

Project: DRA FRONT END FILTER PROJECT

Title: Proof of Security (IIc)

Ref: DS/FMU/FEF/012

Issue: Revision : 2.6

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 the first conjunct 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	3
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 Constant Definition	5
4	AUXILIARY THEOREMS	5
5	PROOF OF SECURITY OF CRITICAL COMPONENTS	19
5.1	Insert Lemmas	19
5.2	Delete Lemmas	24
5.3	Update Lemmas	40
5.4	Proof of Conjunct 1	55
6	CLOSING DOWN	56
7	THE THEORY fef012	57
7.1	Parents	57
7.2	Children	57
7.3	Theorems	57
8	INDEX	64

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.
- [4] DS/FMU/FEF/011. *Proof of Security (IIb)*. G.M. Prout, ICL Secure Systems, WIN01.
- [5] DS/FMU/FEF/013. *Proof of Security (IID)*. G.M. Prout, ICL Secure Systems, WIN01.

0.3 Changes History

Issue 2.3 Ported to version 0.3 ProofPower.

Issue 2.4 Now deletes theory if necessary.

Issue 2.5 Allowed for set difference being left-associative.

Issue 2.6 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document, together with [4] and [5], provides a formal proof that the components *hide* and *updateState* satisfy 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 $hide \in secureHide \wedge (hide, updateState) \in secureUpdate$

In this document, we give a proof that the first conjunct of *secureUpdate* holds for *hideR* and *updateStateR*:

| ? \vdash $\forall c_1 c_2 s e \bullet$
| $\neg hideR (c_2, repState s)$
| $= hideR (c_2, Fst (updateStateR (c_1, e, repState s)))$
| $\Rightarrow c_2 \text{ dominates } c_1$

2 PRELIMINARIES

The following ProofPower instructions set up the new theory *fef012*.

```
SML
|open_theory "fef011";
|(force_delete_theory "fef012" handle _ => ());
|new_theory "fef012";
|push_pc "hol";
|add_rw_thms [repState_absState_def] "wrk049a";
|set_merge_pcs["hol", "wrk049", "wrk049a", "pair1"] ;
```

3 CONSISTENCY PROOFS

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

SML

```

| push_consistency_goal  $\Gamma$  destItem  $\neg$ ;
| a( $\exists$ _tac  $\Gamma$  (OutL, OutL o OutR, OutR o OutR)  $\neg$ );
| a(rewrite_tac [get_spec  $\Gamma$  ItemUpdate  $\neg$ ]);
| save_consistency_thm  $\Gamma$  destItem  $\neg$  (pop_thm());
| val destItem_def = get_spec  $\Gamma$  destItem  $\neg$ ;

```

HOL output

```

| destItem_def =
|  $\vdash \forall i c d$ 
| • destItem (ItemUpdate i) = i
|    $\wedge$  destClass (ClassUpdate c) = c
|    $\wedge$  destData (DataUpdate d) = d

```

3.1 Retrieving the Remaining Constant Definition

SML

```

| val deleteQuery_def = conv_rule (MAP_C let_conv) (get_spec  $\Gamma$  deleteQuery  $\neg$ );

```

HOL output

```

| deleteQuery_def =
|  $\vdash \forall clear i ns e s ts$ 
| • deleteQuery (clear, (i, ns), s, ts)
|   = changeSpec
|     i
|     (replaceRows
|       ts
|       (Extract
|         (1 .. # (TS_rows ts)
|           \ revealRow clear ts Image ns
|              $\cap$  {i | R_exist (Nth (TS_rows ts) i) = clear})
|         (TS_rows ts))) s

```

4 AUXILIARY THEOREMS

First we simplify some of the constant defining theorems.

SML

```

| val destVal_def = all_ $\forall$ _intro (nth 2 (strip_ $\wedge$ _rule (all_ $\forall$ _elim giveVal_def)));
| val destError_def = all_ $\forall$ _intro (nth 3 (strip_ $\wedge$ _rule (all_ $\forall$ _elim giveVal_def)));

```

HOL output

```
|destVal_def = ⊢ ∀ v • destVal (giveVal v) = v
|destError_def = ⊢ ∀ e • destError (giveError e) = e
```

SML

```
|val ¬giveVal_eq_giveError_thm = save_thm("¬giveVal_eq_giveError_thm",
|    prove_rule[giveVal_def]⊢∀ v e • ¬(giveVal v = giveError e)⊢);
|val ¬giveError_eq_giveVal_thm = save_thm("¬giveError_eq_giveVal_thm",
|    prove_rule[giveVal_def]⊢∀ e v • ¬(giveError e = giveVal v)⊢);
```

HOL output

```
|¬giveVal_eq_giveError_thm = ⊢ ∀ v e • ¬ giveVal v = giveError e
|¬giveError_eq_giveVal_thm = ⊢ ∀ e v • ¬ giveError e = giveVal v
```

Now some results which will be used in the main proofs.

SML

```
|push_goal([],⊢∀ l (last:Row) s (u : ℕ ↔ (ℕ ↔ Update)) •
|    RelCombine((Squash(Id (Dom(ListRel l ▷ s))))~ § u)
|        (ListRel (l ^ [last]))
|    = RelCombine((Squash(Id (Dom(ListRel l ▷ s))))~ § u)
|        (ListRel l)⊢);
|a(rewrite_tac[rel_combine_def,rel_ext_clauses,list_rel_def,
|    dot_dot_def,length_^_one_thm]);
|a(REPEAT strip_tac);
|(* *** Goal "1" *** *)
|a(∃_tac⊢z⊢THEN asm_rewrite_tac[]);
|(* *** Goal "2" *** *)
|a(DROP_NTH_ASM_T 5(strip_asm_tac o rewrite_rule[enumerate_def,inv_rel_def,dom_def,
|    id_def,squash_def]));
|a(DROP_NTH_ASM_T 8 ante_tac THEN asm_rewrite_tac[]);
|(* *** Goal "3" *** *)
|a(DROP_NTH_ASM_T 5(strip_asm_tac o rewrite_rule[enumerate_def,inv_rel_def,dom_def,
|    id_def,squash_def]));
|a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∨_elim[⊢l⊢,⊢z'⊢,⊢last⊢]nth_^_thm1)));
|a(POP_ASM_T ante_tac THEN asm_rewrite_tac[] THEN ⇒_tac);
|(* *** Goal "4" *** *)
|a(∃_tac⊢z⊢THEN asm_rewrite_tac[]);
|(* *** Goal "5" *** *)
|a(fc_tac[≤_plus_one_thm]);
|(* *** Goal "6" *** *)
|a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∨_elim[⊢l⊢,⊢x⊢,⊢last⊢]nth_^_thm1)));
|a(asm_rewrite_tac[]);
|val rel_combine_one_lemma = save_pop_thm"rel_combine_one_lemma";
```

HOL output

```

rel_combine_one_lemma =
| ⊢ ∀ l last s u
| • RelCombine
|   ((Squash (Id (Dom (ListRel l ▷ s)))) ~ § u)
|   (ListRel (l ^ [last]))
| = RelCombine
|   ((Squash (Id (Dom (ListRel l ▷ s)))) ~ § u)
|   (ListRel l)

