i> . . . . .	81	<i>same_at_c_below_level_purge_above_level_thm1</i> . . . . .	247
<i>resultClass</i> . . . . .	64	<i>same_at_c_below_level_purge_above_level_thm2</i> . . . . .	247
<i>resultClass</i> . . . . .	82	<i>same_at_c_below_level_purge_above_level_thm</i> . . . . .	247
<i>RESULT</i> . . . . .	142	<i>same_at_c_below_level</i> . . . . .	242

---

<i>same_at_c_below_level</i> . . . . .	244	<i>select_list<sub>make</sub></i> . . . . .	188
<i>same_ins_anti_mono_thm</i> . . . . .	234	<i>Select_list<sub>p</sub></i> . . . . .	68
<i>same_ins_equiv_thm</i> . . . . .	234	<i>Select_list<sub>p</sub></i> . . . . .	101
<i>same_ins_same_outs_indexed_equiv_thm</i> . . . . .	234	<i>select_list<sub>type</sub></i> . . . . .	158
<i>same_ins</i> . . . . .	14	<i>select_list<sub>type</sub></i> . . . . .	189
<i>same_ins</i> . . . . .	15	<i>Select_list</i> . . . . .	69
<i>same_lab_val_anti_mono_thm</i> . . . . .	234	<i>Select_list</i> . . . . .	71
<i>same_lab_val_equiv_thm</i> . . . . .	234	<i>Select_list</i> . . . . .	101
<i>same_outputs_equiv_thm</i> . . . . .	241	<i>select_values</i> . . . . .	66
<i>same_outputs_indexed_equiv_thm</i> . . . . .	241	<i>select_values</i> . . . . .	83
<i>same_outs_anti_mono_thm</i> . . . . .	234	<i>select_value<sub>class</sub></i> . . . . .	159
<i>same_outs_equiv_thm</i> . . . . .	234	<i>select_value<sub>class</sub></i> . . . . .	188
<i>same_outs</i> . . . . .	14	<i>select_value<sub>data</sub></i> . . . . .	159
<i>same_outs</i> . . . . .	15	<i>select_value<sub>data</sub></i> . . . . .	188
<i>same_requests_equiv_thm</i> . . . . .	241	<i>select_value<sub>info</sub></i> . . . . .	158
<i>same_requests_filter_obj_thm</i> . . . . .	241	<i>select_value<sub>info</sub></i> . . . . .	188
<i>same_requests_indexed_equiv_thm</i> . . . . .	241	<i>select_value<sub>make</sub></i> . . . . .	159
<i>same_to_level</i> . . . . .	242	<i>select_value<sub>make</sub></i> . . . . .	188
<i>same_to_level</i> . . . . .	243	<i>select_value<sub>type</sub></i> . . . . .	158
<i>same</i> . . . . .	67	<i>select_value<sub>type</sub></i> . . . . .	189
<i>same</i> . . . . .	91	<i>Select</i> . . . . .	20
<i>schroeder_bernstein_thm1</i> . . . . .	255	<i>select</i> . . . . .	66
<i>schroeder_bernstein_thm</i> . . . . .	255	<i>select</i> . . . . .	83
<i>Scope</i> . . . . .	142	<i>seqErr</i> . . . . .	66
<i>Scope</i> . . . . .	152	<i>seqErr</i> . . . . .	88
<i>sc_col_class</i> . . . . .	138	<i>seq</i> . . . . .	63
<i>sc_col_class</i> . . . . .	147	<i>seq</i> . . . . .	79
<i>sc_col_exist</i> . . . . .	138	<i>SetComp</i> . . . . .	276
<i>sc_col_exist</i> . . . . .	147	<i>SetComp</i> . . . . .	277
<i>sc_name</i> . . . . .	138	<i>SetFuncAllAnd_OK<sub>c</sub>-lemma</i> . . . . .	215
<i>sc_name</i> . . . . .	147	<i>SetFuncAllAnd_OK<sub>d</sub>-lemma</i> . . . . .	213
<i>sc_type_field</i> . . . . .	138	<i>SetFuncAllAnd</i> . . . . .	193
<i>sc_type_field</i> . . . . .	147	<i>SetFuncAllAnd</i> . . . . .	198
<i>secureHideR_lemma</i> . . . . .	42	<i>SetFuncAllOr_OK<sub>c</sub>-lemma</i> . . . . .	215
<i>secureHideR</i> . . . . .	39	<i>SetFuncAllOr_OK<sub>d</sub>-lemma</i> . . . . .	213
<i>secureHide_lemma</i> . . . . .	42	<i>SetFuncAllOr</i> . . . . .	193
<i>secureHide</i> . . . . .	36	<i>SetFuncAllOr</i> . . . . .	198
<i>secureItf</i> . . . . .	36	<i>SetFuncAll_OK<sub>c</sub>-lemma</i> . . . . .	214
<i>secureItf</i> . . . . .	37	<i>SetFuncAll_OK<sub>d</sub>-lemma</i> . . . . .	213
<i>secureSSQL</i> . . . . .	113	<i>SetFuncAll</i> . . . . .	193
<i>secureStf</i> . . . . .	36	<i>SetFuncAll</i> . . . . .	198
<i>secureStf</i> . . . . .	37	<i>SetFuncDistinct_OK<sub>c</sub>-lemma</i> . . . . .	215
<i>secureUpdate</i> . . . . .	36	<i>SetFuncDistinct_OK<sub>d</sub>-lemma</i> . . . . .	213
<i>secure</i> . . . . .	14	<i>SetFuncDistinct</i> . . . . .	193
<i>secure</i> . . . . .	15	<i>SetFuncDistinct</i> . . . . .	198
<i>SelectEffect</i> . . . . .	19	<i>sets_clauses</i> . . . . .	277
<i>SelectEffect</i> . . . . .	26	<i>sets_ext_clauses</i> . . . . .	279
<i>select_list<sub>class</sub></i> . . . . .	159	<i>set_bottom<sub>d</sub></i> . . . . .	113
<i>select_list<sub>class</sub></i> . . . . .	188	<i>set_bottom<sub>d</sub></i> . . . . .	114
<i>select_list<sub>data</sub></i> . . . . .	159	<i>set_bottom<sub>u</sub></i> . . . . .	113
<i>select_list<sub>data</sub></i> . . . . .	188	<i>set_bottom<sub>u</sub></i> . . . . .	114
<i>select_list<sub>info</sub></i> . . . . .	158	<i>set_class_and_value</i> . . . . .	65
<i>select_list<sub>info</sub></i> . . . . .	188	<i>set_class_and_value</i> . . . . .	84
<i>select_list<sub>make</sub></i> . . . . .	159	<i>set_class</i> . . . . .	65

