

Project: DRA FRONT END FILTER PROJECT

Title: Proof of Security (IIb)

Ref: DS/FMU/FEF/011 *Issue: Revision : 2.5* *Date:* 5 December 2009

Status: Approved *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 provides a formal proof for two of the conjuncts of the security property on the relationship between *hide* and *updateState* for 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
0.1	Contents List	2
0.2	Document Cross References	2
0.3	Changes History	2
0.4	Changes Forecast	3
1	GENERAL	4
1.1	Scope	4
1.2	Introduction	4
2	PRELIMINARIES	4
3	CONSISTENCY PROOFS	4
3.1	Retrieving the Remaining Definitions of Constants	6
4	AUXILIARY THEOREMS	7
5	PROOF OF SECURITY OF CRITICAL COMPONENTS	13
5.1	Proof of Conjunct 4	13
5.2	Conjunct 3	13
5.2.1	Auxiliary Lemmas	13
5.2.2	Proof of Conjunct 3	57
6	CLOSING DOWN	59
7	THE THEORY fef011	60
7.1	Parents	60
7.2	Children	60
7.3	Theorems	60
8	INDEX	67

0.2 Document Cross References

- [1] DS/FMU/017. *Secure Database Technical Proposal*. High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.
- [2] DS/FMU/FEF/004. *Specification of SSQL Semantics I*. G.M. Prout, ICL Secure Systems, WIN01.
- [3] DS/FMU/FEF/007. *Proof Strategy*. G.M. Prout, ICL Secure Systems, WIN01.

0.3 Changes History

Issue Revision : 2.5 (5 December 2009) Corrected L^AT_EX errors.

Issue 2.5 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document provides a formal proof that the components *hide* and *updateState* satisfy some of their critical requirements, as specified in the proof strategy [3]. It constitutes part of deliverable D6 of work package 1c, as given in section 7 of the Secure Database Technical Proposal, [1].

1.2 Introduction

This document is a proof script which provides a formal proof which contributes to the proof of the second conjunct of *Lemma1*, the requirement on the critical components *hide* and *updateState*, described in the proof strategy document [3].

Lemma1

$$\vdash \quad \text{hide} \in \text{secureHide} \wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate}$$

In this document, we give proofs of the third and fourth conjuncts of *secureUpdate*:

$$\vdash \quad \forall c s e \bullet \text{Fst}(\text{Snd}(\text{updateState}(c, e, s))) = c$$

and

$$\begin{aligned} \vdash \quad & \forall c s_1 s_2 e \bullet \text{hide}(c, s_1) = \text{hide}(c, s_2) \\ & \Rightarrow \text{Snd}(\text{updateState}(c, e, s_1)) = \text{Snd}(\text{updateState}(c, e, s_2)) \end{aligned}$$

2 PRELIMINARIES

The following ProofPower instructions set up the new theory *fef011*.

SML

```
open_theory "fef010";
(force_delete_theory "fef011" handle _ => ());
new_theory "fef011";
push_merge_pcs["hol", "wrk049", "pair1"];
```

3 CONSISTENCY PROOFS

We satisfy the consistency proof obligations for constants defined in [2] that are needed in the proofs that follow.

SML

```

| val InsertEffect_def = get_spec⌈InsertEffect⌋;
| push_consistency_goal⌈destInsert⌋;
| a(rewrite_tac[InsertEffect_def]);
| a(∃_tac⌈(OutL, OutL o OutR, OutL o OutR o OutR, OutR o OutR o OutR)⌋);
| a(rewrite_tac[]);
| save_consistency_thm⌈destInsert⌋(pop_thm());
| val destInsert_def = get_spec⌈destInsert⌋;

```

HOL output

```

| destInsert_def =
| ⊢ ∀ i d u s
| • destInsert (InsertEffect i) = i
|   ∧ destDelete (DeleteEffect d) = d
|   ∧ destUpdate (UpdateEffect u) = u
|   ∧ destSelect (SelectEffect s) = s

```

SML

```

| push_consistency_goal⌈tabFromEffect⌋;
| a(rewrite_tac[get_spec⌈InsertEffect⌋]);
| a(∃_tac⌈λ x : Effect •
|   if IsL x
|   then Fst(OutL x)
|   else if IsL(OutR x)
|   then Fst(OutL(OutR x))
|   else Fst(OutL(OutR(OutR x)))⌋);
| a(rewrite_tac[sum_clauses]);
| save_consistency_thm⌈tabFromEffect⌋(pop_thm());
| val tabFromEffect_def = get_spec⌈tabFromEffect⌋;

```

HOL output

```

| tabFromEffect_def =
| ⊢ ∀ i d u
| • tabFromEffect (InsertEffect i) = Fst i
|   ∧ tabFromEffect (DeleteEffect d) = Fst d
|   ∧ tabFromEffect (UpdateEffect u) = Fst u

```

SML

```

| push_consistency_goalΓ giveVal∇;
| a(∃_tacΓ InL, InR, OutL, OutR, IsL, IsR∇);
| a(rewrite_tac[sum_clauses]);
| a(REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
| a(∃_tacΓ OutL ve∇ THEN asm_rewrite_tac[]);
| a(∃_tacΓ OutR ve∇ THEN asm_rewrite_tac[]);
| save_consistency_thmΓ giveVal∇(pop_thm());
| val giveVal_def = get_specΓ giveVal∇;

```

HOL output

```

| giveVal_def =
| ⊢ ∀ v e ve
|   • giveVal v = InL v
|     ∧ giveError e = InR e
|     ∧ destVal (giveVal v) = v
|     ∧ destError (giveError e) = e
|     ∧ (isVal ve ⇔ (∃ v1 • ve = giveVal v1))
|     ∧ (isError ve ⇔ (∃ e1 • ve = giveError e1))

```

3.1 Retrieving the Remaining Definitions of Constants

SML

```

| val changeSpec_def = conv_rule(MAP_C let_conv)(get_specΓ changeSpec∇);
| val replaceRows_def = get_specΓ replaceRows∇;
| val updateQuery_def = get_specΓ updateQuery∇;
| val isInsert_def = get_specΓ isInsert∇;
| val isDelete_def = get_specΓ isDelete∇;
| val isUpdate_def = get_specΓ isUpdate∇;
| val isSelect_def = get_specΓ isSelect∇;
| val getTable_def = get_specΓ getTable∇;
| val tabExists_def = get_specΓ tabExists∇;
| val colDefaults_def = get_specΓ colDefaults∇;
| val visibleCols_def = get_specΓ visibleCols∇;
| val revealRow_def = conv_rule(MAP_C let_conv)(get_specΓ revealRow∇);
| val insertQuery_def = conv_rule(MAP_C let_conv)(get_specΓ insertQuery∇);
| val updateRow_def = conv_rule(MAP_C let_conv)(get_specΓ updateRow∇);
| val updateField_def = get_specΓ updateField∇;
| val updateStateR_def = conv_rule(MAP_C let_conv)(get_specΓ updateStateR∇);
| val updateState_def = conv_rule(MAP_C let_conv)(get_specΓ updateState∇);

```

4 AUXILIARY THEOREMS

First we simplify some of the constant defining theorems.

SML

```

| val giveVal_eq_thm = save_thm("giveVal_eq_thm",prove_rule[giveVal_def]
|    $\lceil \forall x y \bullet \text{giveVal } x = \text{giveVal } y \Leftrightarrow x = y \rceil$ );
| val giveError_eq_thm = save_thm("giveError_eq_thm",prove_rule[giveVal_def]
|    $\lceil \forall x y \bullet \text{giveError } x = \text{giveError } y \Leftrightarrow x = y \rceil$ );
| val  $\neg$ isError_giveVal_thm = save_thm(" $\neg$ isError_giveVal_thm",
|   prove_rule[get_spec $\lceil$ isError $\rceil$ ] $\lceil \forall v \bullet \neg \text{isError}(\text{giveVal } v) \rceil$ );
| val  $\neg$ isVal_giveError_thm = save_thm(" $\neg$ isVal_giveError_thm",
|   prove_rule[get_spec $\lceil$ isError $\rceil$ ] $\lceil \forall e \bullet \neg \text{isVal}(\text{giveError } e) \rceil$ );
| val isVal_def = all_ $\forall$ _intro(nth 4 (strip_ $\wedge$ _rule (all_ $\forall$ _elim giveVal_def)));
| val isError_def = all_ $\forall$ _intro(nth 5 (strip_ $\wedge$ _rule (all_ $\forall$ _elim giveVal_def)));

```

HOL output

```

| giveVal_eq_thm =  $\vdash \forall x y \bullet \text{giveVal } x = \text{giveVal } y \Leftrightarrow x = y$ 
| giveError_eq_thm =  $\vdash \forall x y \bullet \text{giveError } x = \text{giveError } y \Leftrightarrow x = y$ 
|  $\neg$ isError_giveVal_thm =  $\vdash \forall v \bullet \neg \text{isError}(\text{giveVal } v)$ 
|  $\neg$ isVal_giveError_thm =  $\vdash \forall e \bullet \neg \text{isVal}(\text{giveError } e)$ 
| isVal_def =  $\vdash \forall ve \bullet \text{isVal } ve \Leftrightarrow (\exists v_1 \bullet ve = \text{giveVal } v_1)$ 
| isError_def =  $\vdash \forall ve \bullet \text{isError } ve \Leftrightarrow (\exists e_1 \bullet ve = \text{giveError } e_1)$ 

```

SML

```

| val thm1 = (push_goal([], $\lceil \forall x y z \bullet \neg((z = \text{InL } x) \wedge (z = \text{InR } y)) \rceil$ );
| a(REPEAT strip_tac THEN asm_rewrite_tac[]);
| pop_thm());

```

We provide the obvious results for sum types.

SML

```

| push_goal([], $\lceil \forall q \bullet (\text{isInsert } q \wedge \neg(\text{isDelete } q \vee \text{isUpdate } q \vee \text{isSelect } q))$ 
|    $\vee (\text{isDelete } q \wedge \neg(\text{isInsert } q \vee \text{isUpdate } q \vee \text{isSelect } q))$ 
|    $\vee (\text{isUpdate } q \wedge \neg(\text{isInsert } q \vee \text{isDelete } q \vee \text{isSelect } q))$ 
|    $\vee (\text{isSelect } q \wedge \neg(\text{isInsert } q \vee \text{isDelete } q \vee \text{isUpdate } q)) \rceil$ );
| a(rewrite_tac[isInsert_def,InsertEffect_def]);
| a( $\forall$ _tac THEN strip_tac THEN strip_tac);

```

SML

```

(* *** Goal "1" *** *)
a(prove_tac[] THEN_TRY asm_rewrite_tac[]);
(* *** Goal "1.1" *** *)
a(strip_asm_tac( $\forall$ _elim $\Gamma$ q $\neg$ sum_cases_thm));
(* *** Goal "1.1.1" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.1.2" *** *)
a(strip_asm_tac( $\forall$ _elim $\Gamma$ z $\neg$ sum_cases_thm));
(* *** Goal "1.1.2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2" *** *)
a(strip_asm_tac( $\forall$ _elim $\Gamma$ z' $\neg$ sum_cases_thm));
(* *** Goal "1.1.2.2.1" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2.2" *** *)
a( $\exists$ _tac $\Gamma$ z'' $\neg$  THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "1.2" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.3" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.4" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.5" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.6" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.7" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.8" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "1.9" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "1.10" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

| (* *** Goal "2" *** *)
| a(asm_prove_tac[] THEN_TRY asm_rewrite_tac[]);
| (* *** Goal "2.1" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.2" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.3" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.4" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.5" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.6" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);

```

SML

```

| (* *** Goal "2.7" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.8" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "2.9" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "2.10" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "2.11" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "2.12" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

| (* *** Goal "3" *** *)
| a(asm_prove_tac[] THEN_TRY asm_rewrite_tac[]);
| (* *** Goal "3.1" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.2" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.3" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.4" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.5" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.6" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);

```

SML

```

| (* *** Goal "3.7" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.8" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "3.9" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "3.10" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "3.11" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "3.12" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

| (* *** Goal "4" *** *)
| a(asm_prove_tac[] THEN_TRY asm_rewrite_tac[]);
| (* *** Goal "4.1" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "4.2" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "4.3" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "4.4" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "4.5" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "4.6" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "4.7" *** *)
| a(fc_tac[thm1] THEN asm_fc_tac[]);
| (* *** Goal "4.8" *** *)
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| val query_type = save_pop_thm"query_type";

```

HOL output

```

| query_type =  $\vdash \forall q$ 
|   • isInsert q  $\wedge \neg$  (isDelete q  $\vee$  isUpdate q  $\vee$  isSelect q)
|      $\vee$  isDelete q  $\wedge \neg$  (isInsert q  $\vee$  isUpdate q  $\vee$  isSelect q)
|      $\vee$  isUpdate q  $\wedge \neg$  (isInsert q  $\vee$  isDelete q  $\vee$  isSelect q)
|      $\vee$  isSelect q  $\wedge \neg$  (isInsert q  $\vee$  isDelete q  $\vee$  isUpdate q)

```

SML

```

push_goal([],Γ∀u• (isItem u ∧ ¬(isClass u ∨ isData u))
  ∨ (isClass u ∧ ¬(isItem u ∨ isData u))
  ∨ (isData u ∧ ¬(isItem u ∨ isClass u))∇);
a(rewrite_tac[get_specΓisItem∇,get_specΓItemUpdate∇]);
a(prove_tac[] THEN_TRY asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(strip_asm_tac(∀_elimΓu∇sum_cases_thm));
(* *** Goal "1.1" *** *)
a(asm_fc_tac[]);
(* *** Goal "1.2" *** *)
a(strip_asm_tac(∀_elimΓz∇sum_cases_thm));
(* *** Goal "1.2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "1.2.2" *** *)
a(∃_tacΓz'∇THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2" *** *)
a(fc_tac[thm1] THEN asm_fc_tac[]);
(* *** Goal "3" *** *)
a(fc_tac[thm1] THEN asm_fc_tac[]);
(* *** Goal "4" *** *)
a(fc_tac[thm1] THEN asm_fc_tac[]);
val update_type = save_pop_thm"update_type";

```

HOL output

```

update_type =
| ⊢ ∀ u
| • isItem u ∧ ¬ (isClass u ∨ isData u)
|   ∨ isClass u ∧ ¬ (isItem u ∨ isData u)
|   ∨ isData u ∧ ¬ (isItem u ∨ isClass u)

```

SML

```

push_goal([],Γ∀v• (isVal v ∧ ¬ isError v)
  ∨ (isError v ∧ ¬ isVal v)∇);
a(rewrite_tac[isVal_def,isError_def,giveVal_def]);
a(REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(fc_tac[sum_cases_thm]);
a(∃_tacΓz∇THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(∃_tacΓe1∇THEN asm_rewrite_tac[]);
val val_or_error_type = save_pop_thm"val_or_error_type";

```

HOL output

```
| val_or_error_type =
| ⊢ ∀ v • isVal v ∧ ¬ isError v ∨ isError v ∧ ¬ isVal v
```

5 PROOF OF SECURITY OF CRITICAL COMPONENTS

5.1 Proof of Conjunct 4

SML

```
| push_goal([], ⊢ ∀ c s e • Fst(Snd(updateState (c,e,s))) = c⊢);
| a(REPEAT ∀_tac);
| a(LEMMA_T⊢ e = (Fst e, Snd e)⊢ pure_once_asm_rewrite_thm_tac
|   THEN_LIST[rewrite_tac[], rewrite_tac[updateStateR_def, updateState_def]]);
| a(EVERY[strip_asm_tac(∀_elim⊢ Fst e⊢ query_type),
|   asm_rewrite_tac[],
|   cases_tac⊢ ¬ Snd e = []⊢,
|   asm_rewrite_tac[],
|   cases_tac⊢ tabExists c (tabFromEffect (Fst e)) (repState s)⊢,
|   asm_rewrite_tac[],
|   cases_tac⊢ c dominates TS_class (getTable (tabFromEffect (Fst e)) (repState s))⊢,
|   asm_rewrite_tac[]]);
| val conjunct4 = save_pop_thm "conjunct4";
```

HOL output

```
| conjunct4 = ⊢ ∀ c s e • Fst (Snd (updateState (c, e, s))) = c
```

5.2 Conjunct 3

5.2.1 Auxiliary Lemmas

Before we give a proof of conjunct 3, we provide a series of auxiliary results.