```

SML

```

push_goal([],⌈∀ l last c t us • ((RelCombine
    ((Squash
        (Id
            (Dom
                (ListRel (l ∧ [last])
                    ▷ {r|c dominates R_exist r}))))~
        % us)
        (ListRel (l ∧ [last]))
        % Graph (updateRow c (TS_class t)))
        ▷ {x|isError x}
        % Graph destError
    = {} ⇒ ((RelCombine
        ((Squash
            (Id
                (Dom
                    (ListRel l ▷ {r|c dominates R_exist r}))))~
            % us)
            (ListRel l)
            % Graph (updateRow c (TS_class t)))
            ▷ {x|isError x}
            % Graph destError
        = {}⌋);
a(REPEAT ∀_tac);
a(cases_tac⌈c dominates R_exist last⌋THEN asm_rewrite_tac
    [squash_∧_thm]);
(* *** Goal "1" *** *)
a(rewrite_tac[∪_null_thm]THEN strip_tac);
a(DROP_NTH_ASM_T 3 (fn _ => id_tac) THEN POP_ASM_T(fn _ => id_tac));
set_labelled_goal"2";
(* *** Goal "1 & 2" *** *)
a(POP_ASM_T(fn _ => id_tac) THEN strip_tac);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[rel_combine_one_lemma]));
val rel_combine_null_lemma = save_pop_thm"rel_combine_null_lemma";

```

HOL output

```

rel_combine_null_lemma =
| ⊢ ∀ l last c t us
| • ((RelCombine
|   ((Squash
|     (Id
|       (Dom
|         (ListRel (l ^ [last])
|           ▷ {r|c dominates R_exist r}))))~
|     % us)
|   (ListRel (l ^ [last]))
|   % Graph (updateRow c (TS_class t)))
|   ▷ {x|isError x})
|   % Graph destError
| = {}
| ⇒ ((RelCombine
|   ((Squash
|     (Id (Dom (ListRel l ▷ {r|c dominates R_exist r}))))~
|     % us)
|   (ListRel l)
|   % Graph (updateRow c (TS_class t)))
|   ▷ {x|isError x})
|   % Graph destError
| = {}

```

SML

```

push_goal([], ⌈∀ l s (u : ℕ ↔ (ℕ ↔ Update)) c t •
  Dom((RelCombine((Squash(Id(Dom
    (ListRel l ▷ {r|c dominates R_exist r}))))~ % u)
    (ListRel l)
    % Graph (updateRow c (TS_class t))) % Graph destVal)
  ⊆ Dom(ListRel l)⌋);
a(REPEAT ∀_tac);
a(rewrite_tac[enumerate_def, inv_rel_def, dom_def, id_def, squash_def,
  rel_combine_def, list_rel_def, dot_dot_def, sets_ext_clauses]);
a(rename_tac[]); (* to eliminate duplicate variable names *)
a(REPEAT strip_tac);
a(prove_∃_tac THEN asm_rewrite_tac[]);
val dom_rel_combine_null_⊆_lemma = save_pop_thm"dom_rel_combine_null_⊆_lemma";

```

HOL output

```

dom_rel_combine_null_⊆_lemma =
| ⊢ ∀ l s u c t
|   • Dom
|     ((RelCombine
|       ((Squash
|         (Id (Dom (ListRel l ▷ {r|c dominates R_exist r}))))~
|           § u)
|         (ListRel l)
|         § Graph (updateRow c (TS_class t)))
|         § Graph destVal)
|     ⊆ Dom (ListRel l)

```

SML

```

val destVal_fun_thm = save_thm("destVal_fun_thm",
|   tac_proof(([], ⊢ Graph (destVal: Row + Errors → Row) ∈ Functional⊢),
|   rewrite_tac[functional_def] THEN REPEAT strip_tac THEN asm_rewrite_tac[]));

```

SML

```

val updateRow_fun_thm = save_thm("updateRow_fun_thm",
|   tac_proof(([], ⊢ ∀ c t • Graph (updateRow c (TS_class t)) ∈ Functional⊢),
|   rewrite_tac[functional_def] THEN REPEAT strip_tac THEN asm_rewrite_tac[]));

```

SML

```

push_goal([], ⊢ ∀ l s (u : ℕ ↔ (ℕ ↔ Update)) c t •
|   u ∈ Functional
|   ⇒
|   (RelCombine((Squash(Id(Dom
|     (ListRel l ▷ {r|c dominates R_exist r}))))~ § u)
|     (ListRel l)
|     § Graph (updateRow c (TS_class t))) § Graph destVal ∈ Functional⊢);
a(REPEAT strip_tac);
a(lemma_tac ⊢ RelCombine ((Squash (Id (Dom
|   (ListRel l ▷ {r|c dominates R_exist r}))))~ § u)
|   (ListRel l) ∈ Functional⊢);

```

SML

```

(* *** Goal "1" *** *)
a(lemma_tacΓ(Squash(Id(Dom
  (ListRel l ▷ {r|c dominates R_exist r}))))~ ∈ Functional⊃);
(* *** Goal "1.1" *** *)
a(rewrite_tac[squash_id_thm,enumerate_thm,inv_rel_def,
  functional_def] THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 4 (rewrite_thm_tac o eq_sym_rule)
  THEN POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
(* *** Goal "1.2" *** *)
a(bc_tac[rel_combine_fun_thm] THEN bc_tac[§_fun_thm1]
  THEN asm_rewrite_tac[list_rel_fun_thm]);

```

SML

```

(* *** Goal "2" *** *)
a(bc_tac[§_fun_thm] THEN bc_tac[§_fun_thm] THEN
  asm_rewrite_tac[destVal_fun_thm,updateRow_fun_thm]);
val conjunct1_fun_lemma = save_pop_thm"conjunct1_fun_lemma";

```

HOL output

```

conjunct1_fun_lemma =
| ⊢ ∀ l s u c t
| • u ∈ Functional
| ⇒ (RelCombine
|   ((Squash
|     (Id (Dom (ListRel l ▷ {r|c dominates R_exist r}))))~
|     § u)
|   (ListRel l)
|   § Graph (updateRow c (TS_class t)))
|   § Graph destVal
| ∈ Functional

```

SML

```

push_goal([],⌈∀ l last c (u : ℕ ↔ (ℕ ↔ Update)) t •
  u ∈ Functional
  ⇒
  RelList (ListRel (l ⌈[last])
    ⊕ (RelCombine
      ((Squash
        (Id
          (Dom
            (ListRel l
              ▷ {r|c dominates R_exist r}))))~
        § u)
      (ListRel l)
      § Graph (updateRow c (TS_class t))) § Graph destVal)
  = RelList (ListRel l ⊕ (RelCombine
    ((Squash
      (Id
        (Dom
          (ListRel l
            ▷ {r|c dominates R_exist r}))))~
      § u)
    (ListRel l)
    § Graph (updateRow c (TS_class t))) § Graph destVal) ⌈[last]⌋);

```

SML

```

a(REPEAT strip_tac);
a(bc_tac[rel_list_⊕_ax1]);
(* *** Goal "1" *** *)
a(bc_tac[conjunct1_fun_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(rewrite_tac[dom_rel_combine_null_⊆_lemma]);
val conjunct1_lemma1 = save_pop_thm "conjunct1_lemma1";

```

HOL output

```

conject1_lemma1 = ⊢ ∀ l last c u t
  • u ∈ Functional
    ⇒ RelList
      (ListRel (l ∩ [last])
        ⊕ (RelCombine
          ((Squash
            (Id
              (Dom
                (ListRel l
                  ▷ {r|c dominates R_exist r}))))~
            % u)
          (ListRel l)
          % Graph (updateRow c (TS_class t)))
          % Graph destVal)
      = RelList
        (ListRel l
          ⊕ (RelCombine
            ((Squash
              (Id
                (Dom
                  (ListRel l
                    ▷ {r|c dominates R_exist r}))))~
              % u)
            (ListRel l)
            % Graph (updateRow c (TS_class t)))
            % Graph destVal)
        ∩ [last]