<i>set_class</i> . . . . .	84	<i>snd_def</i> . . . . .	273
<i>Set_clause</i> . . . . .	69	<i>Snd</i> . . . . .	272
<i>Set_clause</i> . . . . .	70	<i>Snd</i> . . . . .	273
<i>Set_clause</i> . . . . .	102	<i>special_machine</i> . . . . .	242
<i>set_comp_def</i> . . . . .	277	<i>special_machine</i> . . . . .	244
<i>set_def</i> . . . . .	277	<i>splice</i> . . . . .	136
<i>set_dif_clauses</i> . . . . .	278	<i>splice</i> . . . . .	143
<i>set_dif_def</i> . . . . .	277	<i>split3</i> . . . . .	136
<i>set_dif_∪_thm</i> . . . . .	286	<i>split3</i> . . . . .	143
<i>set_func_all</i> . . . . .	65	<i>split4</i> . . . . .	136
<i>set_func_all</i> . . . . .	84	<i>split4</i> . . . . .	143
<i>set_func_type</i> . . . . .	156	<i>split5</i> . . . . .	137
<i>set_func_type</i> . . . . .	177	<i>split5</i> . . . . .	143
<i>set_thm1</i> . . . . .	283	<i>split_def</i> . . . . .	260
<i>set_value</i> . . . . .	65	<i>split_thm</i> . . . . .	210
<i>set_value</i> . . . . .	84	<i>Split</i> . . . . .	258
<i>SET</i> . . . . .	276	<i>Split</i> . . . . .	260
<i>SET</i> . . . . .	277	<i>squash_doms_lemma</i> . . . . .	46
<i>simplest_witness_label_secure_thm</i> . . . . .	247	<i>squash_id_fin_thm</i> . . . . .	283
<i>simplest_witness</i> . . . . .	242	<i>squash_id_thm</i> . . . . .	257
<i>simplest_witness</i> . . . . .	245	<i>squash_null_thm1</i> . . . . .	284
<i>simple_witness_thm</i> . . . . .	247	<i>squash_null_thm</i> . . . . .	284
<i>simplify<sub>ands</sub></i> . . . . .	157	<i>squash_single_thm</i> . . . . .	283
<i>simplify<sub>ands</sub></i> . . . . .	186	<i>squash_thm</i> . . . . .	257
<i>simplify<sub>ors</sub></i> . . . . .	157	<i>squash_∪_thm</i> . . . . .	284
<i>simplify<sub>ors</sub></i> . . . . .	186	<i>squash_∧_thm</i> . . . . .	287
<i>singleton_∪_finite_thm</i> . . . . .	249	<i>Squash</i> . . . . .	274
<i>SingleValue_OK<sub>c</sub>_lemma</i> . . . . .	215	<i>Squash</i> . . . . .	275
<i>SingleValue_OK<sub>d</sub>_lemma</i> . . . . .	213	<i>SSQLam</i> . . . . .	35
<i>SingleValue</i> . . . . .	193	<i>SsqlCol</i> . . . . .	141
<i>SingleValue</i> . . . . .	199	<i>SsqlCol</i> . . . . .	147
<i>size_0_thm</i> . . . . .	250	<i>SsqlName</i> . . . . .	142
<i>size_1_thm</i> . . . . .	250	<i>SSQLtf</i> . . . . .	36
<i>size_empty_thm</i> . . . . .	250	<i>SSQLtf</i> . . . . .	37
<i>size_list_rel_eq_thm</i> . . . . .	288	<i>star1</i> . . . . .	64
<i>size_list_rel_thm</i> . . . . .	286	<i>star1</i> . . . . .	71
<i>size_list_rel_∧_▷_thm</i> . . . . .	286	<i>star1</i> . . . . .	82
<i>size_singleton_thm</i> . . . . .	250	<i>star2</i> . . . . .	64
<i>size_singleton_thm</i> . . . . .	251	<i>star2</i> . . . . .	71
<i>size_singleton_∪_thm</i> . . . . .	250	<i>star2</i> . . . . .	82
<i>size_squash_id_dom_thm</i> . . . . .	286	<i>star3</i> . . . . .	64
<i>size_squash_plus1_thm</i> . . . . .	288	<i>star3</i> . . . . .	71
<i>size_squash_thm</i> . . . . .	286	<i>star3</i> . . . . .	83
<i>size_thm1</i> . . . . .	251	<i>starstar1</i> . . . . .	64
<i>size_thm2</i> . . . . .	251	<i>starstar1</i> . . . . .	71
<i>size_thm3</i> . . . . .	251	<i>starstar1</i> . . . . .	83
<i>size_thm4</i> . . . . .	251	<i>starstar2</i> . . . . .	64
<i>size_thm5</i> . . . . .	251	<i>starstar2</i> . . . . .	71
<i>size_thm6</i> . . . . .	251	<i>starstar2</i> . . . . .	83
<i>size_thm7</i> . . . . .	251	<i>StateDirs</i> . . . . .	131
<i>size_∪_thm</i> . . . . .	250	<i>StateDirs</i> . . . . .	133
<i>size_∧_one_thm</i> . . . . .	288	<i>StateR</i> . . . . .	20
<i>Size</i> . . . . .	248	<i>StateS</i> . . . . .	19
<i>Size</i> . . . . .	249	<i>StateS</i> . . . . .	26

---

<i>StateTables</i> .....	131	<i>sum_clauses</i> .....	281
<i>StateTables</i> .....	133	<i>sum_def</i> .....	281
<i>state_type_def</i> .....	26	<i>sum_fns_thm</i> .....	282
<i>State<sub>tS</sub></i> .....	113	<i>sum_local_thm</i> .....	292
<i>State<sub>tS</sub></i> .....	114	<i>Sup_consistent</i> .....	299
<i>state<sub>t</sub>_type_def</i> .....	115	<i>Sup</i> .....	294
<i>State<sub>t</sub></i> .....	113	<i>Sup</i> .....	296
<i>State</i> .....	20	<i>Surjective</i> .....	6
<i>SterlingName</i> .....	124	<i>Surjective</i> .....	7
<i>SterlingName</i> .....	125	<i>SwordType</i> .....	142
<i>sterling_columns</i> .....	158	<i>SWORD_construction</i> .....	235
<i>sterling_columns</i> .....	187	<i>SWORD_construction</i> .....	238
<i>sterling</i> .....	17	<i>SWORD_ml_secure</i> .....	235
<i>sterling</i> .....	21	<i>SWORD_ml_secure</i> .....	237
<i>Stf</i> .....	35	<i>symbolTable</i> .....	141
<i>STP_secure_E</i> .....	131	<i>symbolTable</i> .....	152
<i>STP_secure_E</i> .....	136	<i>Symmetric</i> .....	5
<i>STP_TYPE</i> .....	118	<i>Symmetric</i> .....	7
<i>STP</i> .....	159	<i>s_identifiers</i> .....	141
<i>STP</i> .....	189	<i>s_identifiers</i> .....	152
<i>StrictLinearOrder</i> .....	267	<i>s_k_thm</i> .....	11
<i>StrictLinearOrder</i> .....	268	<i>s_tables</i> .....	141
<i>StrictPartialOrder</i> .....	267	<i>s_tables</i> .....	152
<i>StrictPartialOrder</i> .....	268	<i>tabExists_cleanTable_lemma1</i> .....	61
<i>StringVal</i> .....	17	<i>tabExists_cleanTable_lemma</i> .....	48
<i>StringVal</i> .....	22	<i>tabExists_lemma1</i> .....	61
<i>STRING</i> .....	10	<i>tabExists_lemma</i> .....	44
<i>String</i> .....	20	<i>tabExists</i> .....	28
<i>ST_STACK</i> .....	142	<i>tabExists</i> .....	31
<i>ST_STACK</i> .....	152	<i>tabFromEffect_consistent</i> .....	43
<i>st_stack</i> .....	153	<i>tabFromEffect</i> .....	19
<i>st_stack</i> .....	160	<i>tabFromEffect</i> .....	27
<i>SubsetClosed</i> .....	280	<i>TableComputationsSecure_thm</i> .....	231
<i>subsys_secureA</i> .....	117	<i>TableComputationsSecure</i> .....	216
<i>subsys_secureA</i> .....	119	<i>TableComputationsSecure</i> .....	217
<i>Subsys_SecureA</i> .....	216	<i>TableComputations_consistent</i> .....	208
<i>subsys_secureB</i> .....	118	<i>TableComputations</i> .....	195
<i>subsys_secureB</i> .....	119	<i>TableComputations</i> .....	204
<i>Subsys_SecureB</i> .....	216	<i>TableContents_OK<sub>c</sub>_lemma</i> .....	228
<i>subsys_secureC</i> .....	118	<i>TableContents_OK<sub>d</sub>_lemma</i> .....	228
<i>subsys_secureC</i> .....	119	<i>TableContents</i> .....	194
<i>Subsys_SecureC</i> .....	216	<i>TableContents</i> .....	202
<i>subsys_secureD</i> .....	118	<i>TableDetail</i> .....	142
<i>subsys_secureD</i> .....	120	<i>TableDetail</i> .....	151
<i>Subsys_SecureD</i> .....	216	<i>TableInfo</i> .....	141
<i>subsys_secureE</i> .....	118	<i>TableInfo</i> .....	148
<i>subsys_secureE</i> .....	120	<i>TableName</i> .....	142
<i>Subsys_SecureE</i> .....	216	<i>TableRow<sub>d</sub></i> .....	131
<i>subsys_secure</i> .....	118	<i>TableRow<sub>d</sub></i> .....	134
<i>subsys_secure</i> .....	120	<i>TableSpecification</i> .....	142
<i>Suc</i> .....	313	<i>TableSpecS</i> .....	18
<i>Suffix</i> .....	274	<i>TableSpecS</i> .....	26
<i>Suffix</i> .....	275	<i>TableSpec<sub>d</sub></i> .....	131
<i>sum_cases_thm</i> .....	282	<i>TableSpec<sub>d</sub></i> .....	133