SML

```
| push_goal([], ⊢ ∀ c s • isState(hideR(c, repState s))⊢);
| a(REPEAT strip_tac);
| a(strip_asm_tac (∀_elim⊢ s⊢ isState_lemma));
| a(strip_asm_tac (list_∀_elim⊢ c⊢, repState s⊢ hideR_lemma));
| val isState_lemma3 = save_pop_thm "isState_lemma3";
```

HOL output

```
| isState_lemma3 = ⊢ ∀ c s • isState (hideR (c, repState s))
```

SML

```

push_goal([],Γ∀ c s1 s2 •
  hide(c,s1) = hide(c,s2)
  ⇔
  hideR (c, repState s1) = hideR (c, repState s2)Γ);
a(rewrite_tac[hide_def,rewrite_rule[isState_lemma3]
  (list_∇_elim[ΓhideR (c, repState s1)Γ,ΓhideR (c, repState s2)Γ]isState_lemma2)]);
val hide_eq_lemma = save_pop_thm"hide_eq_lemma";

```

HOL output

```

hide_eq_lemma =
| ⊢ ∀ c s1 s2
| • hide (c, s1) = hide (c, s2)
| ⇔ hideR (c, repState s1) = hideR (c, repState s2) : THM

```

SML

```

push_goal([],Γ∀ c s1 s2
  • hide (c, s1) = hide (c, s2)
  ⇒ ∀ i • tabExists c i (repState s1) ⇔ tabExists c i (repState s2)Γ);
a(rewrite_tac[tabExists_def,hideR_def,hide_eq_lemma,▷_thm,
  r_∅_r_thm,rel_ext_clauses,graph_thm]);
a(REPEAT ∇_tac);
a(strip_asm_tac (rewrite_rule[get_specΓisStateΓ,get_specΓStateSΓ,↔_def,∩_def]
  (∇_elimΓs1ΓisState_lemma)));
a(strip_asm_tac (rewrite_rule[get_specΓisStateΓ,get_specΓStateSΓ,↔_def,∩_def]
  (∇_elimΓs2ΓisState_lemma)));
a(DROP_NTH_ASM_T 4 ante_tac THEN DROP_NTH_ASM_T 2 ante_tac
  THEN rewrite_tac[↔_def,get_specΓIdeLΓ,get_specΓDirectorySΓ,∩_def,×_def,
  get_specΓUniverseΓ,dom_def,▷_thm,r_∅_r_thm,graph_thm,rel_ext_clauses,get_specΓ$PΓ]
  THEN strip_tac THEN strip_tac THEN strip_tac);
a(∇_tac THEN ⇔_T strip_asm_tac);

```

SML

```

(* *** Goal "1" *** *)
a(strip_asm_tac(list_∀_elim[⌈ repState s1 ⌋, ⌈ Front i ⌋, ⌈ y ⌋]at_thm1));
a(DROP_NTH_ASM_T 6 (asm_tac o list_∀_elim[⌈ Front i ⌋, ⌈ cleanDirectory c y ⌋]));
a(LEMMA_T ⌈ ∃ z
    • (c dominates Dir_exist z ∧ (Front i, z) ∈ repState s1)
      ∧ cleanDirectory c y = cleanDirectory c z ⌋ asm_tac);
(* *** Goal "1.1" *** *)
a(∃_tac ⌈ y ⌋ THEN asm_rewrite_tac []);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac []);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN ⇒_tac);
a(∧_tac THEN LIST[∃_tac ⌈ z ⌋ THEN asm_rewrite_tac [],
    strip_asm_tac(list_∀_elim[⌈ repState s2 ⌋, ⌈ Front i ⌋, ⌈ z ⌋]at_thm1)]);
a(DROP_NTH_ASM_T 8 ante_tac THEN DROP_NTH_ASM_T 7 ante_tac
    THEN asm_rewrite_tac []);
a(REPEAT ⇒_tac);
a(DROP_NTH_ASM_T 4 ante_tac THEN
    asm_rewrite_tac[cleanDirectory_def, dir_components, get_spec ⌈ MkDirectory ⌋]
    THEN strip_tac);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac [] THEN strip_tac
    THEN asm_rewrite_tac []);
a(DROP_NTH_ASM_T 10 ante_tac THEN asm_rewrite_tac [] THEN strip_tac);
a(DROP_NTH_ASM_T 12 (strip_asm_tac o list_∀_elim[⌈ Front i ⌋, ⌈ y ⌋]));
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac []);
a(rewrite_tac[rel_ext_clauses, graph_thm, r_⊖_r_thm]
    THEN ⇒_T (asm_tac o list_∀_elim[⌈ Last i ⌋, ⌈ cleanTable c y' ⌋]));
a(LEMMA_T ⌈ ∃ z • (Last i, z) ∈ Dir_tables y ∧ cleanTable c y' = cleanTable c z ⌋ asm_tac);

```

SML

```

(* *** Goal "1.2.1" *** *)
a( $\exists$ _tac $\ulcorner$ y' $\urcorner$  THEN asm_rewrite_tac[]);
(* *** Goal "1.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN  $\Rightarrow$ _tac);
a( $\exists$ _tac $\ulcorner$ z' $\urcorner$  THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(strip_asm_tac(list_ $\forall$ _elim $\ulcorner$ repState s2 $\urcorner$ , $\ulcorner$ Front i $\urcorner$ , $\ulcorner$ y $\urcorner$ at_thm1));
a(DROP_NTH_ASM_T 6 (asm_tac o list_ $\forall$ _elim $\ulcorner$ Front i $\urcorner$ , $\ulcorner$ cleanDirectory c y $\urcorner$ ));
a(LEMMA_T $\ulcorner$  $\exists$  z
  • (c dominates Dir_exist z  $\wedge$  (Front i, z)  $\in$  repState s2)
     $\wedge$  cleanDirectory c y = cleanDirectory c z $\urcorner$ asm_tac);
(* *** Goal "2.1" *** *)
a( $\exists$ _tac $\ulcorner$ y $\urcorner$  THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN  $\Rightarrow$ _tac);
a( $\wedge$ _tac THEN LIST[ $\exists$ _tac $\ulcorner$ z $\urcorner$  THEN asm_rewrite_tac[],
  strip_asm_tac(list_ $\forall$ _elim $\ulcorner$ repState s1 $\urcorner$ , $\ulcorner$ Front i $\urcorner$ , $\ulcorner$ z $\urcorner$ at_thm1)]);
a(DROP_NTH_ASM_T 8 ante_tac THEN DROP_NTH_ASM_T 7 ante_tac
  THEN asm_rewrite_tac[]);
a(REPEAT  $\Rightarrow$ _tac);
a(DROP_NTH_ASM_T 4 ante_tac THEN
  asm_rewrite_tac[cleanDirectory_def,dir_components,get_spec $\ulcorner$ MkDirectory $\urcorner$ ]
  THEN strip_tac);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[] THEN strip_tac
  THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 10 ante_tac THEN asm_rewrite_tac[] THEN strip_tac);
a(DROP_NTH_ASM_T 13 (strip_asm_tac o list_ $\forall$ _elim $\ulcorner$ Front i $\urcorner$ , $\ulcorner$ y $\urcorner$ ));
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
a(rewrite_tac[rel_ext_clauses,graph_thm,r_3_r_thm]
  THEN  $\Rightarrow$ _T (asm_tac o list_ $\forall$ _elim $\ulcorner$ Last i $\urcorner$ , $\ulcorner$ cleanTable c y' $\urcorner$ ));
a(LEMMA_T $\ulcorner$  $\exists$  z • (Last i, z)  $\in$  Dir_tables y  $\wedge$  cleanTable c y' = cleanTable c z $\urcorner$ asm_tac);
(* *** Goal "2.2.1" *** *)
a( $\exists$ _tac $\ulcorner$ y' $\urcorner$  THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN  $\Rightarrow$ _tac);
a( $\exists$ _tac $\ulcorner$ z' $\urcorner$  THEN asm_rewrite_tac[]);
val tabExists_lemma = save_pop_thm"tabExists_lemma";

```

HOL output

```

| tabExists_lemma =
| ⊢ ∀ c s1 s2
|   • hide (c, s1) = hide (c, s2)
|     ⇒ (∀ i • tabExists c i (repState s1) ⇔ tabExists c i (repState s2))

```

SML

```

| push_goal([], ⊢ ∀ c t1 t2 s1 s2 i • (c dominates TS_class t1 ∧ c dominates TS_class t2
|   ∧ cleanTable c t1 = cleanTable c t2)
|   ⇒ Snd (insertQuery (c, i, s1, t1)) = Snd (insertQuery (c, i, s2, t2))⊢);

```

SML

```

| a(REPEAT strip_tac);
| a(POP_ASM_T ante_tac THEN asm_rewrite_tac
|   [cleanTable_def, get_spec⊢ MkTableSpec⊢, tab_components] THEN strip_tac);
| a(lemma_tac⊢ colDefaults c t1 = colDefaults c t2⊢);
| (* *** Goal "1" *** *)
| a(asm_rewrite_tac[ext_thm, colDefaults_def, visibleCols_def]);
| (* *** Goal "2" *** *)
| a(LEMMA_T⊢ i = (Fst i, Snd i)⊢ pure_once_asm_rewrite_thm_tac
|   THEN_LIST[rewrite_tac[], asm_rewrite_tac[insertQuery_def]]);
| a(cases_tac⊢ ⊃ Elms
|   (Map
|     (MkRow c o colDefaults c t2)
|     (Snd i))
|   ⊆ RowS⊢ THEN asm_rewrite_tac[]);
| val cleanTable_insertQuery_lemma = save_pop_thm "cleanTable_insertQuery_lemma";

```

HOL output

```

| cleanTable_insertQuery_lemma =
| ⊢ ∀ c t1 t2 s1 s2 i
|   • c dominates TS_class t1
|     ∧ c dominates TS_class t2
|     ∧ cleanTable c t1 = cleanTable c t2
|     ⇒ Snd (insertQuery (c, i, s1, t1))
|       = Snd (insertQuery (c, i, s2, t2))

```

SML

```

push_goal([],Γ∀ c t1 t2 • cleanTable c t1 = cleanTable c t2 ⇒
  ((c dominates (TS_class t1) ∧ c dominates (TS_class t2))
   ⇒ updateRow c (TS_class t1) = updateRow c (TS_class t2))⊃);
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 3 ante_tac);
a(asm_rewrite_tac[cleanTable_def,tab_components,get_specΓMkTableSpec⊃
  THEN strip_tac THEN asm_rewrite_tac]);
val updateRow_lemma = save_pop_thm"updateRow_lemma";

```

HOL output

```

updateRow_lemma =
| ⊢ ∀ c t1 t2
| • cleanTable c t1 = cleanTable c t2
|   ⇒ c dominates TS_class t1 ∧ c dominates TS_class t2
|   ⇒ updateRow c (TS_class t1) = updateRow c (TS_class t2)

```

SML

```

push_goal([],Γ∀ c tc d1 d2 u •
  (isError(updateField c tc (u,d1)) ∧ replaceData c d1 = replaceData c d2)
   ⇒ updateField c tc (u,d1) = updateField c tc (u,d2)⊃);
a(REPEAT strip_tac);
a(POP_ASM_T (ante_tac o rewrite_rule[replaceData_def]));
a(cases_tacΓc dominates Dat_class d1⊃ THEN
  cases_tacΓc dominates Dat_class d2⊃ THEN asm_rewrite_tac[] THEN strip_tac);
(* *** Goal "1" *** *)
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(POP_ASM_T(strip_asm_tac o rewrite_rule[get_specΓMkData⊃,data_components]));
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "3" *** *)
a(POP_ASM_T(strip_asm_tac o rewrite_rule[get_specΓMkData⊃,data_components]));
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "4" *** *)
a(POP_ASM_T(strip_asm_tac o rewrite_rule[get_specΓMkData⊃,data_components]));
a(DROP_NTH_ASM_T 4 ante_tac THEN rewrite_tac[updateField_def]);
a(cases_tacΓc = tc⊃ THEN asm_rewrite_tac[]);
a(cases_tacΓisItem u⊃ THEN asm_rewrite_tac[]);
a(cases_tacΓisClass u⊃ THEN asm_rewrite_tac[]);
a(cases_tacΓdestClass u dominates Dat_class d2⊃
  THEN asm_rewrite_tac[¬isError_giveVal_thm]);
val isError_updateField_lemma = save_pop_thm"isError_updateField_lemma";

```

HOL output

```

| isError_updateField_lemma =
| ⊢ ∀ c tc d1 d2 u
|   • isError (updateField c tc (u, d1))
|     ∧ replaceData c d1 = replaceData c d2
|     ⇒ updateField c tc (u, d1) = updateField c tc (u, d2)

```

SML

```

| push_goal([], ⊢ ∀ c r1 r2 t u •
|   (isError (updateRow c (TS_class t) (u, r1)) ∧
|     isError (updateRow c (TS_class t) (u, r2))
|     ∧ cleanRow c (Snd (cleanColCons c t)) r1
|       = cleanRow c (Snd (cleanColCons c t)) r2 ∧
|       Dom u ⊆ {n | ∃ c' • c' ∈ visibleCols c t ∧ CS_posn c' = n})
|     ⇒ updateRow c (TS_class t) (u, r1) = updateRow c (TS_class t) (u, r2)⊢);
| a(REPEAT strip_tac);
| a(POP_ASM_T (strip_asm_tac o rewrite_rule[sets_ext_clauses, visibleCols_def, dom_def]));
| a(DROP_NTH_ASM_T 2 (strip_asm_tac o rewrite_rule
|   [cleanRow_def, get_spec⊢ MkRow⊢, row_components]));
| a(DROP_NTH_ASM_T 5 ante_tac THEN DROP_NTH_ASM_T 4 ante_tac
|   THEN rewrite_tac[updateRow_def]);