```

SML

```

push_goal([],Γ∀ c last l u us t •
  (us ∈ Functional
   ∧ c dominates R_exist last
   ∧ (#(Squash (Id (Dom(ListRel (l ^ [last])
                        ▷ {r|c dominates R_exist r}))))),u) ∈ us)
⇒
RelList
  (ListRel (l ^ [last])
   ⊕ (RelCombine
      ((Squash
        (Id
         (Dom
          (ListRel (l ^ [last])
                  ▷ {r|c dominates R_exist r}))))~
        % us)
      (ListRel (l ^ [last]))
      % Graph (updateRow c (TS_class t)))
      % Graph destVal) = RelList
  (ListRel l
   ⊕ (RelCombine
      ((Squash
        (Id
         (Dom
          (ListRel l
                  ▷ {r
                    |c dominates R_exist r}))))~
        % us)
      (ListRel l)
      % Graph (updateRow c (TS_class t)))
      % Graph destVal) ^ [destVal(updateRow c (TS_class t)(u,last))]Γ);

```

SML

```

a(REPEAT strip_tac);
a(LEMMA_Tr(RelCombine
  ((Squash
    (Id
      (Dom
        (ListRel (l  $\wedge$  [last])
           $\triangleright$  {r
            |c
              dominates R_exist
                r}))))~
    % us)
    (ListRel (l  $\wedge$  [last]))
    % Graph (updateRow c (TS_class t))
    % Graph destVal
  =
  ((RelCombine
    ((Squash
      (Id
        (Dom
          (ListRel l
             $\triangleright$  {r
              |c
                dominates R_exist r}))))~
        % us)
        (ListRel l)
        % Graph (updateRow c (TS_class t))
        % Graph destVal)  $\cup$ 
    {(#l + 1, destVal(updateRow c (TS_class t)(u, last)))}
  rewrite_thm_tac);

```

SML

```

(* *** Goal "1" *** *)
a(asm_rewrite_tac[squash_∧_thm,inv_rel_singleton_thm,rel_combine_one_lemma]);
a(LEMMA_TΓ((RelCombine
              ({(# l + 1,
                #
                (Squash
                 (Id
                  (Dom
                   (ListRel l
                    ▷ {r|c dominates R_exist r}))))
                  + 1})
              § us)
              (ListRel (l ∧ [last]))
              § Graph (updateRow c (TS_class t))
              § Graph destVal) =
  {(# l + 1,
   destVal
   (updateRow c (TS_class t) (u, last)))Γrewrite_thm_tac);
a(rewrite_tac[rel_combine_def,rel_ext_clauses]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "1.1" *** *)
a(strip_asm_tac(list_∇_elimΓlΓ,ΓlastΓ,Γ{r|c dominates R_exist r}Γsize_squash_plus1_thm));
a(DROP_NTH_ASM_T 8 ante_tac THEN POP_ASM_T rewrite_thm_tac);
a(DROP_NTH_ASM_T 5 (rewrite_thm_tac o eq_sym_rule)THEN ⇒_tac);
a(lemma_tacΓu = Fst z'Γ);
(* *** Goal "1.1.1" *** *)
a(DROP_NTH_ASM_T 8 (asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∇_elimΓz''Γ,ΓuΓ,ΓFst z'Γ));
(* *** Goal "1.1.2" *** *)
a(lemma_tacΓlast = Snd z'Γ);
(* *** Goal "1.1.2.1" *** *)
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[list_rel_∧_singleton_thm]
  THEN REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
a(swap_nth_asm_concl_tac 1);
a(rewrite_tac[list_rel_def,dot_dot_def]);
(* *** Goal "1.1.2.2" *** *)
a(asm_rewrite_tac[]);

```

SML

```

| (* *** Goal "1.2" *** *)
| a( $\exists$ _tac $\lceil$ (updateRow c (TS_class t)(u, last)) $\rceil$  THEN asm_rewrite_tac[]);
| a( $\exists$ _tac $\lceil$ (u,last) $\rceil$  THEN rewrite_tac[list_rel_ $\wedge$ _singleton_thm, $\cup$ _def]);
| a(prove_ $\exists$ _tac);
| a(strip_asm_tac(list_ $\forall$ _elim $\lceil$  $\lceil$ l $\rceil$ , $\lceil$ last $\rceil$ , $\lceil$ {r|c dominates R_exist r} $\rceil$  $\rceil$ size_squash_plus1_thm));
| a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule) THEN asm_rewrite_tac[]);

```

SML

```

| (* *** Goal "2" *** *)
| a(bc_tac[rel_list_ $\oplus$ _ax2]);
| (* *** Goal "2.1" *** *)
| a(bc_tac[conjunct1_fun_lemma] THEN asm_rewrite_tac[]);
| (* *** Goal "2.2" *** *)
| a(rewrite_tac[dom_rel_combine_null_ $\subseteq$ _lemma]);
| val conjunct1_lemma2 = save_pop_thm"conjunct1_lemma2";

```

HOL output

```

conject1_lemma2 =
| ⊢ ∀ c last l u us t
  • us ∈ Functional
    ∧ c dominates R_exist last
    ∧ (#
      (Squash
        (Id
          (Dom
            (ListRel (l ^ [last])
              ▷ {r|c dominates R_exist r}))))), u)
      ∈ us
    ⇒ RelList
      (ListRel (l ^ [last])
        ⊕ (RelCombine
          ((Squash
            (Id
              (Dom
                (ListRel (l ^ [last])
                  ▷ {r|c dominates R_exist r}))))~
            % us)
          (ListRel (l ^ [last]))
          % Graph (updateRow c (TS_class t))
          % Graph destVal)
        = RelList
          (ListRel l
            ⊕ (RelCombine
              ((Squash
                (Id
                  (Dom
                    (ListRel l
                      ▷ {r|c dominates R_exist r}))))~
                % us)
              (ListRel l)
              % Graph (updateRow c (TS_class t))
              % Graph destVal)
            ^ [destVal (updateRow c (TS_class t) (u, last))]

```

5 PROOF OF SECURITY OF CRITICAL COMPONENTS

The main proof of this section is built up from a series of lemmas.

5.1 Insert Lemmas

SML

```

push_goal([],⌈∀ c1 c2 t ds •
  ⊃ c2 dominates c1
  ⇒ cleanTable c2 t = cleanTable c2 (replaceRows t
    ((TS_rows t) ^ (Map((MkRow c1) o (colDefaults c1 t))ds)))⌋);
a(REPEAT strip_tac);
a(rewrite_tac[cleanTable_def,replaceRows_def,get_spec⌈MkTableSpec⌋]);
a(cases_tac⌈c2 dominates TS_class t⌋THEN
  asm_rewrite_tac[tab_components,get_spec⌈MkTableSpec⌋,cleanColCons_def]);
a(rewrite_tac[cleanRows_def,revealRow_def]);
a(lemma_tac⌈(Map (MkRow c1 o colDefaults c1 t) ds)⌋ {r|c2 dominates R_exist r} = []⌋);
(* *** Goal "1" *** *)
a(REV_LIST_INDUCTION_T⌈ds⌋asm_tac);
(* *** Goal "1.1" *** *)
a(rewrite_tac[map_def,|_def]);
(* *** Goal "1.2" *** *)
a(rewrite_tac[|_thm,get_spec⌈MkRow⌋]);
a(∀_tac THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[]);
val insertRows_lemma = save_pop_thm"insertRows_lemma";