---

---

<i>TableSpec</i> . . . . .	20	<i>timeNow</i> . . . . .	73
<i>TableSpec</i> . . . . .	25	<i>times_assoc_thm</i> . . . . .	316
<i>tables_fun_thm</i> . . . . .	42	<i>times_clauses</i> . . . . .	317
<i>table_computation_induction_thm</i> . . . . .	229	<i>times_comm_thm</i> . . . . .	316
<i>TABLE_COMP</i> . . . . .	195	<i>times_def</i> . . . . .	314
<i>table_name</i> . . . . .	157	<i>times_plus_distrib_thm</i> . . . . .	316
<i>table_name</i> . . . . .	180	<i>timeType</i> . . . . .	67
<i>Table_spec</i> . . . . .	66	<i>timeType</i> . . . . .	88
<i>Table_spec</i> . . . . .	70	<i>TimeVal</i> . . . . .	17
<i>Table_spec</i> . . . . .	84	<i>TimeVal</i> . . . . .	22
<i>Table<sub>d</sub></i> . . . . .	131	<i>Time</i> . . . . .	19
<i>Table<sub>d</sub></i> . . . . .	134	<i>time</i> . . . . .	19
<i>tab_components</i> . . . . .	40	<i>ti_row_class</i> . . . . .	139
<i>Tab</i> . . . . .	20	<i>ti_row_class</i> . . . . .	148
<i>Tail</i> . . . . .	274	<i>ti_table_class</i> . . . . .	139
<i>take_data</i> . . . . .	64	<i>ti_table_class</i> . . . . .	148
<i>take_data</i> . . . . .	81	<i>ti_table_exist_class</i> . . . . .	139
<i>tc_class_name</i> . . . . .	139	<i>ti_table_exist_class</i> . . . . .	148
<i>tc_class_name</i> . . . . .	148	<i>tl_def</i> . . . . .	260
<i>tc_dinary_name</i> . . . . .	139	<i>Tl</i> . . . . .	258
<i>tc_dinary_name</i> . . . . .	148	<i>Tl</i> . . . . .	260
<i>tc_sterling_name</i> . . . . .	139	<i>tooTall</i> . . . . .	16
<i>tc_sterling_name</i> . . . . .	148	<i>tooTall</i> . . . . .	21
<i>td_columns</i> . . . . .	141	<i>tooWide</i> . . . . .	16
<i>td_columns</i> . . . . .	151	<i>tooWide</i> . . . . .	21
<i>td_constraints</i> . . . . .	141	<i>Total</i> . . . . .	6
<i>td_constraints</i> . . . . .	151	<i>Total</i> . . . . .	7
<i>td_corrName</i> . . . . .	141	<i>transform<sub>selectquery</sub></i> . . . . .	159
<i>td_corrName</i> . . . . .	151	<i>transform<sub>selectquery</sub></i> . . . . .	188
<i>td_genCorr</i> . . . . .	141	<i>Transitive</i> . . . . .	5
<i>td_genCorr</i> . . . . .	151	<i>Transitive</i> . . . . .	7
<i>td_implementation</i> . . . . .	141	<i>TRANS_STATE</i> . . . . .	159
<i>td_implementation</i> . . . . .	151	<i>TRANS_STATE</i> . . . . .	160
<i>td_info</i> . . . . .	141	<i>Trans</i> . . . . .	267
<i>td_info</i> . . . . .	151	<i>Trans</i> . . . . .	268
<i>td_rowClass</i> . . . . .	141	<i>tree_cases_lemma1</i> . . . . .	293
<i>td_rowClass</i> . . . . .	151	<i>tree_cases_lemma</i> . . . . .	293
<i>td_tableName</i> . . . . .	141	<i>tree_def</i> . . . . .	290
<i>td_tableName</i> . . . . .	151	<i>tree_fin_rec_thm</i> . . . . .	294
<i>test</i> . . . . .	66	<i>tree_induction_lemma1</i> . . . . .	293
<i>test</i> . . . . .	87	<i>tree_induction_lemma2</i> . . . . .	293
<i>Text</i> . . . . .	236	<i>tree_induction_lemma</i> . . . . .	293
<i>TF_TYPE</i> . . . . .	118	<i>tree_induction_tac_lemma</i> . . . . .	293
<i>tf<sub>t</sub></i> . . . . .	113	<i>tree_induction_thm</i> . . . . .	294
<i>Theorem1</i> . . . . .	217	<i>tree_local_thm</i> . . . . .	293
<i>Theorem2</i> . . . . .	217	<i>tree_prim_rec_thm</i> . . . . .	294
<i>Theorem3</i> . . . . .	217	<i>TREE</i> . . . . .	289
<i>Theorem4</i> . . . . .	217	<i>Tree</i> . . . . .	289
<i>Theorem5</i> . . . . .	231	<i>TREE</i> . . . . .	290
<i>thm_040_1</i> . . . . .	234	<i>Tree</i> . . . . .	290
<i>thm_040_2</i> . . . . .	234	<i>Trich</i> . . . . .	267
<i>timeFormatToInterval</i> . . . . .	154	<i>Trich</i> . . . . .	268
<i>timeFormatToInterval</i> . . . . .	160	<i>TriOp_OK<sub>c</sub>_lemma</i> . . . . .	214
<i>timeNow</i> . . . . .	62	<i>TriOp_OK<sub>d</sub>_lemma</i> . . . . .	213