```

SML

```

| a(cases_tac⊢ ⊢ u ∈ Functional⊢ THEN asm_rewrite_tac[]);
| a(cases_tac⊢ ((RelCombine u (R_data r1)
|   % Graph (updateField c (TS_class t))
|   ▷ {x | isError x}
|   % Graph destError = {}⊢
|   THEN cases_tac⊢ ((RelCombine u (R_data r2)
|     % Graph (updateField c (TS_class t))
|     ▷ {x | isError x}
|     % Graph destError = {}⊢
|     THEN asm_rewrite_tac[¬isError_giveVal_thm, giveError_eq_thm]));
| a(⇒_T (fn _ => id_tac) THEN ⇒_T (fn _ => id_tac) THEN
|   POP_ASM_T (fn _ => id_tac) THEN POP_ASM_T (fn _ => id_tac));

```

SML

```

a(lemma_tac $\Gamma$ ((RelCombine u (R_data r1)
                % Graph (updateField c (TS_class t))
                 $\triangleright$  {x|isError x}
                % Graph destError
= ((RelCombine u (R_data r2)
   % Graph (updateField c (TS_class t))
    $\triangleright$  {x|isError x}
   % Graph destError $\neg$ ));

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[rel_ext_clauses,r-%r_thm,rel_combine_def,graph_thm, $\triangleright$ _thm]);
a(REPEAT strip_tac);
(* *** Goal "1.1" *** *)
a(DROP_NTH_ASM_T 9 (asm_tac o  $\forall$ _elim $\Gamma$ x $\neg$ ));
a(LEMMA_T $\Gamma$  $\exists$ y $\bullet$ (x, y)  $\in$  u $\neg$ asm_tac);
(* *** Goal "1.1.1" *** *)
a( $\exists$ _tac $\Gamma$ Fst z' $\neg$  THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac
  THEN  $\Rightarrow$ _T asm_tac);
a(DROP_NTH_ASM_T 8 (asm_tac o rewrite_rule[rel_ext_clauses,filterRow_def,
  r-%r_thm,graph_thm, $\triangleleft$ _thm]));
a(POP_ASM_T (asm_tac o list_ $\forall$ _elim $\Gamma$ x $\neg$ , $\Gamma$ replaceData c (Snd z' $\neg$ )));

```

SML

```

a(LEMMA_T $\Gamma$  $\exists$ z $\bullet$ (( $\exists$  c' $\bullet$ c'  $\in$  Snd (cleanColCons c t)  $\wedge$  CS_posn c' = x)
   $\wedge$  (x, z)  $\in$  R_data r1)
   $\wedge$  replaceData c (Snd z') = replaceData c z $\neg$ asm_tac);
(* *** Goal "1.1.2.1" *** *)
a( $\exists$ _tac $\Gamma$ Snd z' $\neg$  THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN  $\Rightarrow$ _tac);
a( $\exists$ _tac $\Gamma$ z $\neg$  THEN asm_rewrite_tac[]);
a( $\exists$ _tac $\Gamma$ (Fst z',z'' $\neg$ ) THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 10 ante_tac THEN asm_rewrite_tac[]);
a(LEMMA_T $\Gamma$ z' = (Fst z',Snd z' $\neg$ ) $\neg$  pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], $\Rightarrow$ _tac]);
a(strip_asm_tac (list_ $\forall$ _elim $\Gamma$ c $\neg$ , $\Gamma$ TS_class t $\neg$ , $\Gamma$ Snd z' $\neg$ , $\Gamma$ z'' $\neg$ , $\Gamma$ Fst z' $\neg$ ]
  isError_updateField_lemma));
a(asm_rewrite_tac[]);

```

SML

```

(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 9 (asm_tac o  $\forall$ _elim $\ulcorner$ x $\urcorner$ ));
a(LEMMA_T $\ulcorner$  $\exists$ y $\bullet$  (x, y)  $\in$  u $\urcorner$ asm_tac);
(* *** Goal "1.2.1" *** *)
a( $\exists$ _tac $\ulcorner$ Fst z' $\urcorner$  THEN asm_rewrite_tac[]);
(* *** Goal "1.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac
  THEN  $\Rightarrow$ _T asm_tac);
a(DROP_NTH_ASM_T 8 (asm_tac o rewrite_rule[rel_ext_clauses,filterRow_def,
  r_ $\rightarrow$ _r_thm,graph_thm, $\triangleleft$ _thm]));
a(POP_ASM_T (asm_tac o list_ $\forall$ _elim $\ulcorner$ x $\urcorner$ , $\ulcorner$ replaceData c (Snd z') $\urcorner$ ));
a(LEMMA_T $\ulcorner$  $\exists$ z $\bullet$  (( $\exists$  c' $\bullet$  c'  $\in$  Snd (cleanColCons c t)  $\wedge$  CS_posn c' = x)
   $\wedge$  (x, z)  $\in$  R_data r2)
   $\wedge$  replaceData c (Snd z') = replaceData c z $\urcorner$ asm_tac);

```

SML

```

(* *** Goal "1.2.2.1" *** *)
a( $\exists$ _tac $\ulcorner$ Snd z' $\urcorner$  THEN asm_rewrite_tac[]);
(* *** Goal "1.2.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN  $\Rightarrow$ _tac);
a( $\exists$ _tac $\ulcorner$ z' $\urcorner$  THEN asm_rewrite_tac[]);
a( $\exists$ _tac $\ulcorner$ (Fst z',z'') $\urcorner$  THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 10 ante_tac THEN asm_rewrite_tac[]);
a(LEMMA_T $\ulcorner$ z' = (Fst z',Snd z') $\urcorner$  pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], $\Rightarrow$ _tac]);
a(strip_asm_tac (list_ $\forall$ _elim $\ulcorner$ c $\urcorner$ , $\ulcorner$ TS_class t $\urcorner$ , $\ulcorner$ Snd z' $\urcorner$ , $\ulcorner$ z'' $\urcorner$ , $\ulcorner$ Fst z' $\urcorner$ 
  isError_updateField_lemma));
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[]);
val isError_updateRow_lemma = save_pop_thm"isError_updateRow_lemma";

```

HOL output

```

isError_updateRow_lemma =
 $\vdash \forall c tc d_1 d_2 u$ 
  • isError (updateField c tc (u, d1))
     $\wedge$  replaceData c d1 = replaceData c d2
     $\Rightarrow$  updateField c tc (u, d1) = updateField c tc (u, d2)

```

SML

```

push_goal([],Γ∀ c r1 r2 t u •
  (isVal (updateRow c (TS_class t) (u, r1))
    ∧ cleanRow c (Snd (cleanColCons c t)) r1
      = cleanRow c (Snd (cleanColCons c t)) r2 ∧
    Dom u ⊆ {n | ∃ c' • c' ∈ visibleCols c t ∧ CS_posn c' = n})
  ⇒ isVal (updateRow c (TS_class t) (u, r2))Γ);
a(REPEAT strip_tac);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[sets_ext_clauses,visibleCols_def,dom_def]));
a(DROP_NTH_ASM_T 2 (strip_asm_tac o rewrite_rule
  [cleanRow_def,get_specΓMkRowΓ,row_components]));
a(DROP_NTH_ASM_T 4 ante_tac THEN rewrite_tac[updateRow_def]);

```

SML

```

a(cases_tacΓ¬ u ∈ FunctionalΓ
  THEN cases_tacΓ((RelCombine u (R_data r1)
    § Graph (updateField c (TS_class t)))
    ▷ {x|isError x})
    § Graph destError = {}Γ
  THEN cases_tacΓ((RelCombine u (R_data r2)
    § Graph (updateField c (TS_class t)))
    ▷ {x|isError x})
    § Graph destError = {}Γ
  THEN asm_rewrite_tac[¬isVal_giveError_thm,giveVal_def]);

```

SML

```

(* *** Goal "1" *** *)
a(⇒_tac THEN prove_∃_tac);
(* *** Goal "2" *** *)
a(swap_nth_asm_concl_tac 1);
a(POP_ASM_T (fn _ => id_tac) THEN POP_ASM_T ante_tac);
a(rewrite_tac[rel_ext_clauses,r_§_r_thm,rel_combine_def,graph_thm,▷_thm]);
a(REPEAT strip_tac);
a(swap_nth_asm_concl_tac 5);
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 7 (asm_tac o rewrite_rule
  [rel_ext_clauses,filterRow_def,r_§_r_thm,graph_thm,◁_thm]));
a(POP_ASM_T(asm_tac o list_∀_elim[ΓxΓ,ΓreplaceData c (Snd z')Γ]));
a(DROP_NTH_ASM_T 9 (asm_tac o ∀_elimΓxΓ));
a(LEMMA_TΓ∃z • ((∃ c' • c' ∈ Snd (cleanColCons c t) ∧ CS_posn c' = x)
  ∧ (x, z) ∈ R_data r2)
  ∧ replaceData c (Snd z') = replaceData c zΓasm_tac);

```

SML

```

(* *** Goal "2.1" *** *)
a( $\exists$ _tac $\ulcorner$ Snd z' $\urcorner$  THEN asm_rewrite_tac[]);
a(LEMMA_T $\ulcorner$  $\exists$ y $\bullet$  (x, y)  $\in$  u $\urcorner$ asm_tac);
(* *** Goal "2.1.1" *** *)
a( $\exists$ _tac $\ulcorner$ Fst z' $\urcorner$  THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN
   $\Rightarrow$ _T asm_tac);

```

SML

```

(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN  $\Rightarrow$ _tac);
a( $\exists$ _tac $\ulcorner$ x $\urcorner$  THEN REPEAT strip_tac);
a( $\exists$ _tac $\ulcorner$ y $\urcorner$  THEN rewrite_tac[]);
a( $\exists$ _tac $\ulcorner$ z $\urcorner$  THEN asm_rewrite_tac[]);
a( $\exists$ _tac $\ulcorner$ (Fst z', z'') $\urcorner$  THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
a(LEMMA_T $\ulcorner$ z' = (Fst z', Snd z') $\urcorner$  pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],  $\Rightarrow$ _tac]);
a(strip_asm_tac (list_ $\forall$ _elim $\ulcorner$ c $\urcorner$ ,  $\ulcorner$ TS_class t $\urcorner$ ,  $\ulcorner$ Snd z' $\urcorner$ ,  $\ulcorner$ z'' $\urcorner$ ,  $\ulcorner$ Fst z' $\urcorner$  $\urcorner$ 
  isError_updateField_lemma));
a(asm_rewrite_tac[]);
val isVal_updateRow_lemma = save_pop_thm "isVal_updateRow_lemma";

```

HOL output

```

isVal_updateRow_lemma =
 $\vdash \forall c r_1 r_2 t u$ 
  • isVal (updateRow c (TS_class t) (u, r1))
     $\wedge$  cleanRow c (Snd (cleanColCons c t)) r1
      = cleanRow c (Snd (cleanColCons c t)) r2
     $\wedge$  Dom u  $\subseteq$  {n |  $\exists c' \bullet c' \in$  visibleCols c t  $\wedge$  CS_posn c' = n}
     $\Rightarrow$  isVal (updateRow c (TS_class t) (u, r2))

```

SML

```

push_goal([],Γ∀ c r1 r2 t u •
  (cleanRow c (Snd (cleanColCons c t)) r1
   = cleanRow c (Snd (cleanColCons c t)) r2 ∧
   Dom u ⊆ {n | ∃ c' • c' ∈ visibleCols c t ∧ CS_posn c' = n})
  ⇒ (isError (updateRow c (TS_class t) (u, r1))
     ⇔ isError (updateRow c (TS_class t) (u, r2)))Γ);
a(REPEAT ∀_tac THEN ⇒_tac);
a(REPEAT strip_tac);
(* *** Goal "1" *** *)
a(strip_asm_tac(∀_elimΓ(updateRow c (TS_class t) (u, r2))Γval_or_error_type));
a(DROP_NTH_ASM_T 5(asm_tac o eq_sym_rule));
a(strip_asm_tac(list_∀_elimΓ[cΓ, r2Γ, r1Γ, tΓ, uΓ]isVal_updateRow_lemma));
a(strip_asm_tac(∀_elimΓ(updateRow c (TS_class t) (u, r1))Γval_or_error_type));
(* *** Goal "2" *** *)
a(strip_asm_tac(∀_elimΓ(updateRow c (TS_class t) (u, r1))Γval_or_error_type));
a(strip_asm_tac(list_∀_elimΓ[cΓ, r1Γ, r2Γ, tΓ, uΓ]isVal_updateRow_lemma));
a(strip_asm_tac(∀_elimΓ(updateRow c (TS_class t) (u, r2))Γval_or_error_type));
val isError_⇔_updateRow_lemma = save_pop_thm"isError_⇔_updateRow_lemma";

```

HOL output

```

| ⊢ ∀ c r1 r2 t u
| • cleanRow c (Snd (cleanColCons c t)) r1
|   = cleanRow c (Snd (cleanColCons c t)) r2
|   ∧ Dom u ⊆ {n | ∃ c' • c' ∈ visibleCols c t ∧ CS_posn c' = n}
| ⇒ (isError (updateRow c (TS_class t) (u, r1))
|    ⇔ isError (updateRow c (TS_class t) (u, r2)))

```

SML

```

push_goal([],Γ∀ x y (u : ℕ ↔ (ℕ ↔ Update)) c t • ((x, y) ∈ u ∧ Dom (∪ (Ran u))
  ⊆ {n | ∃ c' • c' ∈ Snd (cleanColCons c t) ∧ CS_posn c' = n})
  ⇒ Dom y ⊆ {n | ∃ c' • c' ∈ visibleCols c t ∧ CS_posn c' = n}Γ);
a(rewrite_tac[sets_ext_clauses, dom_thm, ran_thm, ∪_def] THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 2 (asm_tac o ∀_elimΓx'Γ));
a(LEMMA_TΓ(∃ y s • (x', y) ∈ s ∧ (∃ x • (x, s) ∈ u))Γasm_tac);
(* *** Goal "1" *** *)
a(∃_tacΓy'Γ THEN ∃_tacΓyΓ THEN asm_rewrite_tac[]);
a(∃_tacΓxΓ THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[visibleCols_def]);
val dom_∪_ran_lemma = save_pop_thm"dom_∪_ran_lemma";

```

HOL output

```

| dom_Union_ran_lemma =
| ⊢ ∀ x y u c t
|   • (x, y) ∈ u
|     ∧ Dom (Union (Ran u))
|       ⊆ {n | ∃ c' • c' ∈ Snd (cleanColCons c t) ∧ CS_posn c' = n}
|     ⇒ Dom y ⊆ {n | ∃ c' • c' ∈ visibleCols c t ∧ CS_posn c' = n}

```

SML

```

| push_goal([], ⌈∀(l:Row LIST) (last:Row) (s : Row ℙ) (u : ℕ ↔ (ℕ ↔ Update)) •
|   Dom((Squash (Id (Dom (ListRel l ▷ s))))~
|     ; u) ∩ Dom{(# l + 1, last)}={}⌋);
| a(REPEAT ∀_tac);
| a(rewrite_tac[enumerate_def, r_§_r_thm, inv_rel_def, dom_def, id_def, squash_def,
|   list_rel_def, ▷_thm, ◁_thm, dot_dot_def]);
| a(rewrite_tac[∩_def, sets_ext_clauses] THEN REPEAT strip_tac);
| a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac []);
| val doms_null_lemma1 = save_pop_thm "doms_null_lemma1";