```

HOL output

```

insertRows_lemma =
| ⊃ ∀ c1 c2 t ds
| • ⊃ c2 dominates c1
|   ⇒ cleanTable c2 t
|     = cleanTable
|       c2
|       (replaceRows
|         t
|         (TS_rows t ^ Map (MkRow c1 o colDefaults c1 t) ds))

```

SML

```

push_goal([],Γ∀ c1 c2 s i ds
  • (¬ c2 dominates c1 ∧ tabExists c1 (tabFromEffect (InsertEffect (i,ds))) (repState s))
  ⇒ hideR (c2, repState s)
    = hideR(c2, Fst(insertQuery
      (c1, destInsert(InsertEffect (i,ds)), repState s,getTable
        (tabFromEffect (InsertEffect (i,ds))) (repState s))))Γ);
a(REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN rewrite_tac[destInsert_def,tabFromEffect_def,
  getTable_def,rewrite_rule[dom_def]tabExists_def]);
a(REPEAT strip_tac);
a(strip_asm_tac (pure_rewrite_rule[get_specΓisState⊥,get_specΓStateS⊥,↔_def,∩_def]
  (∀_elimΓs⊥isState_lemma)));
a(strip_asm_tac(list_∀_elim[ΓrepState s⊥,ΓFront i⊥,Γy⊥]at_thm1));

```

SML

```

a(LIST_DROP_NTH_ASM_T[4,5,6](MAP_EVERY ante_tac
  THEN TOP_ASM_T rewrite_thm_tac);
a(REPEAT ⇒_tac THEN DROP_NTH_ASM_T 6 ante_tac THEN
  rewrite_tac[↔_def,get_specΓIdeL⊥,get_specΓDirectoryS⊥,∩_def,×_def,
  get_specΓUniverse⊥,rel_ext_clauses,get_specΓ$P⊥] THEN strip_tac);
a(asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac[↔_def,∩_def] THEN REPEAT ⇒_tac);
a(strip_asm_tac(list_∀_elim[ΓDir_tables y⊥,ΓLast i⊥,Γy'⊥]at_thm1));
a(asm_rewrite_tac[]);
a(REPEAT ∀_tac);
a(rewrite_tac[insertQuery_def]);

```

SML

```

a(cases_tacΓ¬ Elms(Map(MkRow c1 o colDefaults c1 y') ds)
  ⊆ RowS⊥ THEN asm_rewrite_tac[]);
a(asm_rewrite_tac[changeSpec_def,hideR_def,rel_ext_clauses]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[⊕_single]);
a(CASES_TΓ x = Front i¬asm_tac);
(* *** Goal "1.1" *** *)
a(∃_tacΓ MkDirectory
    (Dir_tables y
      ⊕ {(Last i,
          replaceRows
            y'
            (TS_rows y'
              ^ Map
                (MkRow c1 o colDefaults c1 y')
                ds))})
    (Dir_exist y)
    (Dir_class y)¬THEN asm_rewrite_tac[get_specΓ MkDirectory¬]);
a(lemma_tacΓ y = z¬);
(* *** Goal "1.1.1" *** *)
a(DROP_NTH_ASM_T 15 ante_tac THEN TOP_ASM_T (rewrite_thm_tac o eq_sym_rule)
    THEN strip_tac);
a(DROP_NTH_ASM_T 15(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∀_elim[Γ x¬,Γ y¬,Γ z¬]));

```

SML

```

(* *** Goal "1.1.2" *** *)
a(asm_rewrite_tac[cleanDirectory_def]);
a(cases_tac⌈c2 dominates Dir_class z⌋ THEN asm_rewrite_tac
  [get_spec⌈MkDirectory⌋,dir_components]);
a(rewrite_tac[rel_ext_clauses,⊕_single]);
a(REPEAT ∀_tac);
a(cases_tac⌈x' = Last i⌋ THEN asm_rewrite_tac[]);
a(⇔_T strip_asm_tac);
(* *** Goal "1.1.2.1" *** *)
a(lemma_tac⌈y' = z'⌋);
(* *** Goal "1.1.2.1.1" *** *)
a(DROP_NTH_ASM_T 15 ante_tac THEN DROP_NTH_ASM_T 12 ante_tac
  THEN asm_rewrite_tac[] THEN REPEAT ⇒_tac);
a(DROP_NTH_ASM_T 2(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∀_elim[⌈Last i⌋,⌈y'⌋,⌈z'⌋]);
(* *** Goal "1.1.2.1.2" *** *)
a(prove_∃_tac);
a(asm_rewrite_tac[]);
a(bc_tac[insertRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2" *** *)
a(∃_tac⌈y'⌋ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 5(asm_rewrite_thm_tac o eq_sym_rule));
a(conv_tac eq_sym_conv);
a(bc_tac[insertRows_lemma] THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "1.2" *** *)
a( $\exists$ _tac $\Gamma$ z $\neg$ THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(CASES_T $\Gamma$ x = Front i $\neg$ asm_tac);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[ $\oplus$ _single,get_spec $\Gamma$ MkDirectory $\neg$ ,
  dir_components]THEN strip_tac);
a( $\exists$ _tac $\Gamma$ y $\neg$ THEN asm_rewrite_tac[]);
a(GET_NTH_ASM_T 2 (rewrite_thm_tac o eq_sym_rule));
a(GET_NTH_ASM_T 6 rewrite_thm_tac);
a(rewrite_tac[cleanDirectory_def,rel_ext_clauses]);
a(cases_tac $\Gamma$ c2 dominates Dir_class y $\neg$ THEN asm_rewrite_tac
  [get_spec $\Gamma$ MkDirectory $\neg$ ,dir_components]);
a(rewrite_tac[rel_ext_clauses, $\oplus$ _single]);
a(REPEAT  $\forall$ _tac);
a(cases_tac $\Gamma$ x' = Last i $\neg$ THEN asm_rewrite_tac[]);
a( $\Leftrightarrow$ _T strip_asm_tac);
(* *** Goal "2.1.1" *** *)
a( $\exists$ _tac $\Gamma$ y' $\neg$ THEN asm_rewrite_tac[]);
a(conv_tac eq_sym_conv);
a(bc_tac[insertRows_lemma] THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.1.2" *** *)
a prove_ $\exists$ _tac;
a(lemma_tac $\Gamma$ z' = y' $\neg$ );
(* *** Goal "2.1.2.1" *** *)
a(DROP_NTH_ASM_T 13 (asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_ $\forall$ _elim[ $\Gamma$ Last i $\neg$ , $\Gamma$ z' $\neg$ , $\Gamma$ y' $\neg$ ]));
(* *** Goal "2.1.2.2" *** *)
a(asm_rewrite_tac[]);
a(bc_tac[insertRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[ $\oplus$ _single,get_spec $\Gamma$ MkDirectory $\neg$ ,
  dir_components]THEN strip_tac);
a( $\exists$ _tac $\Gamma$ z $\neg$ THEN asm_rewrite_tac[]);
val insertQuery_lemma = save_pop_thm"insertQuery_lemma";

```

HOL output

```

insertQuery_lemma =
| ⊢ ∀ c1 c2 s i ds
| • ¬ c2 dominates c1
|   ∧ tabExists c1 (tabFromEffect (InsertEffect (i, ds))) (repState s)
| ⇒ hideR (c2, repState s)
|   = hideR
|     (c2,
|       insertQuery
|         (c1, destInsert (InsertEffect (i, ds)), repState s,
|           getTable
|             (tabFromEffect (InsertEffect (i, ds)))
|             (repState s)))