---

---

<i>triop_type</i> .....	156	<i>Tuple</i> .....	71
<i>triop_type</i> .....	177	<i>tuple</i> .....	84
<i>TriOp</i> .....	192	<i>TypeDefn</i> .....	262
<i>TriOp</i> .....	197	<i>type_defn_def</i> .....	263
<i>true</i> .....	69	<i>type_lemmas_thm</i> .....	266
<i>Try</i> .....	137	<i>Type</i> .....	20
<i>Try</i> .....	144	<i>t_def</i> .....	262
<i>TsqlClassName</i> .....	142	<i>t_thm</i> .....	265
<i>TsqlCol</i> .....	141	<i>T</i> .....	262
<i>TsqlCol</i> .....	148	<i>UnboundedAbove</i> .....	267
<i>TsqlName</i> .....	142	<i>UnboundedAbove</i> .....	268
<i>TsqlRepr</i> .....	142	<i>UnboundedBelow</i> .....	267
<i>TSQLtf</i> .....	113	<i>UnboundedBelow</i> .....	269
<i>TSQLtf</i> .....	116	<i>uncurry_def</i> .....	273
<i>TS_class</i> .....	18	<i>Uncurry</i> .....	272
<i>TS_class</i> .....	25	<i>Uncurry</i> .....	273
<i>TS_colspecs</i> .....	18	<i>underClassified</i> .....	16
<i>TS_colspecs</i> .....	25	<i>underClassified</i> .....	21
<i>TS_cons</i> .....	18	<i>union</i> .....	65
<i>TS_cons</i> .....	25	<i>union</i> .....	84
<i>TS_maxRow</i> .....	18	<i>unique_name</i> .....	154
<i>TS_maxRow</i> .....	25	<i>unique_name</i> .....	160
<i>TS_rows</i> .....	18	<i>Universe</i> .....	276
<i>TS_rows</i> .....	25	<i>Universe</i> .....	277
<i>Tuple_list_all_tuples</i> .....	68	<i>unparse_thm1</i> .....	293
<i>Tuple_list_all_tuples</i> .....	97	<i>unparse_thm</i> .....	293
<i>Tuple_list_complete<sub>p</sub></i> .....	68	<i>Unparse</i> .....	289
<i>Tuple_list_complete<sub>p</sub></i> .....	95	<i>Unparse</i> .....	290
<i>Tuple_list_complete</i> .....	69	<i>upb_row_class</i> .....	158
<i>Tuple_list_complete</i> .....	101	<i>upb_row_class</i> .....	187
<i>Tuple_list_evaluate</i> .....	68	<i>UpdateEffect</i> .....	19
<i>Tuple_list_evaluate</i> .....	98	<i>UpdateEffect</i> .....	26
<i>tuple_list<sub>class</sub></i> .....	159	<i>updateField</i> .....	29
<i>tuple_list<sub>class</sub></i> .....	188	<i>updateField</i> .....	32
<i>tuple_list<sub>data</sub></i> .....	158	<i>updateQuery_lemma</i> .....	55
<i>tuple_list<sub>data</sub></i> .....	189	<i>updateQuery</i> .....	29
<i>tuple_list<sub>info</sub></i> .....	158	<i>updateQuery</i> .....	33
<i>tuple_list<sub>info</sub></i> .....	188	<i>updateRowList</i> .....	69
<i>tuple_list<sub>makeouter</sub></i> .....	159	<i>updateRowList</i> .....	107
<i>tuple_list<sub>makeouter</sub></i> .....	188	<i>updateRows_lemma</i> .....	54
<i>tuple_list<sub>make</sub></i> .....	158	<i>updateRow_fun_thm</i> .....	50
<i>tuple_list<sub>make</sub></i> .....	188	<i>updateRow_lemma</i> .....	44
<i>tuple_list<sub>maxrowclass</sub></i> .....	158	<i>updateRow</i> .....	29
<i>tuple_list<sub>maxrowclass</sub></i> .....	187	<i>updateRow</i> .....	33
<i>tuple_list<sub>outerinfo</sub></i> .....	159	<i>updateStateR_lemma</i> .....	113
<i>tuple_list<sub>outerinfo</sub></i> .....	188	<i>updateStateR_updateState_lemma</i> .....	112
<i>Tuple_list<sub>p</sub></i> .....	68	<i>updateStateR</i> .....	29
<i>Tuple_list<sub>p</sub></i> .....	99	<i>updateStateR</i> .....	34
<i>tuple_list<sub>type</sub></i> .....	158	<i>updateState<sub>t</sub></i> .....	113
<i>tuple_list<sub>type</sub></i> .....	189	<i>updateState<sub>t</sub></i> .....	116
<i>Tuple_list</i> .....	69	<i>updateState</i> .....	29
<i>Tuple_list</i> .....	71	<i>updateState</i> .....	34
<i>Tuple_list</i> .....	101	<i>update_top_scope</i> .....	155
<i>tuple</i> .....	65	<i>update_top_scope</i> .....	172

---

---

<i>update_type</i> .....	44	<i>Value</i> .....	101
<i>UpDate</i> .....	20	<i>val_or_error_type</i> .....	44
<i>Update</i> .....	20	<i>Val</i> .....	20
<i>update</i> .....	66	<i>View<sub>s</sub></i> .....	131
<i>update</i> .....	83	<i>View<sub>s</sub></i> .....	134
<i>UpperBounds</i> .....	255	<i>View<sub>t_secureE</sub></i> .....	190
<i>UpperBound</i> .....	267	<i>View<sub>t</sub></i> .....	131
<i>UpperBound</i> .....	268	<i>View<sub>t</sub></i> .....	134
<i>upper</i> .....	157	<i>visibleCols</i> .....	28
<i>upper</i> .....	186	<i>visibleCols</i> .....	31
<i>up_down_clauses</i> .....	233	<i>VisibleOutput</i> .....	235
<i>up_down_thm1</i> .....	233	<i>VisibleOutput</i> .....	237
<i>up_down_thm2</i> .....	233	<i>VisibleReq</i> .....	235
<i>up_down_thm3</i> .....	233	<i>VisibleReq</i> .....	237
<i>userClearance</i> .....	62	<i>VI_val</i> .....	17
<i>userClearance</i> .....	73	<i>VI_val</i> .....	22
<i>userDirectory</i> .....	62	<i>VI_worth</i> .....	17
<i>userDirectory</i> .....	73	<i>VI_worth</i> .....	22
<i>userName</i> .....	63	<i>VoidVal</i> .....	17
<i>userName</i> .....	73	<i>VoidVal</i> .....	22
<i>user</i> .....	65	<i>well_factored</i> .....	242
<i>user</i> .....	84	<i>well_factored</i> .....	244
<i>Ustate<sub>t</sub></i> .....	113	<i>WHEN_absolute</i> .....	137
<i>Ustate</i> .....	35	<i>WHEN_absolute</i> .....	142
<i>ValueComputations_consistent</i> .....	208	<i>WHEN_absolute</i> .....	144
<i>ValueComputations</i> .....	195	<i>WHEN_and</i> .....	140
<i>ValueComputations</i> .....	204	<i>WHEN_and</i> .....	142
<i>ValuedItemItem</i> .....	17	<i>WHEN_and</i> .....	150
<i>ValuedItemItem</i> .....	23	<i>WHEN_anonymous_column</i> .....	140
<i>ValuedItem</i> .....	19	<i>WHEN_anonymous_column</i> .....	142
<i>ValuedItem</i> .....	22	<i>WHEN_anonymous_column</i> .....	149
<i>Value_all_binop</i> .....	67	<i>WHEN_anon<sub>s</sub></i> .....	138
<i>Value_all_binop</i> .....	92	<i>WHEN_anon<sub>s</sub></i> .....	147
<i>Value_binop</i> .....	67	<i>WHEN_anon<sub>tC</sub></i> .....	139
<i>Value_binop</i> .....	91	<i>WHEN_anon<sub>tC</sub></i> .....	148
<i>Value_classification</i> .....	67	<i>WHEN_anon<sub>tN</sub></i> .....	141
<i>Value_classification</i> .....	93	<i>WHEN_anon<sub>tN</sub></i> .....	151
<i>VALUE_COMP</i> .....	195	<i>WHEN_anon<sub>t</sub></i> .....	138
<i>Value_monop</i> .....	67	<i>WHEN_anon<sub>t</sub></i> .....	147
<i>Value_monop</i> .....	91	<i>WHEN_anyType</i> .....	138
<i>Value_set_func_all</i> .....	67	<i>WHEN_anyType</i> .....	146
<i>Value_set_func_all</i> .....	92	<i>WHEN_booleanType</i> .....	138
<i>value<sub>class</sub></i> .....	158	<i>WHEN_booleanType</i> .....	146
<i>value<sub>class</sub></i> .....	188	<i>WHEN_classType</i> .....	138
<i>value<sub>data</sub></i> .....	158	<i>WHEN_classType</i> .....	146
<i>value<sub>data</sub></i> .....	188	<i>WHEN_codeType</i> .....	138
<i>value<sub>info</sub></i> .....	158	<i>WHEN_codeType</i> .....	146
<i>value<sub>info</sub></i> .....	188	<i>WHEN_column</i> .....	140
<i>Value<sub>p</sub></i> .....	67	<i>WHEN_column</i> .....	142
<i>Value<sub>p</sub></i> .....	93	<i>WHEN_column</i> .....	150
<i>value<sub>ttype</sub></i> .....	158	<i>WHEN_constant_class</i> .....	140
<i>value<sub>ttype</sub></i> .....	187	<i>WHEN_constant_class</i> .....	142
<i>Value</i> .....	69	<i>WHEN_constant_class</i> .....	150
<i>Value</i> .....	70	<i>WHEN_constant_null</i> .....	140