```

HOL output

```

| doms_null_lemma1 =
| ⊢ ∀ l last s u
|   • Dom ((Squash (Id (Dom (ListRel l ▷ s))))~
|     ; u) ∩ Dom {(# l + 1, last)}
|     = {}

```

SML

```

| push_goal([], ⌈∀(l:Row LIST) (s : Row ℙ) (u : ℕ ↔ (ℕ ↔ Update)) •
|   Dom({(# (Squash (Id (Dom (ListRel l ▷ s)))) + 1, # l + 1)}~
|     ; u) ∩ Dom(ListRel l)={}⌋);
| a(REPEAT ∀_tac);
| a(rewrite_tac[dom_def, inv_rel_def, r_§_r_thm]);
| a(rewrite_tac[∩_def, sets_ext_clauses] THEN REPEAT strip_tac);
| a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac
|   [list_rel_def, ▷_thm, ◁_thm, dot_dot_def]);
| val doms_null_lemma2 = save_pop_thm "doms_null_lemma2";

```

HOL output

```

| doms_null_lemma2 =
| ⊢ ∀ l s u
|   • Dom ({(# (Squash (Id (Dom (ListRel l ▷ s)))) + 1, # l + 1)}~
|     ; u) ∩ Dom (ListRel l)
|     = {}

```

SML

```

push_goal([],Γ∀(l:Row LIST) (last:Row)(s : Row  $\mathbb{P}$ ) (u :  $\mathbb{N} \leftrightarrow (\mathbb{N} \leftrightarrow \text{Update})) \bullet$ 
  ∀ n1 n2 • (n1 ∈ Dom (((RelCombine
    (Squash
      (Id
        (Dom
          (ListRel l ▷ s))))~
        § u)
      (ListRel l)
    § Graph (updateRow c (TS_class t2)))
    ▷ {x|isError x})
    § Graph destError)
  ∧ n2 ∈ Dom (((RelCombine
    ({(#
      (Squash
        (Id
          (Dom
            (ListRel l ▷ s))))
        + 1, # l + 1)}~
      § u)
    {(# l + 1, last)}
    § Graph (updateRow c (TS_class t2)))
    ▷ {x|isError x})
    § Graph destError)) ⇒ n2 > n1¬);

```

SML

```

a(REPEAT ∀_tac);
a(rewrite_tac[list_rel_def,dot_dot_def,<_thm,dom_def,inv_rel_def,r_§-r_thm,
  rel_combine_def,r_§-r_thm,▷_thm]);
a(REPEAT strip_tac);
a(asm_rewrite_tac[less_def]);
val squash_doms_lemma = save_pop_thm"squash_doms_lemma";

```

HOL output

```

squash_doms_lemma =
| ⊢ ∀ l last s u n1 n2
| • n1
|   ∈ Dom
|     (((RelCombine
|       ((Squash (Id (Dom (ListRel l ▷ s)))) ~ % u)
|       (ListRel l)
|       % Graph (updateRow c (TS_class t2)))
|       ▷ {x|isError x})
|       % Graph destError)
|   ∧ n2
|     ∈ Dom
|       (((RelCombine
|         ({(# (Squash (Id (Dom (ListRel l ▷ s)))) + 1,
|           # l + 1)} ~
|           % u)
|         {(# l + 1, last)}
|         % Graph (updateRow c (TS_class t2)))
|         ▷ {x|isError x})
|         % Graph destError)
|     ⇒ n2 > n1

```

SML

```

push_goal([], ⊢ ∀ c t1 t2 •
  cleanRows c (Snd (cleanColCons c t2)) (TS_rows t1)
    = cleanRows c (Snd (cleanColCons c t2)) (TS_rows t2)
  ⇒ #(ListRel (TS_rows t1) ▷ {r|c dominates R_exist r}) =
    #(ListRel (TS_rows t2) ▷ {r|c dominates R_exist r})⌈);
a(REPEAT ∀_tac);
a(lemma_tac ⊢ ∃ l • TS_rows t1 = l⌈ THEN_LIST
  [prove_∃_tac, POP_ASM_T rewrite_thm_tac]);
a(lemma_tac ⊢ ∃ l1 • TS_rows t2 = l1⌈ THEN_LIST
  [prove_∃_tac, POP_ASM_T rewrite_thm_tac]);
a(intro_∀_tac (⌈ l1⌈, ⌈ l':Row LIST⌈ ));
a(rewrite_tac [cleanRows_def]);
a(REV_LIST_INDUCTION_T⌈ l⌈ asm_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[map_def,map_null_thm]);
a(∀_tac THEN ⇒_T (asm_tac o rewrite_rule
  [all_∀_intro(eq_sym_rule(all_∀_elim list_rel_list_thm))]);
a(asm_rewrite_tac[list_rel_null_thm,▷_null_thm]);
(* *** Goal "2" *** *)
a(REPEAT ∀_tac);
a(intro_∀_tac(⌈last⌋,⌈last⌋));
a(REV_LIST_INDUCTION_T⌈l'⌋asm_tac);

```

SML

```

(* *** Goal "2.1" *** *)
a(rewrite_tac[map_def,map_null_thm]);
a(∀_tac THEN cases_tac⌈c dominates R_exist last⌋ THEN asm_rewrite_tac[]);
a(⇒_T (asm_tac o rewrite_rule[all_∀_intro(eq_sym_rule(all_∀_elim list_rel_list_thm))]);
a(asm_rewrite_tac[list_rel_∧_singleton_thm,list_rel_null_thm,
  ▷_null_thm,∪_▷_thm,▷_singleton_thm]);
(* *** Goal "2.2" *** *)
a(REPEAT ∀_tac);
a(cases_tac⌈c dominates R_exist last'⌋ THEN asm_rewrite_tac[] THEN
  cases_tac⌈c dominates R_exist last⌋ THEN asm_rewrite_tac[map_∧_thm]);
(* *** Goal "2.2.1" *** *)
a(⇒_tac THEN DROP_NTH_ASM_T 6 (ante_tac o ∀_elim⌈l'⌋) THEN asm_rewrite_tac[]
  THEN ⇒_tac);
a(asm_rewrite_tac[list_rel_∧_singleton_thm,∪_▷_thm,▷_singleton_thm]);
a(lemma_tac⌈∀ l last • (ListRel l ▷ {r|c dominates R_exist r}) ∩ {(# l + 1, last)}
  = {}⌋);

```

SML

```

| (* *** Goal "2.2.1.1" *** *)
| a(REPEAT  $\forall$ _tac THEN rewrite_tac
|   [list_rel_def,  $\triangleright$ _thm,  $\triangleleft$ _thm, dot_dot_def,  $\cap$ _def, rel_ext_clauses]);
| a(REPEAT strip_tac);
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
| (* *** Goal "2.2.1.2" *** *)
| a(TOP_ASM_T (asm_tac o list_ $\forall$ _elim[ $\lceil l' \rceil$ ,  $\lceil last \rceil$ ]));
| a(asm_tac(list_ $\forall$ _elim[ $\lceil l' \rceil$ ,  $\lceil \{r \mid c \text{ dominates } R\_exist \ r\} \rceil$ ]fin_list_rel_ $\triangleright$ _thm));
| a(asm_tac( $\forall$ _elim( $\# l' + 1$ , last) $\lceil$ fin_set_thm5));
| a(ante_tac(list_ $\forall$ _elim[ $\lceil ListRel \ l' \triangleright \{r \mid c \text{ dominates } R\_exist \ r\} \rceil$ ,
|    $\lceil \{(\# l' + 1, last)\} \rceil$ size_thm7] THEN asm_rewrite_tac[size_thm1, size_singleton_thm]));
| a( $\Rightarrow$ _T rewrite_thm_tac);
| a(LIST_DROP_NTH_ASM_T [1,2,3] (fn _ => id_tac));
| a(POP_ASM_T (asm_tac o list_ $\forall$ _elim[ $\lceil l \rceil$ ,  $\lceil last' \rceil$ ]));
| a(asm_tac(list_ $\forall$ _elim[ $\lceil l \rceil$ ,  $\lceil \{r \mid c \text{ dominates } R\_exist \ r\} \rceil$ ]fin_list_rel_ $\triangleright$ _thm));
| a(asm_tac( $\forall$ _elim( $\# l + 1$ , last') $\lceil$ fin_set_thm5));
| a(ante_tac(list_ $\forall$ _elim[ $\lceil ListRel \ l \triangleright \{r \mid c \text{ dominates } R\_exist \ r\} \rceil$ ,
|    $\lceil \{(\# l + 1, last')\} \rceil$ size_thm7] THEN asm_rewrite_tac[size_thm1, size_singleton_thm]));

```

SML

```

| (* *** Goal "2.2.2" *** *)
| a(DROP_NTH_ASM_T 3(ante_tac o  $\forall$ _elim[ $\lceil last' \rceil$ ] THEN asm_rewrite_tac[map_ $\wedge$ _thm]);
| a( $\Rightarrow$ _T asm_tac THEN  $\Rightarrow$ _tac);
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[] THEN strip_tac);
| a(asm_rewrite_tac[list_rel_ $\wedge$ _ $\triangleright$ _thm]);

```

SML

```

| (* *** Goal "2.2.3" *** *)
| a(DROP_NTH_ASM_T 4(ante_tac o  $\forall$ _elim[ $\lceil l' \wedge [last] \rceil$ ] THEN asm_rewrite_tac[map_ $\wedge$ _thm]);
| a( $\Rightarrow$ _T asm_tac THEN  $\Rightarrow$ _tac);
| a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[] THEN strip_tac);
| a(asm_rewrite_tac[list_rel_ $\wedge$ _ $\triangleright$ _thm]);

```

SML

```

| (* *** Goal "2.2.4" *** *)
| a(asm_rewrite_tac[list_rel_ $\wedge$ _ $\triangleright$ _thm]);
| val cleanRows_size_lemma = save_pop_thm"cleanRows_size_lemma";

```

HOL output

```

cleanRows_size_lemma =
| ⊢ ∀ c t1 t2
| • cleanRows c (Snd (cleanColCons c t2)) (TS_rows t1)
|   = cleanRows c (Snd (cleanColCons c t2)) (TS_rows t2)
| ⇒ # (ListRel (TS_rows t1) ▷ {r|c dominates R_exist r})
|   = # (ListRel (TS_rows t2) ▷ {r|c dominates R_exist r})

```

SML

```

push_goal([], ⊢ ∀ c t1 t2 u •
  (Dom (∪ (Ran u)) ⊆ {n|∃ c' • c' ∈ visibleCols c t2 ∧ CS_posn c' = n}) ∧
  cleanRows c (Snd (cleanColCons c t2)) (TS_rows t1)
    = cleanRows c (Snd (cleanColCons c t2)) (TS_rows t2)
⇒ (((RelCombine ((revealRow c t1)~ § u)
  (ListRel (TS_rows t1))
  § Graph (updateRow c (TS_class t2)))
  ▷ {x|isError x})
  § Graph destError
= {} ⇔ (((RelCombine ((revealRow c t2)~ § u)
  (ListRel (TS_rows t2))
  § Graph (updateRow c (TS_class t2)))
  ▷ {x|isError x})
  § Graph destError = {}));

```

SML

```

a(REPEAT ∀_tac THEN ⇒_tac);
a(strip_asm_tac(list_∇_elim[⊢ c ⊢, ⊢ t1 ⊢, ⊢ t2 ⊢]cleanRows_size_lemma));
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
a(rewrite_tac[revealRow_def]);
a(lemma_tac ⊢ ∃ l • TS_rows t1 = l ⊢ THEN_LIST
  [prove_∃_tac, POP_ASM_T rewrite_thm_tac]);
a(lemma_tac ⊢ ∃ l1 • TS_rows t2 = l1 ⊢ THEN_LIST
  [prove_∃_tac, POP_ASM_T rewrite_thm_tac]);
a(intro_∇_tac(⊢ l1 ⊢, ⊢ l':Row LIST ⊢));
a(rewrite_tac[cleanRows_def]);
a(REV_LIST_INDUCTION_T ⊢ ⊢ l ⊢ asm_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[map_def,map_null_thm]);
a  $\forall$ _tac;
a(REV_LIST_INDUCTION_T $\Gamma$ l' $\neg$ asm_tac THEN rewrite_tac
  [list_rel_null_thm,rel_combine_null_thm1]);
a( $\forall$ _tac THEN cases_tac $\Gamma$ c dominates R_exist last $\neg$ THEN asm_rewrite_tac[]);
a( $\Rightarrow$ _T (asm_tac o rewrite_rule[all_ $\forall$ _intro(eq_sym_rule(all_ $\forall$ _elim list_rel_list_thm)))]));
a(rewrite_tac[list_rel_null_thm,rel_combine_null_thm1, $\wp$ _graph_null_thm,
   $\wp$ _null_thm,squash_ $\wedge$ _thm]
  THEN asm_rewrite_tac[id_dom_null_thm,squash_null_thm,inv_rel_ $\wp$ _null_thm,
  rel_combine_null_thm, $\wp$ _graph_null_thm]);

```

SML

```

(* *** Goal "2" *** *)
a(REPEAT  $\forall$ _tac);
a(intro_ $\forall$ _tac( $\Gamma$ last $\neg$ , $\Gamma$ last $\neg$ ));
a(REV_LIST_INDUCTION_T $\Gamma$ l' $\neg$ asm_tac);
(* *** Goal "2.1" *** *)
a( $\forall$ _tac THEN cases_tac $\Gamma$ c dominates R_exist last $\neg$ 
  THEN asm_rewrite_tac[map_def,map_null_thm]);
a( $\Rightarrow$ _T (asm_tac o rewrite_rule[all_ $\forall$ _intro(eq_sym_rule(all_ $\forall$ _elim list_rel_list_thm)))]));
a(asm_rewrite_tac[squash_ $\wedge$ _thm,list_rel_null_thm,rel_combine_null_thm1,
   $\wp$ _graph_null_thm, $\triangleright$ _null_thm]);
a(rewrite_tac[id_dom_null_thm,squash_null_thm,inv_rel_ $\wp$ _null_thm,
  rel_combine_null_thm, $\wp$ _graph_null_thm, $\triangleright$ _null_thm]);
(* *** Goal "2.2" *** *)
a(REPEAT  $\forall$ _tac);
a(cases_tac $\Gamma$ c dominates R_exist last $\neg$  THEN asm_rewrite_tac[] THEN
  cases_tac $\Gamma$ c dominates R_exist last $\neg$  THEN asm_rewrite_tac[map_ $\wedge$ _thm]);

```

SML

```

(* *** Goal "2.2.1" *** *)
a(REPEAT =>_tac);
a(DROP_NTH_ASM_T 7(ante_tac o  $\forall$ _elim $\lceil$ l' $\rceil$ ) THEN asm_rewrite_tac[]
  THEN =>_T asm_tac);
a(DROP_NTH_ASM_T 7 (fn _ => id_tac));
a(DROP_NTH_ASM_T 4 (fn _ => id_tac));
a(asm_rewrite_tac[squash_ $\wedge$ _thm,inv_rel_ $\cup$ _thm,rel_combine_ $\cup$ _thm1, $\%$ _ $\cup$ _thm]);
a(asm_rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine_ $\cup$ _thm]);
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[size_list_rel_ $\wedge$ _ $\triangleright$ _thm]);
a(=>_tac THEN DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]
  THEN =>_T asm_tac);