```

5.2 Delete Lemmas

SML

```

push_goal([], ⊢ ∀ c1 c2 s t ns •
|   ¬ c2 dominates c1
|   ⇒ cleanTable c2 t = cleanTable c2 (replaceRows t
|     (Extract (1 .. #(TS_rows t) \
|       ((revealRow c1 t) Image ns ∩ {i | R_exist (Nth (TS_rows t) i) = c1}))
|       (TS_rows t)))));
a(REPEAT strip_tac);
a(rewrite_tac[cleanTable_def, replaceRows_def, get_spec ⊢ MkTableSpec ⊣]);
a(cases_tac ⊢ c2 dominates TS_class t ⊣ THEN
|   asm_rewrite_tac[tab_components, get_spec ⊢ MkTableSpec ⊣, cleanColCons_def]);
a(rewrite_tac[cleanRows_def, revealRow_def]);

```

SML

```

a(LEMMA_TΓ TS_rows t | {r|c2 dominates R_exist r} =
  Extract
    (1 .. # (TS_rows t)
      \ Squash
        (Id
          (Dom
            (ListRel (TS_rows t)
              ▷ {r|c1 dominates R_exist r})))
          Image ns
            ∩ {i|R_exist (Nth (TS_rows t) i) = c1}
            (TS_rows t)
            | {r|c2 dominates R_exist r}∇rewrite_thm_tac);
a(lemma_tacΓ ∃ l • TS_rows t = l∇THEN_LIST
  [prove_∃_tac,POP_ASM_T rewrite_thm_tac]);
a(REV_LIST_INDUCTION_TΓ l∇asm_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[extract_def,rel_list_null_thm]);
(* *** Goal "2" *** *)
a(∇_tac THEN cases_tacΓ c2 dominates R_exist last∇ THEN
  cases_tacΓ c1 dominates R_exist last∇
  THEN asm_rewrite_tac[extract_∧_single_ax,squash_∧_thm,length_∧_one_thm,
  dot_dot_def,image_∪_thm,∪_∩_thm]);
(* *** Goal "2.1" *** *)
a(LEMMA_TΓ # l + 1 ∈ {i|1 ≤ i ∧ i ≤ # l + 1}
  \ (Squash
    (Id(Dom(ListRel l
      ▷ {r|c1 dominates R_exist r})))
    Image ns
      ∩ {i|R_exist (Nth (l ∩ [last]) i) = c1}
      ∪ {(#(Squash(Id(Dom(ListRel l
        ▷ {r|c1 dominates R_exist r}))))
        + 1, # l + 1)}
    Image ns
      ∩ {i|R_exist (Nth (l ∩ [last]) i) = c1}∇rewrite_thm_tac);

```

SML

```

(* *** Goal "2.1.1" *** *)
a(REPEAT strip_tac THEN POP_ASM_T(asm_tac o rewrite_rule[nth_length_one_thm])
  THEN DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2" *** *)
a(rewrite_tac[extract_def]);
a(LEMMA_Tr{i|1 ≤ i ∧ i ≤ # l}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth l i) = c1}
  ◁ ListRel l =
  {i|1 ≤ i ∧ i ≤ # l + 1}
  \ (Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
  ∪ {(#(Squash(Id(Dom(ListRel l
    ▷ {r|c1 dominates R_exist r}))))
    + 1, # l + 1)}
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
  ◁ ListRel lrasm_rewrite_thm_tac);

```

SML

```

a(rewrite_tac[set_dif_∪_thm,rel_ext_clauses,<_def,>_def,list_rel_def,image_def,
dot_dot_def,≤_plus1_thm]);
a(REPEAT strip_tac THEN TRY asm_rewrite_tac[]);
(* *** Goal "2.1.2.1" *** *)
a(spec_nth_asm_tac 4 ⊢ x'⊢);
(* *** Goal "2.1.2.2" *** *)
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.3" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∀_elim[⊢ l⊢,⊢ x⊢,⊢ last⊢]nth_∧_thm1)));
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.4" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[nth_length_one_thm]);
a(⇒_tac THEN DROP_NTH_ASM_T 9 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.5" *** *)
a(spec_nth_asm_tac 6 ⊢ x'⊢);
(* *** Goal "2.1.2.6" *** *)
a(spec_nth_asm_tac 6 ⊢ x'⊢);
(* *** Goal "2.1.2.7" *** *)
a(spec_nth_asm_tac 6 ⊢ x'⊢);
(* *** Goal "2.1.2.8" *** *)
a(spec_nth_asm_tac 6 ⊢ x'⊢);
(* *** Goal "2.1.2.9" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.1.2.10" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.11" *** *)
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.13" *** *)
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.14" *** *)
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.15" *** *)
a(spec_nth_asm_tac 5  $\lceil x' \rceil$ );
(* *** Goal "2.1.2.16" *** *)
a(spec_nth_asm_tac 5  $\lceil x' \rceil$ );
(* *** Goal "2.1.2.17" *** *)
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.18" *** *)
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.19" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∅_elim[ $\lceil l \rceil$ ,  $\lceil x \rceil$ ,  $\lceil last \rceil$ ]nth_∧_thm1)));
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.20" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∅_elim[ $\lceil l \rceil$ ,  $\lceil x \rceil$ ,  $\lceil last \rceil$ ]nth_∧_thm1)));
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2" *** *)
a(LEMMA_Tr# l + 1
  ∈ {i|1 ≤ i ∧ i ≤ # l + 1}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}rrewrite_thm_tac);
(* *** Goal "2.2.1" *** *)
a(rewrite_tac[image_def,squash_def,enumerate_def,list_rel_def,
  id_def,dot_dot_def,length_^_one_thm]);
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 8 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2" *** *)
a(asm_rewrite_tac[extract_def]);
a(LEMMA_Tr{i|1 ≤ i ∧ i ≤ # l}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth l i) = c1}
  ◁ ListRel l={i|1 ≤ i ∧ i ≤ # l + 1}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
  ◁ ListRel lrrewrite_thm_tac);

```

SML

```

a(rewrite_tac[list_rel_def,dot_dot_def,set_dif_def,rel_ext_clauses]);
a(REPEAT strip_tac);
(* *** Goal "2.2.2.1" *** *)
a(asm_rewrite_tac[≤_plus1_thm]);
(* *** Goal "2.2.2.2" *** *)
a(asm_rewrite_tac[≤_plus1_thm]);
(* *** Goal "2.2.2.3" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∀_elim[Γl⊃,Γx⊃,Γlast⊃]nth_∧_thm1)));
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.4" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∀_elim[Γl⊃,Γx⊃,Γlast⊃]nth_∧_thm1)));
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3" *** *)
a(rewrite_tac[image_single_thm,set_dif_∪_thm]);
a(cases_tac⊃# (Squash(Id(Dom(ListRel l ▷ {r|c1 dominates R_exist r})))
      + 1
      ∈ ns⊃THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.3.1" *** *)
a(CASES_T⊃# l + 1
  ∈ ({i|1 ≤ i ∧ i ≤ # l + 1}
    \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
      Image ns
      ∩ {i|R_exist (Nth (l ∩ [last]) i) = c1}
      ∩ ({i|1 ≤ i ∧ i ≤ # l + 1}
        \ {# l + 1} ∩ {i|R_exist (Nth (l ∩ [last]) i) = c1}⊃
        asm_tac THEN TOP_ASM_T asm_rewrite_thm_tac);
(* 2.3.1.1 and 2.3.1.2 the same except for unnecessary assumption 1 *)
(* *** Goal "2.3.1.1" *** *)
a(POP_ASM_T (fn _ => id_tac));
set_labelled_goal"2.3.1.2";