---

---

<i>WHEN_constant_null</i> . . . . .	150	<i>WHEN_simple</i> . . . . .	142
<i>WHEN_constant<sub>ec</sub></i> . . . . .	140	<i>WHEN_simple</i> . . . . .	150
<i>WHEN_constant<sub>ec</sub></i> . . . . .	142	<i>WHEN_specific</i> . . . . .	140
<i>WHEN_constant<sub>ec</sub></i> . . . . .	150	<i>WHEN_specific</i> . . . . .	143
<i>WHEN_constant<sub>tc</sub></i> . . . . .	139	<i>WHEN_specific</i> . . . . .	149
<i>WHEN_constant<sub>tc</sub></i> . . . . .	142	<i>WHEN_stringType</i> . . . . .	138
<i>WHEN_constant<sub>tc</sub></i> . . . . .	148	<i>WHEN_stringType</i> . . . . .	143
<i>WHEN_constant</i> . . . . .	138	<i>WHEN_stringType</i> . . . . .	146
<i>WHEN_constant</i> . . . . .	142	<i>WHEN_timeType</i> . . . . .	138
<i>WHEN_constant</i> . . . . .	147	<i>WHEN_timeType</i> . . . . .	143
<i>WHEN_CONST</i> . . . . .	142	<i>WHEN_timeType</i> . . . . .	146
<i>WHEN_default</i> . . . . .	137	<i>WHEN_upb</i> . . . . .	138
<i>WHEN_default</i> . . . . .	143	<i>WHEN_upb</i> . . . . .	143
<i>WHEN_default</i> . . . . .	144	<i>WHEN_upb</i> . . . . .	147
<i>WHEN_enumType</i> . . . . .	138	<i>WHEN_variable</i> . . . . .	140
<i>WHEN_enumType</i> . . . . .	143	<i>WHEN_variable</i> . . . . .	143
<i>WHEN_enumType</i> . . . . .	146	<i>WHEN_variable</i> . . . . .	150
<i>WHEN_exception</i> . . . . .	137	<i>WHEN</i> . . . . .	142
<i>WHEN_exception</i> . . . . .	143	<i>Where_dominates_lemma</i> . . . . .	224
<i>WHEN_exception</i> . . . . .	144	<i>Where_lemma</i> . . . . .	224
<i>WHEN_fixedType</i> . . . . .	138	<i>Where_OK<sub>d</sub>_lemma</i> . . . . .	225
<i>WHEN_fixedType</i> . . . . .	143	<i>Where_W_lemma</i> . . . . .	224
<i>WHEN_fixedType</i> . . . . .	146	<i>Where</i> . . . . .	194
<i>WHEN_intervalType</i> . . . . .	138	<i>Where</i> . . . . .	201
<i>WHEN_intervalType</i> . . . . .	143	<i>worthless</i> . . . . .	17
<i>WHEN_intervalType</i> . . . . .	146	<i>worthless</i> . . . . .	21
<i>WHEN_local_identifier</i> . . . . .	140	<i>Worth</i> . . . . .	20
<i>WHEN_local_identifier</i> . . . . .	142	<i>Worth</i> . . . . .	313
<i>WHEN_local_identifier</i> . . . . .	150	<i>wrongScope</i> . . . . .	153
<i>WHEN_monoleanType</i> . . . . .	138	<i>wrongScope</i> . . . . .	160
<i>WHEN_monoleanType</i> . . . . .	146	<i>wrongType</i> . . . . .	16
<i>WHEN_name<sub>s</sub></i> . . . . .	138	<i>wrongType</i> . . . . .	21
<i>WHEN_name<sub>s</sub></i> . . . . .	142	<i>wrongWorth</i> . . . . .	153
<i>WHEN_name<sub>s</sub></i> . . . . .	147	<i>wrongWorth</i> . . . . .	159
<i>WHEN_name<sub>tc</sub></i> . . . . .	139	<i>W_lemma</i> . . . . .	224
<i>WHEN_name<sub>tc</sub></i> . . . . .	142	<i>W</i> . . . . .	218
<i>WHEN_name<sub>tc</sub></i> . . . . .	148	<i>W</i> . . . . .	219
<i>WHEN_name<sub>tn</sub></i> . . . . .	140	<i>x_ml_secure</i> . . . . .	232
<i>WHEN_name<sub>tn</sub></i> . . . . .	142	<i>zero_suc_def</i> . . . . .	314
<i>WHEN_name<sub>tn</sub></i> . . . . .	151	<i>Zero</i> . . . . .	313
<i>WHEN_name<sub>t</sub></i> . . . . .	138	<i>zorn_thm1</i> . . . . .	280
<i>WHEN_name<sub>t</sub></i> . . . . .	142	<i>zorn_thm</i> . . . . .	255
<i>WHEN_name<sub>t</sub></i> . . . . .	147	<i>#</i> . . . . .	248
<i>WHEN_none<sub>t</sub></i> . . . . .	138	<i>#</i> . . . . .	274
<i>WHEN_none<sub>t</sub></i> . . . . .	147	<i>&amp;<sub>1</sub></i> . . . . .	63
<i>WHEN_nullType</i> . . . . .	138	<i>&amp;<sub>1</sub></i> . . . . .	71
<i>WHEN_nullType</i> . . . . .	146	<i>&amp;<sub>1</sub></i> . . . . .	78
<i>WHEN_ok</i> . . . . .	137	<i>&amp;<sub>2</sub></i> . . . . .	63
<i>WHEN_ok</i> . . . . .	142	<i>&amp;<sub>2</sub></i> . . . . .	71
<i>WHEN_ok</i> . . . . .	144	<i>&amp;<sub>2</sub></i> . . . . .	78
<i>WHEN_ors</i> . . . . .	140	<i>&amp;<sub>3</sub></i> . . . . .	63
<i>WHEN_ors</i> . . . . .	142	<i>&amp;<sub>3</sub></i> . . . . .	71
<i>WHEN_ors</i> . . . . .	150	<i>&amp;<sub>3</sub></i> . . . . .	78
<i>WHEN_simple</i> . . . . .	140	<i>&amp;<sub>4</sub></i> . . . . .	63