```

SML

```

a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l  $\triangleright$  {r|c dominates R_exist r})))) $\sim$ 
   $\%$  u $\rceil$ , $\lceil$ {(# l + 1, last)} $\rceil$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l  $\triangleright$  {r|c dominates R_exist r})))) $\sim$ 
   $\%$  u $\rceil$ , $\lceil$ {(# l + 1, last')} $\rceil$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);

```

SML

```

a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma2));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma2));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ {(#
  (Squash
    (Id
      (Dom
        (ListRel l
           $\triangleright$  {r
            |c dominates R_exist r})))) $\sim$ 
          + 1, # l + 1)} $\rceil$  $\sim$ 
   $\%$  u $\rceil$ , $\lceil$ ListRel l $\rceil$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);

```


SML

```

a(LEMMA_TΓ((RelCombine
    ({(#
        (Squash
            (Id
                (Dom
                    (ListRel l
                        ▷ {r
                            |c
                                dominates R_exist r}})))
                    + 1, # l + 1)}~
                % u)
            {(# l + 1, last')}
            % Graph (updateRow c (TS_class t2)))
            ▷ {x|isError x}
            % Graph destError) = {} ⇔ (((RelCombine
                ({(#
                    (Squash
                        (Id
                            (Dom
                                (ListRel l'
                                    ▷ {r
                                        |c
                                            dominates R_exist r}})))
                            + 1, # l' + 1)}~
                                % u)
                            {(# l' + 1, last')}
                            % Graph (updateRow c (TS_class t2)))
                            ▷ {x|isError x}
                            % Graph destError) = {}Γrewrite_thm_tac);

```

SML

```

a(POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[rel_ext_clauses,rel_combine_def,r_g-r_thm,inv_rel_def,▷_thm,graph_thm]);
a(REPEAT strip_tac);
(* *** Goal "2.2.1.1" *** *)
a(swap_nth_asm_concl_tac 7);
a(REPEAT strip_tac);
a(∃_tac# l + 1 THEN rewrite_tac[]);
a(REPEAT strip_tac);
a(∃_tac# y THEN rewrite_tac[]);
a(∃_tac# z THEN asm_rewrite_tac[]);
a(∃_tac# (Fst z',last') THEN rewrite_tac[]);
a(DROP_NTH_ASM_T 2 ante_tac);

```

SML

```

a(LEMMA_T# z' = (Fst z',Snd z') THEN pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],pure_asm_rewrite_tac[] THEN ⇒_tac]);
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[] THEN ⇒_tac);
a(DROP_NTH_ASM_T 12 (asm_tac o rewrite_rule[visibleCols_def]));
a(strip_asm_tac(list_∇_elim[# z'',# Fst z',# u, # c, # t₂] dom_∪_ran_lemma));
a(strip_asm_tac(list_∇_elim[# c, # last', # last, # t₂, # Fst z'] isError_↔_updateRow_lemma));
a(strip_asm_tac(list_∇_elim[# c, # last', # last, # t₂, # Fst z'] isError_updateRow_lemma));
a(asm_rewrite_tac[]);
a(∃_tac# z'' THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_∇_elim[# l, # l', # {r|c dominates R_exist r} ] size_squash_id_dom_thm));
a(asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.1.2" *** *)
a(swap_nth_asm_concl_tac 7);
a(REPEAT strip_tac);
a( $\exists$ _tac $\lceil$ # l' + 1 $\rceil$  THEN rewrite_tac[]);
a(REPEAT strip_tac);
a( $\exists$ _tac $\lceil$ y $\rceil$  THEN rewrite_tac[]);
a( $\exists$ _tac $\lceil$ z $\rceil$  THEN asm_rewrite_tac[]);
a( $\exists$ _tac $\lceil$ (Fst z',last) $\rceil$  THEN rewrite_tac[]);
a(DROP_NTH_ASM_T 2 ante_tac);
a(LEMMA_T $\lceil$ z' = (Fst z',Snd z') $\rceil$  pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],pure_asm_rewrite_tac[] THEN  $\Rightarrow$ _tac]);
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[] THEN  $\Rightarrow$ _tac);
a(DROP_NTH_ASM_T 12 (asm_tac o rewrite_rule[visibleCols_def]));
a(strip_asm_tac(list $\forall$ _elim $\lceil$ z'' $\rceil$ , $\lceil$ Fst z' $\rceil$ , $\lceil$ u $\rceil$ , $\lceil$ c $\rceil$ , $\lceil$ t $\rceil$ dom $\cup$ _ran_lemma));
a(strip_asm_tac(list $\forall$ _elim $\lceil$ c $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ t $\rceil$ , $\lceil$ Fst z' $\rceil$ isError $\Leftrightarrow$ _updateRow_lemma));
a(strip_asm_tac(list $\forall$ _elim $\lceil$ c $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ t $\rceil$ , $\lceil$ Fst z' $\rceil$ isError_updateRow_lemma));
a(asm_rewrite_tac[]);
a( $\exists$ _tac $\lceil$ z'' $\rceil$  THEN asm_rewrite_tac[]);
a(strip_asm_tac(list $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ l' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ size_squash_id_dom_thm));

```

SML

```

(* *** Goal "2.2.2" *** *)
a(DROP_NTH_ASM_T 3(ante_tac o  $\forall$ _elim $\lceil$ last' $\rceil$ ) THEN
  asm_rewrite_tac[map_ $\wedge$ _thm,size_list_rel_ $\wedge$ _▷_thm]);
a( $\Rightarrow$ _T asm_tac THEN REPEAT  $\Rightarrow$ _tac);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
a( $\Rightarrow$ _T rewrite_thm_tac);
a(asm_rewrite_tac[list_rel_ $\wedge$ _▷_thm]
  THEN rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine $\cup$ _thm]);
a(strip_asm_tac(list $\forall$ _elim $\lceil$ l' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l' ▷ {r|c dominates R_exist r})))~
  ; u) $\rceil$ , $\lceil$ {(# l' + 1, last)} $\rceil$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);

```

SML

```

(* *** Goal "2.2.3" *** *)
a(DROP_NTH_ASM_T 4(ante_tac o  $\forall$ _elim $\lceil$ l'  $\wedge$  [last] $\rceil$ ) THEN
  asm_rewrite_tac[map_ $\wedge$ _thm,size_list_rel_ $\wedge$ _▷_thm]);
a( $\Rightarrow$ _T asm_tac THEN REPEAT  $\Rightarrow$ _tac);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
a( $\Rightarrow$ _T (rewrite_thm_tac o eq_sym_rule));
a(asm_rewrite_tac[list_rel_ $\wedge$ _▷_thm]
  THEN rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine_ $\cup$ _thm]);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l ▷ {r|c dominates R_exist r})))) $\sim$ 
  % u) $\rceil$ , $\lceil$ {(# l + 1, last') $\rceil$ } $\rceil$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);

```

SML

```

(* *** Goal "2.2.4" *** *)
a(asm_rewrite_tac[list_rel_ $\wedge$ _▷_thm]
  THEN rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine_ $\cup$ _thm]);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l' $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l' ▷ {r|c dominates R_exist r})))) $\sim$ 
  % u) $\rceil$ , $\lceil$ {(# l' + 1, last) $\rceil$ } $\rceil$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l ▷ {r|c dominates R_exist r})))) $\sim$ 
  % u) $\rceil$ , $\lceil$ {(# l + 1, last') $\rceil$ } $\rceil$ rel_combine_null_thm2));
a(asm_rewrite_tac[]);
val cleanRows_errors_or_vals_lemma = save_pop_thm"cleanRows_errors_or_vals_lemma";

```

HOL output

```

cleanRows_errors_or_vals_lemma =
| ⊢ ∀ c t1 t2 u
| • Dom (∪ (Ran u)) ⊆ {n|∃ c'• c' ∈ visibleCols c t2 ∧ CS_posn c' = n}
|   ∧ cleanRows c (Snd (cleanColCons c t2)) (TS_rows t1)
|     = cleanRows c (Snd (cleanColCons c t2)) (TS_rows t2)
|   ⇒ (((RelCombine ((revealRow c t1)~ § u) (ListRel (TS_rows t1)))
|     § Graph (updateRow c (TS_class t2)))
|     ▷ {x|isError x})
|     § Graph destError
|   = {}
|   ⇔ (((RelCombine ((revealRow c t2)~ § u) (ListRel (TS_rows t2)))
|     § Graph (updateRow c (TS_class t2)))
|     ▷ {x|isError x})
|     § Graph destError
|   = {}

```

SML

```

push_goal([], ⌈∀c t• Graph (updateRow c (TS_class t)) ∈ Functional⌋);
a(rewrite_tac[updateRow_def,functional_def,graph_thm] THEN REPEAT strip_tac);
a(asm_rewrite_tac[]);
val fun_updateRow_thm = save_pop_thm"fun_updateRow_thm";

```

HOL output

```

fun_updateRow_thm =
| ⊢ ∀ c t• Graph (updateRow c (TS_class t)) ∈ Functional

```

SML

```

push_goal([], ⌈Graph destError ∈ Functional⌋);
a(rewrite_tac[get_spec⌈destError⌋,functional_def,graph_thm] THEN REPEAT strip_tac);
a(asm_rewrite_tac[]);
val fun_destError_thm = save_pop_thm"fun_destError_thm";

```

HOL output

```

fun_destError_thm = ⊢ Graph destError ∈ Functional

```

SML

```

push_goal([],Γ∀ l r c t u • u ∈ Functional ⇒
  (((RelCombine ((Squash
    (Id
      (Dom
        (ListRel l
          ▷ {r|c dominates R_exist r}))))~
      § u)
    (ListRel l)
    § Graph (updateRow c (TS_class t)))
  ▷ {x|isError x})
  § Graph destError) ∈ FiniteΓ);

```

SML

```

a(REPEAT strip_tac);
a(bc_tac[fin_§_thm]THEN_TRY rewrite_tac[fun_destError_thm]);
a(bc_tac[fin_▷_thm]);
a(bc_tac[fin_§_thm]THEN_TRY rewrite_tac[fun_updateRow_thm]);
a(bc_tac[fin_rel_combine_thm]THEN_TRY rewrite_tac[fin_list_rel_thm]);
a(bc_tac[fin_§_thm]THEN_TRY asm_rewrite_tac[]);
a(bc_tac[fin_inv_rel_thm]);
a(bc_tac[squash_id_fin_thm]);
a(rewrite_tac[fin_list_rel_▷_thm]);
val fin_lemma1 = save_pop_thm"fin_lemma1";

```

HOL output

```

fin_lemma1 =
| ⊢ ∀ l r c t u
  • u ∈ Functional
  ⇒ ((RelCombine
    ((Squash
      (Id (Dom (ListRel l ▷ {r|c dominates R_exist r}))))~
      § u)
    (ListRel l)
    § Graph (updateRow c (TS_class t)))
  ▷ {x|isError x})
  § Graph destError
  ∈ Finite

```

SML

```

push_goal([],Γ∇ l last r c t u • u ∈ Functional ⇒
  (((RelCombine ({(#
    (Squash
      (Id
        (Dom
          (ListRel l
            ▷ {r
              |c
                dominates R_exist r}}))))
    + 1, # l + 1)}~
    % u)
    {(# l + 1, last)}
    % Graph (updateRow c (TS_class t))
    ▷ {x|isError x}
    % Graph destError) ∈ FiniteΓ);

```

SML

```

a(REPEAT strip_tac);
a(bc_tac[fin_%_thm] THEN_TRY rewrite_tac[fun_destError_thm]);
a(bc_tac[fin_▷_thm]);
a(bc_tac[fin_%_thm] THEN_TRY rewrite_tac[fun_updateRow_thm]);
a(bc_tac[fin_rel_combine_thm] THEN_TRY rewrite_tac[fin_set_thm5]);
a(bc_tac[fin_%_thm] THEN_TRY asm_rewrite_tac[]);
a(rewrite_tac[inv_rel_singleton_thm,fin_set_thm5]);
val fin_lemma2 = save_pop_thm"fin_lemma2";

```

HOL output

```

| fin_lemma2 =
| ⊢ ∀ l last r c t u
|   • u ∈ Functional
|     ⇒ ((RelCombine
|         ( { (#
|             (Squash
|               (Id
|                 (Dom
|                   (ListRel l
|                     ▷ {r
|                       |c dominates R_exist r}))))
|                 + 1, # l + 1) } ~
|             % u)
|         { (# l + 1, last) }
|         % Graph (updateRow c (TS_class t))
|         ▷ {x|isError x}
|         % Graph destError
|         ∈ Finite

```

SML

```

| push_goal([], ⌈∀ c t1 t2 s1 s2 u • (c dominates TS_class t1 ∧ c dominates TS_class t2
|   ∧ cleanTable c t1 = cleanTable c t2)
|   ⇒ Snd (updateQuery (c, u, s1, t1) = Snd (updateQuery (c, u, s2, t2))⌋);
| a(REPEAT strip_tac);
| a(POP_ASM_T ante_tac THEN asm_rewrite_tac
|   [cleanTable_def, get_spec ⌈MkTableSpec⌋, tab_components] THEN strip_tac);
| a(POP_ASM_T ante_tac);
| a(lemma_tac ⌈cleanColCons c t1 = cleanColCons c t2⌋
|   THEN LIST[pure_once_asm_rewrite_tac[prove_rule[pair_clauses]
|   ⌈∀ p • p = (Fst p, Snd p)⌋] THEN asm_rewrite_tac[],
|   DROP_NTH_ASM_T 3(fn _ => id_tac)
|   THEN DROP_NTH_ASM_T 2(fn _ => id_tac)]);
| a(TOP_ASM_T rewrite_thm_tac THEN ⇒_tac);

```

SML

```

a(strip_asm_tac(list_∇_elim[⊢c⊢,⊢t1⊢,⊢t2⊢]cleanRows_size_lemma));
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
a(LEMMA_T⊢u = (Fst u,Snd u)⊢ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],rewrite_tac[updateQuery_def]]);
a(conv_tac(ONCE_MAP_C let_conv)THEN asm_rewrite_tac[visibleCols_def]);
a(cases_tac⊢Snd u ∈ Functional⊢ THEN
  cases_tac⊢Dom (∪ (Ran (Snd u))) ⊆
  {n | ∃ c' • c' ∈ Snd (cleanColCons c t2) ∧ CS_posn c' = n}⊢
  THEN asm_rewrite_tac[]);
a(conv_tac(MAP_C let_conv));
a ⇒_tac;
a(strip_asm_tac (rewrite_rule[visibleCols_def]
  (list_∇_elim[⊢c⊢,⊢t1⊢,⊢t2⊢,⊢Snd u⊢]cleanRows_errors_or_vals_lemma))
  THEN asm_rewrite_tac[]);
a(POP_ASM_T (fn _ => id_tac) THEN POP_ASM_T (fn _ => id_tac) THEN ⇒_tac);