```

SML

```

(* *** Goal "2.3.1.2" *** *)
a(POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[extract_def]);
a(LEMMA_TΓ{i|1 ≤ i ∧ i ≤ # l}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth l i) = c1}
  ◁ ListRel l =
  ({i|1 ≤ i ∧ i ≤ # l + 1}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
  ∩ ({i|1 ≤ i ∧ i ≤ # l + 1}
  \ {# l + 1} ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
  ◁ ListRel l∇rewrite_thm_tac);

```

SML

```

a(rewrite_tac[rel_ext_clauses,◁_def,▷_def,list_rel_def,image_def,dot_dot_def,≤_plus1_thm]);
a(REPEAT strip_tac);
(* *** Goal "2.3.1.2.1" *** *)
a(spec_nth_asm_tac 4Γx'∇);
(* *** Goal "2.3.1.2.2" *** *)
a contr_tac;
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1.2.3" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∇_elim[Γl∇,Γx'∇,Γlast∇]nth_∧_thm1)));
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1.2.4" *** *)
a contr_tac;
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1.2.5" *** *)
a(spec_nth_asm_tac 6Γx'∇);
(* *** Goal "2.3.1.2.6" *** *)
a(spec_nth_asm_tac 6Γx'∇);

```

SML

```

(* *** Goal "2.3.1.2.7" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1.2.9" *** *)
a(spec_nth_asm_tac 6  $\lceil x' \rceil$ );
(* *** Goal "2.3.1.2.10" *** *)
a(spec_nth_asm_tac 5  $\lceil x' \rceil$ );
(* *** Goal "2.3.1.2.11" *** *)
a(spec_nth_asm_tac 5  $\lceil x' \rceil$ );
(* *** Goal "2.3.1.2.12" *** *)
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1.2.13" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∨_elim[ $\lceil l \rceil$ ,  $\lceil x \rceil$ ,  $\lceil last \rceil$ ]nth_∧_thm1)));
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1.2.14" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∨_elim[ $\lceil l \rceil$ ,  $\lceil x \rceil$ ,  $\lceil last \rceil$ ]nth_∧_thm1)));
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.3.2" *** *)
a(CASES_T $\lceil$  # l + 1
  ∈ ({i | 1 ≤ i ∧ i ≤ # l + 1}
    \ Squash (Id (Dom (ListRel l ▷ {r | c1 dominates R_exist r})))
    Image ns
    ∩ {i | R_exist (Nth (l ∧ [last]) i) = c1}
    ∩ {i | 1 ≤ i ∧ i ≤ # l + 1} $\lceil$ 
    asm_tac THEN TOP_ASM_T asm_rewrite_thm_tac);
(* 2.3.2.1 and 2.3.2.2 the same except for unnecessary assumption 1 *)
(* *** Goal "2.3.2.1" *** *)
a(POP_ASM_T (fn _ => id_tac));
set_labelled_goal"2.3.2.2";
(* *** Goal "2.3.2.2" *** *)
a(POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[extract_def]);

```

SML

```

a(LEMMA_TΓ{i|1 ≤ i ∧ i ≤ # l}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
    Image ns
    ∩ {i|R_exist (Nth l i) = c1}
  ∠ ListRel l =
  ({i|1 ≤ i ∧ i ≤ # l + 1}
    \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
      Image ns
      ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
      ∩ {i|1 ≤ i ∧ i ≤ # l + 1}
    ∠ ListRel l∇rewrite_thm_tac);
a(rewrite_tac[rel_ext_clauses,∠_def,▷_def,list_rel_def,image_def,dot_dot_def,≤_plus1_thm]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "2.3.2.2.1" *** *)
a(spec_nth_asm_tac 4 Γx'∇);
(* *** Goal "2.3.2.2.2" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∇_elimΓl∇,Γx'∇,Γlast∇]nth_∧_thm1)));
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.2.2.3" *** *)
a(spec_nth_asm_tac 5 Γx'∇);
(* *** Goal "2.3.2.2.5" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.2.2.7" *** *)
a(spec_nth_asm_tac 5 Γx'∇);
(* *** Goal "2.3.2.2.8" *** *)
a(spec_nth_asm_tac 4 Γx'∇);
(* *** Goal "2.3.2.2.9" *** *)
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.3.2.2.10" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∇_elimΓl∇,Γx'∇,Γlast∇]nth_∧_thm1)));
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.4" *** *)
a(CASES_TΓ# l + 1
  ∈ {i|1 ≤ i ∧ i ≤ # l + 1}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}Γ
  asm_tac THEN TOP_ASM_T asm_rewrite_thm_tac);
(* 2.4.1 and 2.4.2 the same except for unnecessary assumption 1 *)
(* *** Goal "2.4.1" *** *)
a(POP_ASM_T (fn _ => id_tac));
set_labelled_goal"2.4.2";
(* *** Goal "2.4.2" *** *)
a(POP_ASM_T (fn _ => id_tac));
a(rewrite_tac[extract_def]);

```

SML

```

a(LEMMA_TΓ{i|1 ≤ i ∧ i ≤ # l}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth l i) = c1}
  ◁ ListRel l =
  {i|1 ≤ i ∧ i ≤ # l + 1}
  \ Squash (Id (Dom (ListRel l ▷ {r|c1 dominates R_exist r})))
  Image ns
  ∩ {i|R_exist (Nth (l ^ [last]) i) = c1}
  ◁ ListRel lΓrewrite_thm_tac);
a(rewrite_tac[rel_ext_clauses,◁_def,▷_def,list_rel_def,image_def,dot_dot_def,≤_plus1_thm]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "2.4.2.1" *** *)
a(spec_nth_asm_tac 4  $\lceil x' \rceil$ );
(* *** Goal "2.4.2.2" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∀_elim[ $\lceil l \rceil, \lceil x \rceil, \lceil last \rceil$ ]nth_∧_thm1)));
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.4.2.3" *** *)
a(spec_nth_asm_tac 5  $\lceil x' \rceil$ );
(* *** Goal "2.4.2.4" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.4.2.5" *** *)
a(spec_nth_asm_tac 4  $\lceil x' \rceil$ );
(* *** Goal "2.4.2.6" *** *)
a(strip_asm_tac(rewrite_rule[dot_dot_def](list_∀_elim[ $\lceil l \rceil, \lceil x \rceil, \lceil last \rceil$ ]nth_∧_thm1)));
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);
val deleteRows_lemma = save_pop_thm"deleteRows_lemma";

```

HOL output

```

deleteRows_lemma =
  ⊢ ∀ c1 c2 s t ns
    • ¬ c2 dominates c1
      ⇒ cleanTable c2 t
        = cleanTable
          c2
          (replaceRows
            t
            (Extract
              (1 .. # (TS_rows t)
                \ revealRow c1 t Image ns
                ∩ {i | R_exist (Nth (TS_rows t) i) = c1}
              (TS_rows t)))