---

---

$\&_4$ .....	71	$\mathbb{F}_1$ .....	248
$\&_4$ .....	78	$\mathbb{F}$ .....	248
$\&_5$ .....	66	$\mathbb{N}$ _cases_thm .....	316
$\&_5$ .....	71	$\mathbb{N}$ _def .....	313
$\&_5$ .....	85	$\mathbb{N}$ _set_thm1 .....	251
$\&_6$ .....	66	$\mathbb{NR}$ _0_less_thm .....	301
$\&_6$ .....	71	$\mathbb{NR}$ _less_thm .....	302
$\&_6$ .....	86	$\mathbb{NR}$ _one_one_thm .....	301
$\&$ .....	70	$\mathbb{NR}$ _plus_homomorphism_thm1 .....	309
$\&$ .....	71	$\mathbb{NR}$ _plus_homomorphism_thm .....	301
$\beta$ _rewrite_thm .....	266	$\mathbb{NR}$ _times_homomorphism_thm1 .....	309
$\bigcap$ _clauses .....	278	$\mathbb{NR}$ _times_homomorphism_thm .....	308
$\bigcap$ _def .....	277	$\mathbb{NR}$ _≤_thm .....	302
$\bigcap_2$ .....	195	$\mathbb{NR}$ .....	294
$\bigcap_2$ .....	204	$\mathbb{NR}$ .....	296
$\bigcap$ .....	276	$\mathbb{NZ}$ _consistent .....	319
$\bigcap$ .....	277	$\mathbb{NZ}$ _less_thm .....	323
$\bigcup$ _clauses .....	278	$\mathbb{NZ}$ _one_one_thm .....	320
$\bigcup$ _def .....	277	$\mathbb{NZ}$ _plus_homomorphism_thm1 .....	321
$\bigcup$ _finite_thm .....	250	$\mathbb{NZ}$ _plus_homomorphism_thm .....	320
$\bigcup$ .....	276	$\mathbb{NZ}$ _times_homomorphism_thm1 .....	322
$\bigcup$ .....	277	$\mathbb{NZ}$ _times_homomorphism_thm .....	321
$\bigcap$ _clauses .....	278	$\mathbb{NZ}$ _≤_thm .....	321
$\bigcap$ _def .....	277	$\mathbb{NZ}$ .....	317
$\bigcap$ _finite_thm .....	249	$\mathbb{NZ}$ .....	318
$\bigcap$ .....	276	$\mathbb{N}$ .....	248
$\bigcap$ .....	277	$\mathbb{N}$ .....	313
$\bigcup$ _clauses .....	278	$\mathbb{P}$ _clauses .....	278
$\bigcup$ _def .....	277	$\mathbb{P}$ _def .....	277
$\bigcup$ _finite_thm .....	249	$\mathbb{P}$ .....	6
$\bigcup$ _null_thm .....	285	$\mathbb{P}$ .....	276
$\bigcup$ _∩_thm .....	286	$\mathbb{P}$ .....	277
$\bigcup$ _▷_thm .....	258	$\mathbb{R}$ _0_1_thm .....	301
$\bigcup$ .....	276	$\mathbb{R}$ _0_less_0_less_recip_thm .....	309
$\bigcup$ .....	277	$\mathbb{R}$ _0_less_0_less_times_thm .....	308
$\backslash$ .....	276	$\mathbb{R}$ _0_over_thm .....	309
$\backslash$ .....	277	$\mathbb{R}$ _0_≤_0_≤_times_thm .....	309
$\uparrow$ .....	232	$\mathbb{R}$ _0_≤_frac_thm .....	311
$\downarrow$ .....	232	$\mathbb{R}$ _abs_frac_thm .....	311
$\triangleleft$ _null_thm .....	286	$\mathbb{R}$ _abs_minus_thm .....	311
$\triangleleft$ _∪_thm .....	287	$\mathbb{R}$ _add_hom_0_thm .....	303
$\triangleleft$ .....	5	$\mathbb{R}$ _add_hom_extension_thm .....	303
$\triangleleft$ .....	6	$\mathbb{R}$ _add_hom_image_group_thm .....	303
$\emptyset$ _clauses .....	278	$\mathbb{R}$ _add_hom_kernel_group_thm .....	304
$\epsilon$ _axiom .....	256	$\mathbb{R}$ _add_hom_minus_thm .....	303
$\epsilon$ .....	263	$\mathbb{R}$ _complete_thm .....	298
$\eta$ _axiom .....	256	$\mathbb{R}$ _copah_comp_thm .....	305
$\exists$ _def .....	262	$\mathbb{R}$ _copah_dense_thm .....	306
$\exists$ _intro_thm .....	265	$\mathbb{R}$ _copah_double_thm .....	305
$\exists$ _rewrite_thm .....	266	$\mathbb{R}$ _copah_halve_thm .....	305
$\exists_1$ _def .....	264	$\mathbb{R}$ _copah_id_thm .....	305
$\exists_1$ _thm .....	265	$\mathbb{R}$ _copah_sum_thm .....	306
$\exists_1$ .....	264	$\mathbb{R}$ _cross_mult_eq_thm .....	309
$\exists$ .....	262	$\mathbb{R}$ _cross_mult_less_thm .....	309

---

---

$\mathbb{R}$ _def . . . . .	295	$\mathbb{R}$ _opah_strict_thm . . . . .	304
$\mathbb{R}$ _delta_induction_thm . . . . .	302	$\mathbb{R}$ _opah_thm . . . . .	304
$\mathbb{R}$ _eq_recip_thm . . . . .	308	$\mathbb{R}$ _ord_pres_strict_thm . . . . .	303
$\mathbb{R}$ _eq_sup_bc_thm . . . . .	300	$\mathbb{R}$ _over_1_thm . . . . .	309
$\mathbb{R}$ _eq_thm . . . . .	301	$\mathbb{R}$ _over_cancel_eq_thm . . . . .	309
$\mathbb{R}$ _eq_≤_thm . . . . .	298	$\mathbb{R}$ _over_eq_0_thm . . . . .	310
$\mathbb{R}$ _frac_0_thm . . . . .	310	$\mathbb{R}$ _over_over_over_thm . . . . .	310
$\mathbb{R}$ _frac_cancel_eq_thm . . . . .	310	$\mathbb{R}$ _over_plus_over_thm . . . . .	309
$\mathbb{R}$ _frac_cross_mult_eq_thm . . . . .	310	$\mathbb{R}$ _over_recip_thm . . . . .	310
$\mathbb{R}$ _frac_less_frac_thm . . . . .	310	$\mathbb{R}$ _over_times_over_thm . . . . .	310
$\mathbb{R}$ _frac_less_minus_frac_thm . . . . .	311	$\mathbb{R}$ _over_times_recip_thm . . . . .	308
$\mathbb{R}$ _frac_minus_frac_thm1 . . . . .	310	$\mathbb{R}$ _plus_0_thm . . . . .	301
$\mathbb{R}$ _frac_minus_frac_thm . . . . .	310	$\mathbb{R}$ _plus_assoc_thm1 . . . . .	300
$\mathbb{R}$ _frac_plus_frac_thm . . . . .	310	$\mathbb{R}$ _plus_assoc_thm . . . . .	300
$\mathbb{R}$ _frac_recip_thm . . . . .	310	$\mathbb{R}$ _plus_clauses . . . . .	301
$\mathbb{R}$ _frac_times_frac_thm . . . . .	310	$\mathbb{R}$ _plus_comm_thm . . . . .	300
$\mathbb{R}$ _frac_ℕ_thm . . . . .	310	$\mathbb{R}$ _plus_minus_thm . . . . .	301
$\mathbb{R}$ _halve_closed_dense_thm . . . . .	306	$\mathbb{R}$ _plus_mono_thm1 . . . . .	301
$\mathbb{R}$ _less_0_less_thm . . . . .	302	$\mathbb{R}$ _plus_mono_thm2 . . . . .	301
$\mathbb{R}$ _less_antisym_thm . . . . .	298	$\mathbb{R}$ _plus_mono_thm . . . . .	300
$\mathbb{R}$ _less_cases_thm . . . . .	298	$\mathbb{R}$ _plus_order_thm . . . . .	301
$\mathbb{R}$ _less_clauses . . . . .	301	$\mathbb{R}$ _plus_unit_thm . . . . .	300
$\mathbb{R}$ _less_dense_thm . . . . .	298	$\mathbb{R}$ _recip_clauses . . . . .	309
$\mathbb{R}$ _less_irrefl_thm . . . . .	298	$\mathbb{R}$ _semigroup_dense_thm . . . . .	303
$\mathbb{R}$ _less_less_0_thm . . . . .	301	$\mathbb{R}$ _subgroup_dense_thm . . . . .	303
$\mathbb{R}$ _less_strong_dense_thm . . . . .	302	$\mathbb{R}$ _sup_eq_bc_thm . . . . .	300
$\mathbb{R}$ _less_sup_bc_thm . . . . .	299	$\mathbb{R}$ _sup_less_bc_thm . . . . .	300
$\mathbb{R}$ _less_sup_thm . . . . .	299	$\mathbb{R}$ _sup_plus_sup_thm . . . . .	302
$\mathbb{R}$ _less_sup_∈_thm . . . . .	300	$\mathbb{R}$ _sup_plus_thm . . . . .	302
$\mathbb{R}$ _less_trans_thm . . . . .	298	$\mathbb{R}$ _sup_thm . . . . .	299
$\mathbb{R}$ _less_≤_trans_thm . . . . .	298	$\mathbb{R}$ _sup_≤_bc_thm . . . . .	299
$\mathbb{R}$ _less_¬_eq_0_thm . . . . .	309	$\mathbb{R}$ _times_0_thm . . . . .	308
$\mathbb{R}$ _less_¬_eq_thm . . . . .	298	$\mathbb{R}$ _times_1_thm . . . . .	308
$\mathbb{R}$ _max_cons_thm . . . . .	311	$\mathbb{R}$ _times_assoc_thm1 . . . . .	308
$\mathbb{R}$ _max_conv_thm . . . . .	311	$\mathbb{R}$ _times_assoc_thm . . . . .	308
$\mathbb{R}$ _minus_clauses . . . . .	301	$\mathbb{R}$ _times_cancel_thm . . . . .	308
$\mathbb{R}$ _minus_eq_thm . . . . .	301	$\mathbb{R}$ _times_clauses . . . . .	309
$\mathbb{R}$ _minus_frac_less_frac_thm . . . . .	311	$\mathbb{R}$ _times_comm_thm . . . . .	308
$\mathbb{R}$ _minus_recip_thm . . . . .	310	$\mathbb{R}$ _times_eq_0_thm . . . . .	308
$\mathbb{R}$ _min_cons_thm . . . . .	311	$\mathbb{R}$ _times_minus_thm . . . . .	308
$\mathbb{R}$ _min_conv_thm . . . . .	312	$\mathbb{R}$ _times_mono_thm . . . . .	309
$\mathbb{R}$ _monoid_delta_dense_thm . . . . .	303	$\mathbb{R}$ _times_mono_↔_thm . . . . .	309
$\mathbb{R}$ _monoid_dense_thm . . . . .	303	$\mathbb{R}$ _times_order_thm . . . . .	308
$\mathbb{R}$ _opah_complete_thm . . . . .	307	$\mathbb{R}$ _times_plus_distrib_thm . . . . .	308
$\mathbb{R}$ _opah_dense_image_thm . . . . .	304	$\mathbb{R}$ _times_recip_thm . . . . .	308
$\mathbb{R}$ _opah_eq_thm . . . . .	307	$\mathbb{R}$ _times_unit_thm . . . . .	308
$\mathbb{R}$ _opah_extension_thm1 . . . . .	306	$\mathbb{R}$ _unbounded_above_thm . . . . .	298
$\mathbb{R}$ _opah_extension_thm2 . . . . .	307	$\mathbb{R}$ _unbounded_below_thm . . . . .	298
$\mathbb{R}$ _opah_extension_thm . . . . .	307	$\mathbb{R}$ _∈_≤_sup_bc_thm . . . . .	299
$\mathbb{R}$ _opah_inverse_add_hom_thm . . . . .	304	$\mathbb{R}$ _≤_0_≤_thm . . . . .	302
$\mathbb{R}$ _opah_inverse_thm . . . . .	304	$\mathbb{R}$ _≤_antisym_thm . . . . .	298
$\mathbb{R}$ _opah_one_one_thm . . . . .	304	$\mathbb{R}$ _≤_cases_thm . . . . .	298
$\mathbb{R}$ _opah_onto_thm . . . . .	304	$\mathbb{R}$ _≤_clauses . . . . .	302
$\mathbb{R}$ _opah_order_thm . . . . .	307	$\mathbb{R}$ _≤_less_cases_thm . . . . .	298