```

SML

```

a(LEMMA_T⊢Squash
  (((RelCombine ((revealRow c t1) ~ § Snd u)
    (ListRel (TS_rows t1))
    § Graph (updateRow c (TS_class t2)))
  ▷ {x|isError x})
  § Graph destError) = Squash
  (((RelCombine ((revealRow c t2) ~ § Snd u)
    (ListRel (TS_rows t2))
    § Graph (updateRow c (TS_class t2)))
  ▷ {x|isError x})
  § Graph destError)⊢rewrite_thm_tac);

```

SML

```

a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac
  THEN rewrite_tac[cleanRows_def,revealRow_def]);
a(lemma_tac⊢∃ l • TS_rows t1 = l⊢THEN_LIST
  [prove_∃_tac,POP_ASM_T rewrite_thm_tac]);
a(lemma_tac⊢∃ l1 • TS_rows t2 = l1⊢THEN_LIST
  [prove_∃_tac,POP_ASM_T rewrite_thm_tac]);
a(intro_∇_tac(⊢l1⊢,⊢l':Row LIST⊢));
a(REV_LIST_INDUCTION_T⊢l⊢asm_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[map_def,map_null_thm,list_rel_null_thm,rel_combine_null_thm1,§_graph_null_thm,
             §_null_thm,▷_null_thm,squash_null_thm]);
a ∀_tac;
a(REV_LIST_INDUCTION_TΓl'∇asm_tac);
(* *** Goal "1.1" *** *)
a(rewrite_tac[list_rel_null_thm,rel_combine_null_thm1,§_graph_null_thm,
             ▷_null_thm,id_dom_null_thm,squash_null_thm]);
(* *** Goal "1.2" *** *)
a(∀_tac THEN cases_tacΓc dominates R_exist last∇THEN asm_rewrite_tac[]);
a(⇒_T (asm_tac o rewrite_rule[all_∇_intro(eq_sym_rule(all_∇_elim list_rel_list_thm)))]));
a(asm_rewrite_tac[list_rel_null_thm,rel_combine_null_thm,§_graph_null_thm,▷_null_thm,
                 id_dom_null_thm,squash_null_thm,list_rel_∧_▷_thm,inv_rel_§_null_thm]);

```

SML

```

(* *** Goal "2" *** *)
a(REPEAT ∀_tac);
a(intro_∇_tac(Γlast∇,Γlast∇));
a(REV_LIST_INDUCTION_TΓl'∇asm_tac);
(* *** Goal "2.1" *** *)
a(∀_tac THEN cases_tacΓc dominates R_exist last∇
   THEN asm_rewrite_tac[map_def,map_null_thm]);
a(⇒_T (asm_tac o rewrite_rule[all_∇_intro(eq_sym_rule(all_∇_elim list_rel_list_thm)))]));
a(asm_rewrite_tac[list_rel_null_thm,rel_combine_null_thm,§_graph_null_thm,▷_null_thm,
                 id_dom_null_thm,squash_null_thm,list_rel_∧_▷_thm,inv_rel_§_null_thm]);
(* *** Goal "2.2" *** *)
a(REPEAT ∀_tac);
a(cases_tacΓc dominates R_exist last∇ THEN asm_rewrite_tac[] THEN
   cases_tacΓc dominates R_exist last∇ THEN asm_rewrite_tac[map_∧_thm]);

```

SML

```

(* *** Goal "2.2.1" *** *)
a(REPEAT =>_tac);
a(DROP_NTH_ASM_T 7(ante_tac o  $\forall$ _elim $\lceil$ l $\rceil$ ) THEN asm_rewrite_tac[]
  THEN =>_T asm_tac);
a(DROP_NTH_ASM_T 7 (fn _ => id_tac));
a(DROP_NTH_ASM_T 4 (fn _ => id_tac));
a(asm_rewrite_tac[squash_ $\wedge$ _thm,inv_rel_ $\cup$ _thm,rel_combine_ $\cup$ _thm1, $\%$ _ $\cup$ _thm]);
a(asm_rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine_ $\cup$ _thm]);
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[size_list_rel_ $\wedge$ _ $\triangleright$ _thm]);
a(=>_tac THEN DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]
  THEN =>_T asm_tac);

```

SML

```

a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ Snd u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l  $\triangleright$  {r|c dominates R_exist r}))))~
   $\%$  Snd u) $\rceil$ , $\lceil$ {(# l + 1, last') $\rceil$ ]rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T (fn _ => id_tac));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l' $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ Snd u $\rceil$ doms_null_lemma1));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ ((Squash
  (Id (Dom (ListRel l'  $\triangleright$  {r|c dominates R_exist r}))))~
   $\%$  Snd u) $\rceil$ , $\lceil$ {(# l' + 1, last) $\rceil$ ]rel_combine_null_thm2));

```

SML

```

a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T (fn _ => id_tac));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$ , $\lceil$ Snd u $\rceil$ doms_null_lemma2));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ {(#
  (Squash
    (Id
      (Dom
        (ListRel l
           $\triangleright$  {r
            |c dominates R_exist r}))))
    + 1, # l + 1)}~
   $\%$  Snd u) $\rceil$ , $\lceil$ ListRel l $\rceil$ ]rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T (fn _ => id_tac));

```

SML

```

a(strip_asm_tac(list_∀_elim[ $\Gamma$   $l'$ ,  $\Gamma$   $\{r \mid c \text{ dominates } R\_exist\ r\}$ ,  $\Gamma$   $Snd\ u$ ] $\Uparrow$ doms_null_lemma2));
a(strip_asm_tac(list_∀_elim[ $\Gamma$  ( $\#$ 
    (Squash
      (Id
        (Dom
          (ListRel  $l'$ 
             $\triangleright$   $\{r$ 
               $\mid c \text{ dominates } R\_exist\ r\}$ )))))
    + 1,  $\# l' + 1$ )] $\sim$ 
  %  $Snd\ u$ ] $\Uparrow$ ListRel  $l'$ ] $\Uparrow$ rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[% $\cup$ _thm, $\cup$ - $\triangleright$ _thm]);

```

SML

```

a(lemma_tac $\Uparrow$ Ran(((RelCombine ( $\#$ 
    (Squash
      (Id
        (Dom
          (ListRel  $l$ 
             $\triangleright$   $\{r \mid c \text{ dominates } R\_exist\ r\}$ )))))
    + 1,  $\# l + 1$ )] $\sim$ 
  %  $Snd\ u$ )
   $\{(\# l + 1, last')\}$ 
  % Graph (updateRow  $c$  (TS_class  $t_2$ ))
   $\triangleright$   $\{x \mid isError\ x\}$ 
  % Graph destError) = Ran(((RelCombine
    ( $\#$ 
      (Squash
        (Id
          (Dom
            (ListRel  $l'$ 
               $\triangleright$   $\{r \mid c \text{ dominates } R\_exist\ r\}$ )))))
      + 1,  $\# l' + 1$ )] $\sim$ 
      %  $Snd\ u$ )
       $\{(\# l' + 1, last)\}$ 
      % Graph (updateRow  $c$  (TS_class  $t_2$ ))
       $\triangleright$   $\{x \mid isError\ x\}$ 
      % Graph destError) $\Uparrow$ );

```

SML

```

(* *** Goal "2.2.1.1" *** *)
a(POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[ran_def,rel_combine_def,r_9_r_thm,inv_rel_def,>_thm,rel_ext_clauses,graph_thm]);
a(rewrite_tac[sets_ext_clauses]);
a(REPEAT  $\forall$ _tac THEN  $\Leftrightarrow$ _T strip_asm_tac);
(* *** Goal "2.2.1.1.1" *** *)
a( $\exists$ _tac $\lceil$ # l' + 1 $\rceil$  THEN rewrite_tac[]);
a( $\exists$ _tac $\lceil$ z $\rceil$  THEN asm_rewrite_tac[]);
a( $\exists$ _tac $\lceil$ (Fst z',last) $\rceil$  THEN rewrite_tac[]);
a(DROP_NTH_ASM_T 2 ante_tac);

```

SML

```

a(LEMMA_T $\lceil$ z' = (Fst z',Snd z') $\rceil$  pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],pure_asm_rewrite_tac[] THEN  $\Rightarrow$ _tac]);
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[] THEN  $\Rightarrow$ _tac);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ z'' $\rceil$ , $\lceil$ Fst z' $\rceil$ , $\lceil$ Snd u $\rceil$ , $\lceil$ c $\rceil$ , $\lceil$ t2 $\rceil$  dom_ $\cup$ _ran_lemma));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ c $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ t2 $\rceil$ , $\lceil$ Fst z' $\rceil$  isError_ $\Leftrightarrow$ _updateRow_lemma));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ c $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ t2 $\rceil$ , $\lceil$ Fst z' $\rceil$  isError_updateRow_lemma));
a(asm_rewrite_tac[]);
a( $\exists$ _tac $\lceil$ z'' $\rceil$  THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ l' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$  size_squash_id_dom_thm));

```

SML

```

(* *** Goal "2.2.1.1.2" *** *)
a( $\exists$ _tac $\lceil$ # l + 1 $\rceil$  THEN rewrite_tac[]);
a( $\exists$ _tac $\lceil$ z $\rceil$  THEN asm_rewrite_tac[]);
a( $\exists$ _tac $\lceil$ (Fst z',last') $\rceil$  THEN rewrite_tac[]);
a(DROP_NTH_ASM_T 2 ante_tac);
a(LEMMA_T $\lceil$ z' = (Fst z',Snd z') $\rceil$  pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],pure_asm_rewrite_tac[] THEN  $\Rightarrow$ _tac]);
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[] THEN  $\Rightarrow$ _tac);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ z'' $\rceil$ , $\lceil$ Fst z' $\rceil$ , $\lceil$ Snd u $\rceil$ , $\lceil$ c $\rceil$ , $\lceil$ t2 $\rceil$  dom_ $\cup$ _ran_lemma));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ c $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ t2 $\rceil$ , $\lceil$ Fst z' $\rceil$  isError_ $\Leftrightarrow$ _updateRow_lemma));
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ c $\rceil$ , $\lceil$ last' $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ t2 $\rceil$ , $\lceil$ Fst z' $\rceil$  isError_updateRow_lemma));
a(asm_rewrite_tac[]);
a( $\exists$ _tac $\lceil$ z'' $\rceil$  THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_ $\forall$ _elim $\lceil$ l $\rceil$ , $\lceil$ l' $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$  size_squash_id_dom_thm));
a(asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.1.2" *** *)
a(lemma_tac (Dom (((RelCombine ({(#
    (Squash
      (Id
        (Dom
          (ListRel l
            ▷ {r
              |c
                dominates R_exist r}}))))
          + 1, # l + 1)}~
        % Snd u)
    {(# l + 1, last')}
    % Graph (updateRow c (TS_class t2)))
  ▷ {x|isError x}
  % Graph destError) = {# l + 1}
^ Dom(((RelCombine ({(#
    (Squash
      (Id
        (Dom
          (ListRel l'
            ▷ {r
              |c
                dominates R_exist r}}))))
          + 1, # l' + 1)}~
        % Snd u)
    {(# l' + 1, last')}
    % Graph (updateRow c (TS_class t2)))
  ▷ {x|isError x}
  % Graph destError) = {# l' + 1} ∨

```


SML

```

(* *** Goal "2.2.1.2.1" *** *)
a(POP_ASM_T (fn _ => id_tac) THEN POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[rel_combine_def,inv_rel_def,sets_ext_clauses,r_9-r_thm,
  ▷_thm,dom_thm,graph_thm] THEN REPEAT strip_tac);
(* *** Goal "2.2.1.2.1.1" *** *)
a(swap_nth_asm_concl_tac 7 THEN REPEAT strip_tac);
a(∃_tacΓ y∇ THEN asm_rewrite_tac[]);
a(∃_tacΓ z∇ THEN asm_rewrite_tac[]);
a(∃_tacΓ (Fst z',last')∇ THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_∇_elimΓ l∇,l'∇,{r|c dominates R_exist r}∇] size_squash_id_dom_thm));
a(strip_asm_tac(list_∇_elimΓ z''∇,Fst z'∇,Snd u∇,c∇,t2∇] dom_∪_ran_lemma));
a(DROP_NTH_ASM_T 4 ante_tac);
a(LEMMA_TΓ z' = (Fst z',Snd z')∇ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],pure_asm_rewrite_tac[] THEN ⇒_tac]);
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[] THEN ⇒_tac);
a(strip_asm_tac(list_∇_elimΓ c∇,last'∇,last∇,t2∇,Fst z'∇] isError_↔_updateRow_lemma));
a(strip_asm_tac(list_∇_elimΓ c∇,last'∇,last∇,t2∇,Fst z'∇] isError_updateRow_lemma));
a(∃_tacΓ z''∇ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.1.2.1.2" *** *)
a(swap_nth_asm_concl_tac 7 THEN REPEAT strip_tac);
a(∃_tacΓ y∇ THEN asm_rewrite_tac[]);
a(∃_tacΓ z∇ THEN asm_rewrite_tac[]);
a(∃_tacΓ (Fst z',last')∇ THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_∇_elimΓ l∇,l'∇,{r|c dominates R_exist r}∇] size_squash_id_dom_thm));
a(strip_asm_tac(list_∇_elimΓ z''∇,Fst z'∇,Snd u∇,c∇,t2∇] dom_∪_ran_lemma));
a(DROP_NTH_ASM_T 4 ante_tac);
a(LEMMA_TΓ z' = (Fst z',Snd z')∇ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],pure_asm_rewrite_tac[] THEN ⇒_tac]);
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[] THEN ⇒_tac);
a(strip_asm_tac(list_∇_elimΓ c∇,last'∇,last∇,t2∇,Fst z'∇] isError_↔_updateRow_lemma));
a(strip_asm_tac(list_∇_elimΓ c∇,last'∇,last∇,t2∇,Fst z'∇] isError_updateRow_lemma));
a(asm_rewrite_tac[]);
a(∃_tacΓ z''∇ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.1.2.1.3" *** *)
a(swap_nth_asm_concl_tac 7 THEN REPEAT strip_tac);
a(∃_tacΓ yΓ THEN asm_rewrite_tac[]);
a(∃_tacΓ zΓ THEN asm_rewrite_tac[]);
a(∃_tacΓ (Fst z', last)Γ THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_∇_elimΓ lΓ, l'Γ, {r | c dominates R_exist r}Γ size_squash_id_dom_thm));
a(strip_asm_tac(list_∇_elimΓ z''Γ, Fst z'Γ, Snd uΓ, cΓ, t2Γ dom_∪_ran_lemma));
a(DROP_NTH_ASM_T 4 ante_tac);
a(LEMMA_TΓ z' = (Fst z', Snd z')Γ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], pure_asm_rewrite_tac[] THEN ⇒_tac]);
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[] THEN ⇒_tac);
a(strip_asm_tac(list_∇_elimΓ cΓ, last'Γ, lastΓ, t2Γ, Fst z'Γ isError_↔_updateRow_lemma));
a(strip_asm_tac(list_∇_elimΓ cΓ, last'Γ, lastΓ, t2Γ, Fst z'Γ isError_updateRow_lemma));
a(asm_rewrite_tac[]);
a(∃_tacΓ z''Γ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.1.2.1.4" *** *)
a(swap_nth_asm_concl_tac 7 THEN REPEAT strip_tac);
a(∃_tacΓ yΓ THEN asm_rewrite_tac[]);
a(∃_tacΓ zΓ THEN asm_rewrite_tac[]);
a(∃_tacΓ (Fst z', last)Γ THEN asm_rewrite_tac[]);
a(strip_asm_tac(list_∇_elimΓ lΓ, l'Γ, {r | c dominates R_exist r}Γ size_squash_id_dom_thm));
a(strip_asm_tac(list_∇_elimΓ z''Γ, Fst z'Γ, Snd uΓ, cΓ, t2Γ dom_∪_ran_lemma));
a(DROP_NTH_ASM_T 4 ante_tac);
a(LEMMA_TΓ z' = (Fst z', Snd z')Γ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], pure_asm_rewrite_tac[] THEN ⇒_tac]);
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[] THEN ⇒_tac);
a(strip_asm_tac(list_∇_elimΓ cΓ, last'Γ, lastΓ, t2Γ, Fst z'Γ isError_↔_updateRow_lemma));
a(strip_asm_tac(list_∇_elimΓ cΓ, last'Γ, lastΓ, t2Γ, Fst z'Γ isError_updateRow_lemma));
a(∃_tacΓ z''Γ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.1.2.2" *** *)
a(strip_asm_tac(list_∇_elim[(((RelCombine ({(#
      (Squash
        (Id
          (Dom
            (ListRel l
              ▷ {r
                |c
          dominates R_exist r}}))))
      + 1, # l + 1)}~
    % Snd u)
  {(# l + 1, last')}
  % Graph (updateRow c (TS_class t2)))
▷ {x|isError x}
% Graph destError)∇,(((RelCombine ({(#
      (Squash
        (Id
          (Dom
            (ListRel l'
              ▷ {r
                |c
          dominates R_exist r}}))))
      + 1, # l' + 1)}~
    % Snd u)
  {(# l' + 1, last')}
  % Graph (updateRow c (TS_class t2)))
▷ {x|isError x}
% Graph destError)∇,[# l + 1∇, # l' + 1∇]squash_single_thm));