```

SML

```

push_goal([],Γ∇ c1 c2 s i ns
  • (¬ c2 dominates c1 ∧ tabExists c1 (tabFromEffect (DeleteEffect (i,ns))) (repState s))
  ⇒ hideR (c2, repState s)
    = hideR(c2, deleteQuery
      (c1, destDelete(DeleteEffect (i,ns)), repState s,getTable
        (tabFromEffect (DeleteEffect (i,ns))) (repState s)))Γ);
a(REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN rewrite_tac[destInsert_def,tabFromEffect_def,
  getTable_def,rewrite_rule[dom_def]tabExists_def]);
a(REPEAT strip_tac);
a(strip_asm_tac (pure_rewrite_rule[get_specΓisStateΓ,get_specΓStateSΓ,↔_def,∩_def]
  (∇_elimΓsΓisState_lemma)));
a(strip_asm_tac(list_∇_elim[ΓrepState sΓ,ΓFront iΓ,ΓyΓ]at_thm1));
a(LIST_DROP_NTH_ASM_T[4,5,6](MAP_EVERY ante_tac
  THEN TOP_ASM_T rewrite_thm_tac);
a(REPEAT ⇒_tac THEN DROP_NTH_ASM_T 6 ante_tac THEN
  rewrite_tac[↔_def,get_specΓIdeLΓ,get_specΓDirectorySΓ,∩_def,×_def,
  get_specΓUniverseΓ,rel_ext_clauses,get_specΓ$ℙΓ] THEN strip_tac);
a(asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac[↔_def,∩_def] THEN REPEAT ⇒_tac);
a(strip_asm_tac(list_∇_elim[ΓDir_tables yΓ,ΓLast iΓ,Γy'Γ]at_thm1));
a(asm_rewrite_tac[]);
a(REPEAT ∇_tac);
a(rewrite_tac[deleteQuery_def]);
a(asm_rewrite_tac[changeSpec_def,hideR_def,rel_ext_clauses]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[⊕_single]);
a(CASES_TΓ x = Front i∩ asm_tac);
(* *** Goal "1.1" *** *)
a(∃_tacΓ MkDirectory(Dir_tables y
    ⊕ {(Last i,
        replaceRows
            y'
            (Extract
                (1 .. # (TS_rows y')
                \ revealRow c1 y' Image ns
                ∩ {i
                | R_exist (Nth (TS_rows y') i) = c1})
                (TS_rows y'))}))
    (Dir_exist y)
    (Dir_class y)∩ THEN asm_rewrite_tac[get_specΓ MkDirectory∩]);
a(lemma_tacΓ y = z∩);

```

SML

```

(* *** Goal "1.1.1" *** *)
a(DROP_NTH_ASM_T 14 ante_tac THEN TOP_ASM_T (rewrite_thm_tac o eq_sym_rule)
  THEN strip_tac);
a(DROP_NTH_ASM_T 14(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∀_elim[⌈x⌋,⌈y⌋,⌈z⌋]));
(* *** Goal "1.1.2" *** *)
a(asm_rewrite_tac[cleanDirectory_def]);
a(cases_tac⌈c2 dominates Dir_class z⌋ THEN asm_rewrite_tac
  [get_spec⌈MkDirectory⌋,dir_components]);
a(rewrite_tac[rel_ext_clauses,⊕_single]);
a(REPEAT ∀_tac);
a(cases_tac⌈x' = Last i⌋ THEN asm_rewrite_tac[]);
a(⇔_T strip_asm_tac);
(* *** Goal "1.1.2.1" *** *)
a(lemma_tac⌈y' = z'⌋);
(* *** Goal "1.1.2.1.1" *** *)
a(DROP_NTH_ASM_T 14 ante_tac THEN DROP_NTH_ASM_T 11 ante_tac
  THEN asm_rewrite_tac[] THEN REPEAT ⇒_tac);
a(DROP_NTH_ASM_T 2(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∀_elim[⌈Last i⌋,⌈y'⌋,⌈z'⌋]));
(* *** Goal "1.1.2.1.2" *** *)
a(prove_∃_tac);
a(asm_rewrite_tac[]);
a(bc_tac[deleteRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2" *** *)
a(∃_tac⌈y'⌋ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 5(asm_rewrite_thm_tac o eq_sym_rule));
a(conv_tac eq_sym_conv);
a(bc_tac[deleteRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "1.2" *** *)
a(∃_tac⌈z⌋ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2" *** *)
a(CASES_TΓ x = Front i∇asm_tac);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac
  [⊕_single,get_specΓ MkDirectory∇,dir_components] THEN strip_tac);
a(∃_tacΓ y∇ THEN asm_rewrite_tac[]);
a(GET_NTH_ASM_T 2 (rewrite_thm_tac o eq_sym_rule));
a(GET_NTH_ASM_T 6 rewrite_thm_tac);
a(rewrite_tac[cleanDirectory_def,rel_ext_clauses]);
a(cases_tacΓ c2 dominates Dir_class y∇ THEN asm_rewrite_tac
  [get_specΓ MkDirectory∇,dir_components]);
a(rewrite_tac[rel_ext_clauses,⊕_single]);
a(REPEAT ∇_tac);
a(cases_tacΓ x' = Last i∇ THEN asm_rewrite_tac[]);
a(⇔_T strip_asm_tac);
(* *** Goal "2.1.1" *** *)
a(∃_tacΓ y'∇ THEN asm_rewrite_tac[]);
a(conv_tac eq_sym_conv);
a(bc_tac[deleteRows_lemma] THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.1.2" *** *)
a prove_∃_tac;
a(lemma_tacΓ z' = y'∇);
(* *** Goal "2.1.2.1" *** *)
a(DROP_NTH_ASM_T 12 (asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∇_elim[ΓLast i∇,Γz'∇,Γy'∇]));
(* *** Goal "2.1.2.2" *** *)
a(asm_rewrite_tac[]);
a(bc_tac[deleteRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac
  [⊕_single,get_specΓ MkDirectory∇,dir_components] THEN strip_tac);
a(∃_tacΓ z∇ THEN asm_rewrite_tac[]);
val deleteQuery_lemma = save_pop_thm"deleteQuery_lemma";

```

HOL output

```

deleteQuery_lemma =
| ⊢ ∀ c1 c2 s i ns
| • ¬ c2 dominates c1
|   ∧ tabExists c1 (tabFromEffect (DeleteEffect (i, ns))) (repState s)
| ⇒ hideR (c2, repState s)
|   = hideR
|     (c2,
|       deleteQuery
|         (c1, destDelete (DeleteEffect (i, ns)), repState s,
|           getTable
|             (tabFromEffect (DeleteEffect (i, ns)))
|             (repState s)))

```

5.3 Update Lemmas

SML

```

push_goal([], ⊢ ∀ c1 c2 d tc u
| • c1 dominates tc ∧ c2 dominates tc
| ⇒ (¬ c2 dominates c1 ∧ isVal(updateField c1 tc (u, d))
| ⇒ replaceData c2 d = replaceData c2 (destVal(updateField c1 tc (u, d))))⊢);
a(REPEAT strip_tac);
a(lemma_tac ⊢ ¬ c1 = tc⊢);
(* *** Goal "1" *** *)
a contr_tac;
a(DROP_NTH_ASM_T 4 ante_tac THEN (POP_ASM_T (rewrite_thm_tac o eq_sym_rule))
| THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN rewrite_tac[updateField_def]);
a(cases_tac ⊢ isItem u⊢ THEN asm_rewrite_tac[destVal_def]
| THEN POP_ASM_T(strip_asm_tac o rewrite_rule[get_spec ⊢ isItem⊢])
| THEN asm_rewrite_tac[¬isVal_giveError_thm]);
a(cases_tac ⊢ Dat_class d dominates c1⊢
| THEN asm_rewrite_tac[destVal_def, get_spec ⊢ MkData⊢, ¬isVal_giveError_thm]);
a(⇒_tac THEN rewrite_tac[replaceData_def]);
a(lemma_tac ⊢ ¬ c2 dominates Dat_class d⊢);
(* *** Goal "2.1" *** *)
a contr_tac;
a(fc_tac[dominates_trans] THEN asm_fc_tac[]);
(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[get_spec ⊢ MkData⊢]);
val replaceData_updateField_lemma = save_pop_thm"replaceData_updateField_lemma";