---

$\mathbb{R}_{<}$ _less_trans_thm	298	$\mathbb{Z}$ _times_comm_thm	321
$\mathbb{R}_{<}$ _refl_thm	298	$\mathbb{Z}$ _times_eq_0_thm	322
$\mathbb{R}_{<}$ _sup_bc_thm	299	$\mathbb{Z}$ _times_minus_thm	321
$\mathbb{R}_{<}$ _sup_thm	299	$\mathbb{Z}$ _times_order_thm	322
$\mathbb{R}_{<}$ _trans_thm	298	$\mathbb{Z}$ _times_plus_distrib_thm	322
$\mathbb{R}_{<}$ _≤_0_thm	302	$\mathbb{Z}$ _N_abs_minus_thm	324
$\mathbb{R}_{<}$ _¬_less_thm	298	$\mathbb{Z}$ _N_abs_thm	323
$\mathbb{R}_{<}$ _¬_recip_0_thm	309	$\mathbb{Z}$ _N_cases_thm	322
$\mathbb{R}_{<}$ _¬_≤_less_thm	298	$\mathbb{Z}$ _N_induction_thm	321
$\mathbb{R}_{<}$ _⊆_sup_thm	299	$\mathbb{Z}$ _N_plus1_thm	322
$\mathbb{R}$	295	$\mathbb{Z}$ _N_plus_thm	322
$\mathbb{Z}$ _abs_eq_0_thm	324	$\mathbb{Z}$ _N_times_thm	322
$\mathbb{Z}$ _abs_minus_thm	324	$\mathbb{Z}$ _N_¬_minus_thm	322
$\mathbb{Z}$ _abs_plus_thm	324	$\mathbb{Z}$ _N_¬_plus1_thm	322
$\mathbb{Z}$ _abs_thm	324	$\mathbb{Z}$ _∈_N_thm	323
$\mathbb{Z}$ _abs_times_thm	324	$\mathbb{Z}$ _≤_antisym_thm	323
$\mathbb{Z}$ _abs_N_thm	324	$\mathbb{Z}$ _≤_cases_thm	323
$\mathbb{Z}$ _cases_thm1	320	$\mathbb{Z}$ _≤_clauses	321
$\mathbb{Z}$ _cases_thm	320	$\mathbb{Z}$ _≤_induction_thm	324
$\mathbb{Z}$ _cov_induction_thm	324	$\mathbb{Z}$ _≤_less_eq_thm	323
$\mathbb{Z}$ _def	318	$\mathbb{Z}$ _≤_less_trans_thm	323
$\mathbb{Z}$ _div_mod_unique_lemma1	324	$\mathbb{Z}$ _≤_minus_thm	321
$\mathbb{Z}$ _div_mod_unique_lemma2	324	$\mathbb{Z}$ _≤_plus_N_thm	323
$\mathbb{Z}$ _div_mod_unique_lemma3	324	$\mathbb{Z}$ _≤_refl_thm	323
$\mathbb{Z}$ _div_mod_unique_thm	324	$\mathbb{Z}$ _≤_trans_thm	323
$\mathbb{Z}$ _eq_thm1	322	$\mathbb{Z}$ _≤_≤_0_thm1	323
$\mathbb{Z}$ _eq_thm	320	$\mathbb{Z}$ _≤_≤_0_thm	321
$\mathbb{Z}$ _fun_∃_thm	324	$\mathbb{Z}$ _¬_less_thm	323
$\mathbb{Z}$ _induction_thm	320	$\mathbb{Z}$ _¬_N_thm	322
$\mathbb{Z}$ _less_cases_thm	323	$\mathbb{Z}$ _¬_≤_thm	323
$\mathbb{Z}$ _less_clauses	323	$\mathbb{Z}\mathbb{R}$ _consistent	312
$\mathbb{Z}$ _less_irrefl_thm	323	$\mathbb{Z}\mathbb{R}$ _minus_thm	312
$\mathbb{Z}$ _less_less_0_thm1	323	$\mathbb{Z}\mathbb{R}$ _plus_homomorphism_thm	312
$\mathbb{Z}$ _less_less_0_thm	323	$\mathbb{Z}\mathbb{R}$ _times_homomorphism_thm	312
$\mathbb{Z}$ _less_trans_thm	323	$\mathbb{Z}\mathbb{R}$ _NZ_thm	312
$\mathbb{Z}$ _less_≤_trans_thm	323	$\mathbb{Z}\mathbb{R}$	294
$\mathbb{Z}$ _minus_clauses	320	$\mathbb{Z}\mathbb{R}$	297
$\mathbb{Z}$ _minus_less_thm	323	$\mathbb{Z}$	318
$\mathbb{Z}$ _minus_thm	322	$\forall$ _def	262
$\mathbb{Z}$ _minus_N_≤_thm	323	$\forall$ _rewrite_thm	266
$\mathbb{Z}$ _minus_≤_thm	321	$\forall$	262
$\mathbb{Z}$ _plus0_thm	320	$\geq$ _def	314
$\mathbb{Z}$ _plus_assoc_thm1	320	$\geq_R$	294
$\mathbb{Z}$ _plus_assoc_thm	320	$\geq_R$	295
$\mathbb{Z}$ _plus_clauses	320	$\geq_R$	296
$\mathbb{Z}$ _plus_comm_thm	320	$\geq_Z$	317
$\mathbb{Z}$ _plus_eq_thm	322	$\geq_Z$	318
$\mathbb{Z}$ _plus_minus_thm	320	$\geq_Z$	319
$\mathbb{Z}$ _plus_order_thm	320	$\geq$	295
$\mathbb{Z}$ _times0_thm	322	$\geq$	313
$\mathbb{Z}$ _times1_thm	322	$\geq$	314
$\mathbb{Z}$ _times_assoc_thm1	322	$\geq$	318
$\mathbb{Z}$ _times_assoc_thm	321	$\in$ _in_clauses	279
$\mathbb{Z}$ _times_clauses	322	$\in_l$ _elems_thm	288