```

SML

```

a(fc_tac[fin_lemma1] THEN fc_tac[fin_lemma2] THEN bc_tac[squash_∪_thm]
  THEN asm_rewrite_tac[squash_doms_lemma]);
(* *** Goal "2.2.1.2.3" *** *)
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN rewrite_tac[dom_null_thm]
  THEN REPEAT ⇒_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2.2" *** *)
a(DROP_NTH_ASM_T 3(ante_tac o  $\forall$ _elim $\Gamma$ last' $\Uparrow$ ) THEN
  asm_rewrite_tac[map_ $\wedge$ _thm,size_list_rel_ $\wedge$ _▷_thm]);
a( $\Rightarrow$ _T asm_tac THEN REPEAT  $\Rightarrow$ _tac);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
a( $\Rightarrow$ _T rewrite_thm_tac);
a(asm_rewrite_tac[list_rel_ $\wedge$ _▷_thm]
  THEN rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine_ $\cup$ _thm]);
a(lemma_tac $\Gamma$ Dom((Squash (Id (Dom
  (ListRel l'
    ▷ {r|c dominates R_exist r}))))~
  % Snd u)  $\cap$  Dom{(# l' + 1, last)}={}) $\Uparrow$ );
(* *** Goal "2.2.2.1" *** *)
a(rewrite_tac[enumerate_def,r_ $\%$ _r_thm,inv_rel_def,dom_def,id_def,squash_def,
  list_rel_def,▷_thm,◁_thm,dot_dot_def]);
a(rewrite_tac[ $\cap$ _def,sets_ext_clauses] THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2" *** *)
a(strip_asm_tac(list_ $\forall$ _elim $\Gamma$ ((Squash
  (Id (Dom (ListRel l' ▷ {r|c dominates R_exist r}))))~
  % Snd u) $\Uparrow$ , $\Gamma$ {(# l' + 1, last)} $\Uparrow$ ]rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);

```

SML

```

(* *** Goal "2.2.3" *** *)
a(DROP_NTH_ASM_T 4(ante_tac o  $\forall$ _elim $\Gamma$  l'  $\wedge$  [last] $\Uparrow$ ) THEN
  asm_rewrite_tac[map_ $\wedge$ _thm,size_list_rel_ $\wedge$ _▷_thm]);
a( $\Rightarrow$ _T asm_tac THEN REPEAT  $\Rightarrow$ _tac);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
a( $\Rightarrow$ _T (rewrite_thm_tac o eq_sym_rule));
a(asm_rewrite_tac[list_rel_ $\wedge$ _▷_thm]
  THEN rewrite_tac[list_rel_ $\wedge$ _singleton_thm,rel_combine_ $\cup$ _thm]);
a(lemma_tac $\Gamma$  Dom((Squash (Id (Dom (ListRel l
  ▷ {r|c dominates R_exist r})))) $\sim$ 
  § Snd u)  $\cap$  Dom{(# l + 1, last')}={}) $\Uparrow$ );
(* *** Goal "2.2.3.1" *** *)
a(rewrite_tac[enumerate_def,r_§_r_thm,inv_rel_def,dom_def,id_def,squash_def,
  list_rel_def,▷_thm,◁_thm,dot_dot_def]);
a(rewrite_tac[ $\cap$ _def,sets_ext_clauses] THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.3.2" *** *)
a(strip_asm_tac(list_ $\forall$ _elim $\Gamma$ ((Squash
  (Id (Dom (ListRel l ▷ {r|c dominates R_exist r})))) $\sim$ 
  § Snd u) $\Uparrow$ , $\Gamma$ {(# l + 1, last')} $\Uparrow$ ]rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);

```

SML

```

(* *** Goal "2.2.4" *** *)
a(asm_rewrite_tac[list_rel_⊆_▷_thm]
  THEN rewrite_tac[list_rel_⊆_singleton_thm,rel_combine_∪_thm]);
a(lemma_tac[∀l (last:Row) • Dom((Squash (Id (Dom (ListRel l
  ▷ {r|c dominates R_exist r}))))~
  § Snd u) ∩ Dom{(# l + 1, last)}={})]);
(* *** Goal "2.2.4.1" *** *)
a(REPEAT ∀_tac);
a(rewrite_tac[enumerate_def,r_§_r_thm,inv_rel_def,dom_def,id_def,squash_def,
  list_rel_def,▷_thm,◁_thm,dot_dot_def]);
a(rewrite_tac[∩_def,sets_ext_clauses] THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.4.2" *** *)
a(TOP_ASM_T(strip_asm_tac o list_∀_elim[⊆_l',⊆_last]));
a(strip_asm_tac(list_∀_elim[⊆_((Squash
  (Id (Dom (ListRel l' ▷ {r|c dominates R_exist r}))))~
  § Snd u)⊆,⊆_{(# l' + 1, last)}⊆]rel_combine_null_thm2));
a(POP_ASM_T rewrite_thm_tac);
a(POP_ASM_T (fn _ => id_tac)
  THEN POP_ASM_T(strip_asm_tac o list_∀_elim[⊆_l',⊆_last']));
a(strip_asm_tac(list_∀_elim[⊆_((Squash
  (Id (Dom (ListRel l ▷ {r|c dominates R_exist r}))))~
  § Snd u)⊆,⊆_{(# l + 1, last')}⊆]rel_combine_null_thm2));
a(asm_rewrite_tac[]);
val cleanTable_updateQuery_lemma = save_pop_thm"cleanTable_updateQuery_lemma";

```

HOL output

```

cleanTable_updateQuery_lemma =
⊢ ∀ c t1 t2 s1 s2 u
  • c dominates TS_class t1
    ∧ c dominates TS_class t2
    ∧ cleanTable c t1 = cleanTable c t2
  ⇒ Snd (updateQuery (c, u, s1, t1))
    = Snd (updateQuery (c, u, s2, t2))

```

SML

```

push_goal([],Γ∀ c s1 s2 • hide (c, s1) = hide (c, s2) ⇒
(∀ i • tabExists c i (repState s1)
⇒ ((cleanTable c (getTable i (repState s1))
=
cleanTable c (getTable i (repState s2)))
∧
(c dominates TS_class (getTable i (repState s1))
⇔
c dominates TS_class (getTable i (repState s2))))Γ);

```

SML

```

a(REPEAT ∀_tac THEN ⇒_tac THEN ∀_tac THEN ⇒_tac);
a(strip_asm_tac(list_∀_elim[ΓcΓ,Γs1Γ,Γs2Γ]tabExists_lemma));
a(POP_ASM_T (strip_asm_tac o ∀_elimΓiΓ));
a(DROP_ASMS_T (MAP_EVERY ante_tac) THEN rewrite_tac[dom_def,tabExists_def,
hideR_def,hide_eq_lemma,▷_thm,r_?_r_thm,rel_ext_clauses,graph_thm]
THEN REPEAT ⇒_tac);
a(rewrite_tac[getTable_def]);
a(strip_asm_tac (rewrite_rule[get_specΓisStateΓ,get_specΓStateSΓ,↔_def,∩_def]
(∀_elimΓs1ΓisState_lemma)));
a(strip_asm_tac (rewrite_rule[get_specΓisStateΓ,get_specΓStateSΓ,↔_def,∩_def]
(∀_elimΓs2ΓisState_lemma)));

```

SML

```

a(DROP_NTH_ASM_T 4 ante_tac THEN DROP_NTH_ASM_T 2 ante_tac THEN
rewrite_tac[↔_def,get_specΓIdeLΓ,get_specΓDirectorySΓ,∩_def,×_def,
get_specΓUniverseΓ,dom_def,▷_thm,r_?_r_thm,graph_thm,rel_ext_clauses,
get_specΓ$PΓ]
THEN strip_tac THEN strip_tac);
a(strip_asm_tac(list_∀_elim[ΓrepState s1Γ,ΓFront iΓ,ΓyΓ]at_thm1));
a(strip_asm_tac(list_∀_elim[ΓrepState s2Γ,ΓFront iΓ,Γy''Γ]at_thm1));
a(LIST_DROP_NTH_ASM_T [7,8,9,11,12,13](MAP_EVERY ante_tac)
THEN asm_rewrite_tac[] THEN REPEAT ⇒_tac);
a(DROP_NTH_ASM_T 10 (strip_asm_tac o list_∀_elim[ΓFront iΓ,Γy''Γ]));
a(DROP_NTH_ASM_T 10 (strip_asm_tac o list_∀_elim[ΓFront iΓ,ΓyΓ]));
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN rewrite_tac[↔_def,∩_def]
THEN REPEAT ⇒_tac);

```

SML

```

a(strip_asm_tac(list_∀_elim[Dir_tables y⊢, Last i⊢, y'⊢]at_thm1));
a(strip_asm_tac(list_∀_elim[Dir_tables y''⊢, Last i⊢, y'''⊢]at_thm1));
a(asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 19 (asm_tac o list_∀_elim[Front i⊢, cleanDirectory c y⊢]));
a(LEMMA_T⊢∃ z
  • (c dominates Dir_exist z ∧ (Front i, z) ∈ repState s1)
    ∧ cleanDirectory c y = cleanDirectory c z⊢asm_tac);
(* *** Goal "1" *** *)
a(∃_tac⊢y⊢ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN ⇒_tac);
a(lemma_tac⊢z = y''⊢);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 18(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∀_elim[Front i⊢, z⊢, y''⊢]));
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac);
a(asm_rewrite_tac[cleanDirectory_def, dir_components, get_spec⊢MkDirectory⊢] THEN strip_tac);
a(DROP_NTH_ASM_T 3 ante_tac THEN rewrite_tac[rel_ext_clauses, graph_thm, r_§-r_thm]
  THEN ⇒_T (asm_tac o list_∀_elim[Last i⊢, cleanTable c y'⊢]));
a(LEMMA_T⊢∃ z • (Last i, z) ∈ Dir_tables y ∧ cleanTable c y' = cleanTable c z⊢asm_tac);

```

SML

```

(* *** Goal "2.2.1" *** *)
a(∃_tac⊢y'⊢ THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN ⇒_tac);
a(lemma_tac⊢z' = y'''⊢);
(* *** Goal "2.2.2.1" *** *)
a(DROP_NTH_ASM_T 11(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∀_elim[Last i⊢, z'⊢, y'''⊢]));

```

SML

```

(* *** Goal "2.2.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac
  THEN =>_tac THEN asm_rewrite_tac[] THEN POP_ASM_T ante_tac);
a(rewrite_tac[cleanTable_def]);
a(cases_tacΓ c dominates TS_class y'''∇ THEN cases_tacΓ c dominates TS_class y'∇
  THEN asm_rewrite_tac[tab_components,get_specΓ MkTableSpec∇]
  THEN strip_tac THEN_TRY asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.1" *** *)
a(REPEAT strip_tac THEN contr_tac);
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2" *** *)
a(REPEAT strip_tac THEN contr_tac);
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
val tabExists_cleanTable_lemma = save_pop_thm"tabExists_cleanTable_lemma";

```

HOL output

```

tabExists_cleanTable_lemma = ⊢ ∀ c s1 s2
  • hide (c, s1) = hide (c, s2)
    ⇒ (∀ i
      • tabExists c i (repState s1)
        ⇒ (cleanTable c (getTable i (repState s1))
            = cleanTable c (getTable i (repState s2))
            ∧ (c dominates TS_class (getTable i (repState s1))
                ⇔ c dominates TS_class (getTable i (repState s2))))))

```

5.2.2 Proof of Conjunct 3

SML

```

push_goal([],Γ∀ c s1 s2 e
  • hide (c, s1) = hide (c, s2)
    ⇒ Snd (updateState (c, e, s1)) = Snd (updateState (c, e, s2))∇);
a(REPEAT strip_tac);
a(LEMMA_TΓ e = (Fst e, Snd e)∇ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],rewrite_tac[updateState_def,updateStateR_def]]);
a(lemma_tacΓ (tabExists c (tabFromEffect (Fst e)) (repState s1)
  ⇒ tabExists c (tabFromEffect (Fst e)) (repState s2))
  ∧ (tabExists c (tabFromEffect (Fst e)) (repState s2)
  ⇒ tabExists c (tabFromEffect (Fst e)) (repState s1))∇);

```

SML

```

| (* *** Goal "1" *** *)
| a(strip_asm_tac(list_∇_elim[⌈ c ⌋, ⌈ s1 ⌋, ⌈ s2 ⌋]tabExists_lemma) THEN asm_rewrite_tac[]);
| (* *** Goal "2" *** *)
| (* invalid tables in s1 and s2 *)
| a(strip_asm_tac(∇_elim ⌈ Fst e ⌋ query_type) THEN asm_rewrite_tac[]
|   THEN cases_tac⌈ ¬ Snd e = [] ⌋ THEN asm_rewrite_tac[]);
| (* *** Goal "3" *** *)
| (* valid tables in s1 and s2 *)
| a(strip_asm_tac (list_∇_elim[⌈ c ⌋, ⌈ s1 ⌋, ⌈ s2 ⌋]tabExists_cleanTable_lemma));
| a(POP_ASM_T (strip_asm_tac o ∇_elim⌈ tabFromEffect (Fst e) ⌋));
| (* *** Goal "3.1" *** *)
| (* table classes not dominated by c *)
| a(EVERY[strip_asm_tac(∇_elim ⌈ Fst e ⌋ query_type),
|   asm_rewrite_tac[],
|   cases_tac⌈ ¬ Snd e = [] ⌋,
|   asm_rewrite_tac[]]);