$\in_l$ -extract_thm	289	$\neg \exists$ _thm	265
$\in_l$ - $\hat{\ }_thm$	288	$\neg \forall$ _thm	265
$\in_l$	282	$\neg \wedge$ _thm	265
$\in_l$	283	$\neg \Leftrightarrow$ _thm	265
$\in$	276	$\neg \leq_{plus1}$ _thm	287
$\in$	277	$\neg \leq$ _thm	316
$\mapsto$	253	$\neg \neg$ _thm	265
$\lambda$	263	$\neg \vee$ _thm	265
$\wedge$ _def	262	$\neg \Rightarrow$ _thm	265
$\wedge$ _rewrite_thm	265	$\neg$	262
$\wedge$ _thm	265	$\vee$ _def	262
$\wedge$	262	$\vee$ _rewrite_thm	265
$\Leftrightarrow$ _rewrite_thm	265	$\vee$ _thm	265
$\Leftrightarrow$ _thm	265	$\vee$	262
$\Leftrightarrow$	5	$\%fun$ _thm1	285
$\Leftrightarrow$	6	$\%fun$ _thm	285
$\Leftrightarrow$	264	$\%graph$ _null_thm	285
$\leq$ _antisym_thm	316	$\%null$ _thm1	285
$\leq$ _cases_thm	316	$\%null$ _thm	285
$\leq$ _clauses	315	$\%singleton$ _thm	283
$\leq$ _def	314	$\% \cup$ _thm1	283
$\leq$ _least_upper_bound_thm	316	$\% \cup$ _thm	283
$\leq$ _max_thm	250	$\%f$	242
$\leq$ _plus1_thm	316	$\%f$	243
$\leq$ _plus_one_thm	287	$\%$	5
$\leq$ _trans_thm	315	$\%$	6
$\leq$ _well_order_thm	316	$\mapsto$	5
$\leq_R$	294	$\mapsto$	6
$\leq_R$	295	$\triangleleft$ _null_thm	286
$\leq_R$	296	$\triangleleft$ _thm	286
$\leq_Z$	317	$\triangleleft$	5
$\leq_Z$	318	$\triangleleft$	6
$\leq_Z$	319	$\triangleright$	5
$\leq$	295	$\triangleright$	6
$\leq$	313	$\hat{\ }_Cons$ _thm	284
$\leq$	314	$\hat{\ }_empty$ _thm	290
$\leq$	318	$\hat{\ }_thm1$	287
$\neg$ giveError_eq_giveVal_thm	49	$\hat{\ }_thm$	287
$\neg$ giveVal_eq_giveError_thm	49	$\hat{\ }_ \uparrow$ _thm	286
$\neg$ isError_giveVal_thm	44	$\hat{\ }$	274
$\neg$ isVal_giveError_thm	44	$\mapsto$	253
$\neg$ _cons_thm	284	$\hat{\ }_N$	294
$\neg$ _def	262	$\hat{\ }_N$	295
$\neg$ _empty_list_tree_lemma	293	$\hat{\ }_N$	297
$\neg$ _f_thm	265	$\hat{\ }_Z$ _consistent	312
$\neg$ _if_thm	265	$\hat{\ }_Z$	294
$\neg$ _less_plus1_thm	316	$\hat{\ }_Z$	295
$\neg$ _less_thm	316	$\hat{\ }_Z$	297
$\neg$ _plus1_thm	314	$\hat{\ }$	11
$\neg$ _plus1_ $\leq$ _thm	316	$\hat{\ }$	12
$\neg$ _rewrite_thm	265	$\hat{\ }$	295
$\neg$ _thm1	265	$\uparrow$ _extract_null_thm	288
$\neg$ _thm	265	$\uparrow$ _null_map_hide_lemma	210
$\neg$ _t_thm	265	$\uparrow$ _null_thm	288

---

$\uparrow\_thm1$ .....	251	$\subseteq\_size\_less\_thm$ .....	250
$\uparrow\_thm2$ .....	251	$\subseteq\_size\_<\_thm$ .....	250
$\uparrow\_thm3$ .....	251	$\subseteq$ .....	276
$\uparrow\_thm4$ .....	251	$\subseteq$ .....	277
$\uparrow\_&cap\_lemma$ .....	210	$\rightarrow$ .....	253
$\uparrow\_&uparrow\_thm$ .....	233	$\times\_def$ .....	273
$\uparrow$ .....	274	$\times\_local\_thm$ .....	292
$\uparrow$ .....	275	$\times$ .....	5
$\ominus\_clauses$ .....	278	$\times$ .....	6
$\ominus\_def$ .....	277	$\times$ .....	272
$\ominus$ .....	276	$\times$ .....	273
$\ominus$ .....	277	$\times\Rightarrow$ .....	248
$\oplus\_null\_thm1$ .....	288	$\times\Rightarrow$ .....	249
$\oplus\_null\_thm2$ .....	288	$\times\rightarrow$ .....	253
$\oplus\_null\_thm$ .....	287	$\Rightarrow$ .....	248
$\oplus\_single\_thm1$ .....	287	$\Rightarrow$ .....	249
$\oplus\_single\_thm2$ .....	287	$\rightarrow$ .....	253
$\oplus\_single$ .....	286	$\Rightarrow$ .....	253
$\oplus\_thm$ .....	286	$*$ .....	6
$\oplus$ .....	6	$*$ .....	7
$\oplus$ .....	7	$+$ .....	6
$\Rightarrow\_antisym\_axiom$ .....	256	$+$ .....	7
$\Rightarrow\_rewrite\_thm$ .....	266	$\perp$ .....	232
$\Rightarrow\_thm$ .....	265	$\sim$ .....	6
$\rightarrow$ .....	253	$\sim$ .....	282
$\Rightarrow$ .....	263	$-1$ .....	294
$\rightarrow$ .....	263	$-1$ .....	295
$\triangleright\_null\_thm$ .....	285	$-1$ .....	297
$\triangleright\_singleton\_thm$ .....	285		
$\triangleright$ .....	5		
$\triangleright$ .....	6		
$\sim\_R\_consistent$ .....	300		
$\sim\_R$ .....	294		
$\sim\_R$ .....	296		
$\sim\_Z\_consistent$ .....	319		
$\sim\_Z$ .....	317		
$\sim\_Z$ .....	318		
$\sim$ .....	276		
$\sim$ .....	277		
$\sim$ .....	295		
$\sim$ .....	318		
$\subset\_antisym\_thm$ .....	269		
$\subset\_clauses$ .....	278		
$\subset\_def$ .....	277		
$\subset\_irrefl\_thm$ .....	269		
$\subset\_trans\_thm$ .....	269		
$\subset$ .....	276		
$\subset$ .....	277		
$\subseteq\_clauses$ .....	278		
$\subseteq\_cleanColCons\_lemma1$ .....	41		
$\subseteq\_cleanColCons\_lemma$ .....	41		
$\subseteq\_def$ .....	277		
$\subseteq\_dir\_lemma$ .....	42		
$\subseteq\_finite\_thm$ .....	249		

---