```

SML

```

| (* *** Goal "3.2" *** *)
| (* table classes dominated by c *)
| a(EVERY[strip_asm_tac(∇_elim ⌈ Fst e ⌋ query_type),
|   asm_rewrite_tac[],
|   cases_tac⌈ ¬ Snd e = [] ⌋,
|   asm_rewrite_tac[]]);

```

SML

```

| (* Select and Delete automatically proven *)
| (* *** Goal "3.2.1" *** *)
| (* *** Insert *** *)
| a(strip_asm_tac(list_∇_elim[⌈ c ⌋, ⌈ getTable (tabFromEffect (Fst e)) (repState s1) ⌋,
|   ⌈ getTable (tabFromEffect (Fst e)) (repState s2) ⌋,
|   ⌈ repState s1 ⌋, ⌈ repState s2 ⌋, ⌈ destInsert (Fst e) ⌋]cleanTable_insertQuery_lemma));

```

SML

```

| (* *** Goal "3.2.2" *** *)
| (* *** Update *** *)
| a(strip_asm_tac(list_∇_elim[⌈ c ⌋, ⌈ getTable (tabFromEffect (Fst e)) (repState s1) ⌋,
|   ⌈ getTable (tabFromEffect (Fst e)) (repState s2) ⌋,
|   ⌈ repState s1 ⌋, ⌈ repState s2 ⌋, ⌈ destUpdate (Fst e) ⌋]cleanTable_updateQuery_lemma));
| val conjunct3 = save_pop_thm "conjunct3";

```

HOL output

```
| conjunct3 = ⊢ ∀ c s1 s2 e  
| • hide (c, s1) = hide (c, s2)  
|   ⇒ Snd (updateState (c, e, s1)) = Snd (updateState (c, e, s2))
```

6 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

SML

```
| pop_pc();
```

7 THE THEORY fef011

7.1 Parents

fef010

7.2 Children

fef012

7.3 Theorems

destInsert_consistent
destDelete_consistent
destUpdate_consistent
destSelect_consistent

$\vdash \text{Consistent}$
(λ
 (*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

$\vdash \text{Consistent}$
(λ *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

$\vdash \text{Consistent}$
(λ
 (*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*)

$$\begin{aligned} & \wedge (isVal' ve \Leftrightarrow (\exists v_1 \bullet ve = giveVal' v_1)) \\ & \wedge (isError' ve \\ & \Leftrightarrow (\exists e_1 \bullet ve = giveError' e_1))) \end{aligned}$$

giveVal_eq_thm

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

giveError_eq_thm

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

 \neg isError_giveVal_thm

$$\vdash \forall v \bullet \neg isError (giveVal v)$$

 \neg isVal_giveError_thm

$$\vdash \forall e \bullet \neg isVal (giveError e)$$

query_type

$$\begin{aligned} \vdash \forall q & \\ & \bullet isInsert q \\ & \quad \wedge \neg (isDelete q \vee isUpdate q \vee isSelect q) \\ & \quad \vee isDelete q \\ & \quad \wedge \neg (isInsert q \vee isUpdate q \vee isSelect q) \\ & \quad \vee isUpdate q \\ & \quad \wedge \neg (isInsert q \vee isDelete q \vee isSelect q) \\ & \quad \vee isSelect q \\ & \quad \wedge \neg (isInsert q \vee isDelete q \vee isUpdate q) \end{aligned}$$

update_type

$$\begin{aligned} \vdash \forall u & \\ & \bullet isItem u \wedge \neg (isClass u \vee isData u) \\ & \quad \vee isClass u \wedge \neg (isItem u \vee isData u) \\ & \quad \vee isData u \wedge \neg (isItem u \vee isClass u) \end{aligned}$$

val_or_error_type

$$\vdash \forall v \bullet isVal v \wedge \neg isError v \vee isError v \wedge \neg isVal v$$

conjunct4

$$\vdash \forall c s e \bullet Fst (Snd (updateState (c, e, s))) = c$$

isState_lemma3

$$\vdash \forall c s \bullet isState (hideR (c, repState s))$$

hide_eq_lemma

$$\begin{aligned} \vdash \forall c s_1 s_2 & \\ & \bullet hide (c, s_1) = hide (c, s_2) \\ & \quad \Leftrightarrow hideR (c, repState s_1) \\ & \quad = hideR (c, repState s_2) \end{aligned}$$

tabExists_lemma

$$\begin{aligned} \vdash \forall c s_1 s_2 & \\ & \bullet hide (c, s_1) = hide (c, s_2) \\ & \quad \Rightarrow (\forall i \\ & \quad \bullet tabExists c i (repState s_1) \\ & \quad \Leftrightarrow tabExists c i (repState s_2)) \end{aligned}$$

cleanTable_insertQuery_lemma

$$\begin{aligned} \vdash \forall c t_1 t_2 s_1 s_2 i & \\ & \bullet c \text{ dominates } TS_class t_1 \\ & \quad \wedge c \text{ dominates } TS_class t_2 \\ & \quad \wedge cleanTable c t_1 = cleanTable c t_2 \\ & \quad \Rightarrow Snd (insertQuery (c, i, s_1, t_1)) \\ & \quad = Snd (insertQuery (c, i, s_2, t_2)) \end{aligned}$$

updateRow_lemma

- $$\vdash \forall c t_1 t_2$$
- $\text{cleanTable } c t_1 = \text{cleanTable } c t_2$
 $\Rightarrow c \text{ dominates } TS_class t_1$
 $\wedge c \text{ dominates } TS_class t_2$
 $\Rightarrow \text{updateRow } c (TS_class t_1)$
 $= \text{updateRow } c (TS_class t_2)$

isError_updateField_lemma

- $$\vdash \forall c tc d_1 d_2 u$$
- $\text{isError } (\text{updateField } c tc (u, d_1))$
 $\wedge \text{replaceData } c d_1 = \text{replaceData } c d_2$
 $\Rightarrow \text{updateField } c tc (u, d_1)$
 $= \text{updateField } c tc (u, d_2)$

isError_updateRow_lemma

- $$\vdash \forall c r_1 r_2 t u$$
- $\text{isError } (\text{updateRow } c (TS_class t) (u, r_1))$
 $\wedge \text{isError } (\text{updateRow } c (TS_class t) (u, r_2))$
 $\wedge \text{cleanRow } c (\text{Snd } (\text{cleanColCons } c t)) r_1$
 $= \text{cleanRow } c (\text{Snd } (\text{cleanColCons } c t)) r_2$
 $\wedge \text{Dom } u$
 $\subseteq \{n\}$
 $|\exists c' \bullet c' \in \text{visibleCols } c t \wedge CS_posn c' = n\}$
 $\Rightarrow \text{updateRow } c (TS_class t) (u, r_1)$
 $= \text{updateRow } c (TS_class t) (u, r_2)$

isVal_updateRow_lemma

- $$\vdash \forall c r_1 r_2 t u$$
- $\text{isVal } (\text{updateRow } c (TS_class t) (u, r_1))$
 $\wedge \text{cleanRow } c (\text{Snd } (\text{cleanColCons } c t)) r_1$
 $= \text{cleanRow } c (\text{Snd } (\text{cleanColCons } c t)) r_2$
 $\wedge \text{Dom } u$
 $\subseteq \{n\}$
 $|\exists c' \bullet c' \in \text{visibleCols } c t \wedge CS_posn c' = n\}$
 $\Rightarrow \text{isVal } (\text{updateRow } c (TS_class t) (u, r_2))$

isError_⇔_updateRow_lemma

- $$\vdash \forall c r_1 r_2 t u$$
- $\text{cleanRow } c (\text{Snd } (\text{cleanColCons } c t)) r_1$
 $= \text{cleanRow } c (\text{Snd } (\text{cleanColCons } c t)) r_2$
 $\wedge \text{Dom } u$
 $\subseteq \{n\}$
 $|\exists c' \bullet c' \in \text{visibleCols } c t \wedge CS_posn c' = n\}$
 $\Rightarrow (\text{isError } (\text{updateRow } c (TS_class t) (u, r_1)))$
 $\Leftrightarrow \text{isError } (\text{updateRow } c (TS_class t) (u, r_2))$

dom_∪_ran_lemma

- $$\vdash \forall x y u c t$$
- $(x, y) \in u$
 $\wedge \text{Dom } (\cup (\text{Ran } u))$
 $\subseteq \{n\}$
 $|\exists c'$
 - $c' \in \text{Snd } (\text{cleanColCons } c t)$

$$\begin{aligned} & \wedge CS_posn\ c' = n\} \\ \Rightarrow & Dom\ y \\ & \subseteq \{n\} \\ & |\exists\ c'\bullet\ c' \in visibleCols\ c\ t \wedge CS_posn\ c' = n\} \end{aligned}$$

doms_null_lemma1

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

doms_null_lemma2

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

squash_doms_lemma

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

cleanRows_size_lemma

$$\begin{aligned}
& \vdash \forall c \ t_1 \ t_2 \\
& \bullet \text{ cleanRows} \\
& \quad c \\
& \quad (Snd \ (cleanColCons \ c \ t_2)) \\
& \quad (TS_rows \ t_1) \\
& = \text{ cleanRows} \\
& \quad c \\
& \quad (Snd \ (cleanColCons \ c \ t_2)) \\
& \quad (TS_rows \ t_2) \\
& \Rightarrow \# \\
& \quad (ListRel \ (TS_rows \ t_1) \\
& \quad \triangleright \ {r|c \ \text{dominates} \ R_exist \ r}) \\
& = \# \\
& \quad (ListRel \ (TS_rows \ t_2) \\
& \quad \triangleright \ {r|c \ \text{dominates} \ R_exist \ r})
\end{aligned}$$

cleanRows_errors_or_vals_lemma

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

fun_updateRow_thm

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

fun_destError_thm

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

fin_lemma1 $\vdash \forall l r c t u$

- $u \in \text{Functional}$

$$\Rightarrow ((\text{RelCombine} (\text{Squash} (\text{Id} (\text{Dom} (\text{ListRel } l \triangleright \{r \mid c \text{ dominates } R_exist\ r\}))) \sim \text{; } u) (\text{ListRel } l) \text{; } \text{Graph} (\text{updateRow } c \text{ (TS_class } t))) \triangleright \{x \mid \text{isError } x\}) \text{; } \text{Graph } \text{destError} \in \text{Finite}$$

fin_lemma2 $\vdash \forall l \text{ last } r c t u$

- $u \in \text{Functional}$

$$\Rightarrow ((\text{RelCombine} (\{(\# (\text{Squash} (\text{Id} (\text{Dom} (\text{ListRel } l \triangleright \{r \mid c \text{ dominates } R_exist\ r\}))) + 1, \# l + 1)\} \sim \text{; } u) \{(\# l + 1, \text{last})\} \text{; } \text{Graph} (\text{updateRow } c \text{ (TS_class } t))) \triangleright \{x \mid \text{isError } x\}) \text{; } \text{Graph } \text{destError} \in \text{Finite}$$

cleanTable_updateQuery_lemma

$$\vdash \forall c t_1 t_2 s_1 s_2 u$$

- $c \text{ dominates } \text{TS_class } t_1$
- $\wedge c \text{ dominates } \text{TS_class } t_2$
- $\wedge \text{cleanTable } c t_1 = \text{cleanTable } c t_2$

$$\Rightarrow \text{Snd} (\text{updateQuery } (c, u, s_1, t_1)) = \text{Snd} (\text{updateQuery } (c, u, s_2, t_2))$$

tabExists_cleanTable_lemma

$$\vdash \forall c s_1 s_2$$

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

$$\Rightarrow (\forall i$$

- $\text{tabExists } c i (\text{repState } s_1)$

$$\Rightarrow \text{cleanTable } c (\text{getTable } i (\text{repState } s_1)) = \text{cleanTable } c (\text{getTable } i (\text{repState } s_2))$$

$$\begin{aligned} & \wedge (c \\ & \quad \text{dominates } TS_class \\ & \quad \text{(getTable } i \text{ (repState } s_1)) \\ & \Leftrightarrow c \\ & \quad \text{dominates } TS_class \\ & \quad \text{(getTable } i \text{ (repState } s_2)))) \\ \mathbf{conjunct3} \quad & \vdash \forall c \ s_1 \ s_2 \ e \\ & \bullet \text{hide } (c, s_1) = \text{hide } (c, s_2) \\ & \quad \Rightarrow \text{Snd } (\text{updateState } (c, e, s_1)) \\ & \quad = \text{Snd } (\text{updateState } (c, e, s_2)) \end{aligned}$$

8 INDEX

<i>changeSpec_def</i>	6	<i>visibleCols_def</i>	6
<i>cleanRows_errors_or_vals_lemma</i>	37	\neg <i>isError_giveVal_thm</i>	7
<i>cleanRows_size_lemma</i>	29	\neg <i>isVal_giveError_thm</i>	7
<i>cleanTable_insertQuery_lemma</i>	17		
<i>cleanTable_updateQuery_lemma</i>	54		
<i>colDefaults_def</i>	6		
<i>conjunct3</i>	58		
<i>conjunct4</i>	13		
<i>destInsert_def</i>	5		
<i>doms_null_lemma1</i>	25		
<i>doms_null_lemma2</i>	25		
<i>dom_U_ran_lemma</i>	24		
<i>fef011</i>	4		
<i>fin_lemma1</i>	39		
<i>fin_lemma2</i>	40		
<i>fun_destError_thm</i>	38		
<i>fun_updateRow_thm</i>	38		
<i>getTable_def</i>	6		
<i>giveError_eq_thm</i>	7		
<i>giveVal_def</i>	6		
<i>giveVal_eq_thm</i>	7		
<i>hide_eq_lemma</i>	14		
<i>InsertEffect_def</i>	5		
<i>insertQuery_def</i>	6		
<i>isDelete_def</i>	6		
<i>isError_def</i>	7		
<i>isError_updateField_lemma</i>	18		
<i>isError_updateRow_lemma</i>	21		
<i>isError_\Leftrightarrow_updateRow_lemma</i>	24		
<i>isInsert_def</i>	6		
<i>isSelect_def</i>	6		
<i>isState_lemma3</i>	13		
<i>isUpdate_def</i>	6		
<i>isVal_def</i>	7		
<i>isVal_updateRow_lemma</i>	23		
<i>query_type</i>	11		
<i>replaceRows_def</i>	6		
<i>revealRow_def</i>	6		
<i>squash_doms_lemma</i>	26		
<i>tabExists_cleanTable_lemma</i>	57		
<i>tabExists_def</i>	6		
<i>tabExists_lemma</i>	16		
<i>tabFromEffect_def</i>	5		
<i>updateField_def</i>	6		
<i>updateQuery_def</i>	6		
<i>updateRow_def</i>	6		
<i>updateRow_lemma</i>	18		
<i>updateStateR_def</i>	6		
<i>updateState_def</i>	6		
<i>update_type</i>	12		
<i>val_or_error_type</i>	12		