```

HOL output

```

replaceData_updateField_lemma =
| ⊢ ∀ c1 c2 d tc u
| • c1 dominates tc ∧ c2 dominates tc
|   ⇒ ¬ c2 dominates c1 ∧ isVal (updateField c1 tc (u, d))
|   ⇒ replaceData c2 d
|     = replaceData c2 (destVal (updateField c1 tc (u, d)))

```

SML

```

push_goal([], ⌈∀ c1 c2 r t u
|   • c1 dominates TS_class t ∧ c2 dominates TS_class t
|   ⇒ (¬ c2 dominates c1 ∧ isVal(updateRow c1 (TS_class t) (u, r))
|   ⇒   cleanRow c2 (Snd (cleanColCons c2 t)) r
|     = cleanRow c2 (Snd (cleanColCons c2 t))
|       (destVal(updateRow c1 (TS_class t) (u, r))))⌋);
a(REPEAT strip_tac);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[isVal_def]));
a(asm_rewrite_tac[destVal_def]);
a(POP_ASM_T ante_tac THEN rewrite_tac[updateRow_def]);
a(cases_tac⌈¬ u ∈ Functional⌋ THEN
|   cases_tac⌈((RelCombine u (R_data r)
|     § Graph (updateField c1 (TS_class t)))
|     ▷ {x|isError x} § Graph destError = {}⌋
|   THEN asm_rewrite_tac[¬giveError_eq_giveVal_thm,giveVal_eq_thm]);
a(rewrite_tac[cleanRow_def,⊕_thm,get_spec⌈MkRow⌋,row_components,rel_ext_clauses,
|   filterRow_def,rel_combine_def]);
a(strip_tac THEN asm_rewrite_tac[] THEN REPEAT ∀_tac THEN ⇔_T strip_asm_tac);

```

SML

```

(* *** Goal "1" *** *)
(* Case split on there being an update for x *)
a(cases_tacΓ∃ up • (x,up) ∈ uΓ);
(* *** Goal "1.1" *** *)
a(DROP_NTH_ASM_T 6(asm_tac o list_∀_elim
  [ΓxΓ,ΓdestVal(updateField c1 (TS_class t)(up,z))Γ]));
a(LEMMA_TΓ¬ (∃ y z
  • (∃ z'
    • ((x, Fst z') ∈ u ∧ (x, Snd z') ∈ R_data r)
      ∧ z = updateField c1 (TS_class t) z')
    ∧ y = destVal z)
  ∧ (x, destVal (updateField c1 (TS_class t) (up,z))) ∈ R_data r
  ∨ (∃ z'
    • (∃ z''
      • ((x, Fst z'') ∈ u ∧ (x, Snd z'') ∈ R_data r)
        ∧ z' = updateField c1 (TS_class t) z'')
      ∧ destVal (updateField c1 (TS_class t) (up,z)) = destVal z')Γasm_tac);

```

SML

```

(* *** Goal "1.1.1" *** *)
a(REPEAT strip_tac);
(* *** Goal "1.1.1.1" *** *)
a(∃_tacΓ(updateField c1 (TS_class t) (up, z))ΓTHEN asm_rewrite_tac[]);
a(∃_tacΓ(up, z)ΓTHEN asm_rewrite_tac[]);
(* *** Goal "1.1.1.2" *** *)
a(∃_tacΓ(updateField c1 (TS_class t) (up, z))ΓTHEN asm_rewrite_tac[]);
a(∃_tacΓ(up, z)ΓTHEN asm_rewrite_tac[]);
(* *** Goal "1.1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN ⇒_tac);
a(∃_tacΓdestVal (updateField c1 (TS_class t) (up, z))ΓTHEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 8(strip_asm_tac o
  rewrite_rule[rel_ext_clauses,rel_combine_def]));
a(list_spec_nth_asm_tac 1 [ΓxΓ,ΓdestError(updateField c1 (TS_class t) (up, z))Γ]);
a(spec_nth_asm_tac 1Γ(updateField c1 (TS_class t) (up, z))Γ);

```

SML

```

(* *** Goal "1.1.2.1" *** *)
a(strip_asm_tac(∀_elimΓ(updateField c1 (TS_class t) (up, z))Γval_or_error_type));
a(strip_asm_tac(list_∀_elimΓ[c1Γ, c2Γ, zΓ, TS_class tΓ, upΓ]replaceData_updateField_lemma));
a(asm_rewrite_tac[]);
a(∃_tacΓ cΓ THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2" *** *)
a(POP_ASM_T (strip_asm_tac o rewrite_rule[] o ∀_elimΓ(up, z)Γ));
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 6(ante_tac o list_∀_elimΓ[xΓ, zΓ] THEN REPEAT strip_tac);
(* *** Goal "1.2.1" *** *)
a(spec_nth_asm_tac 7Γ Fst zΓ);
(* *** Goal "1.2.2" *** *)
a(spec_nth_asm_tac 7Γ Fst zΓ);
(* *** Goal "1.2.3" *** *)
a(spec_nth_asm_tac 10Γ Fst zΓ);
(* *** Goal "1.2.4" *** *)
a(∃_tacΓ zΓ THEN asm_rewrite_tac[]);
a(∃_tacΓ cΓ THEN asm_rewrite_tac[]);
(* *** Goal "1.2.5" *** *)
a(∃_tacΓ zΓ THEN asm_rewrite_tac[]);
a(∃_tacΓ cΓ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 5(ante_tac o list_∇_elim[⌈x⌋,⌈z⌋]) THEN asm_rewrite_tac[]
  THEN strip_tac);
(* *** Goal "2.1" *** *)
a(∃_tac⌈z⌋ THEN asm_rewrite_tac[]);
a(∃_tac⌈c⌋ THEN asm_rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(∃_tac⌈Snd z''⌋ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 10(strip_asm_tac o
  rewrite_rule[rel_ext_clauses,rel_combine_def]));
a(list_spec_nth_asm_tac 1 [⌈x⌋,⌈destError(updateField c1 (TS_class t) z'')⌋]);
a(spec_nth_asm_tac 1 ⌈(updateField c1 (TS_class t) z'')⌋);
(* *** Goal "2.2.1" *** *)
a(strip_asm_tac(∇_elim⌈(updateField c1 (TS_class t) z'')⌋ val_or_error_type));
a(POP_ASM_T ante_tac THEN LEMMA_T⌈z'' = (Fst z'', Snd z'')⌋
  pure_once_asm_rewrite_thm_tac THEN LIST[rewrite_tac[], ⇒_T asm_tac]);
a(strip_asm_tac(list_∇_elim[⌈c1⌋,⌈c2⌋,⌈Snd z''⌋,⌈TS_class t⌋,⌈Fst z''⌋]
  replaceData_updateField_lemma));
a(asm_rewrite_tac[]);
a(∃_tac⌈c⌋ THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2" *** *)
a(spec_nth_asm_tac 1 ⌈z''⌋);
val cleanRow_updateRow_lemma = save_pop_thm"cleanRow_updateRow_lemma";

```

```

cleanRow_updateRow_lemma =
⊢ ∇ c1 c2 r t u
  • c1 dominates TS_class t ∧ c2 dominates TS_class t
    ⇒ ¬ c2 dominates c1 ∧ isVal (updateRow c1 (TS_class t) (u, r))
    ⇒ cleanRow c2 (Snd (cleanColCons c2 t)) r
      = cleanRow
        c2
        (Snd (cleanColCons c2 t))
        (destVal (updateRow c1 (TS_class t) (u, r)))

```

SML

```

push_goal([],Γ∀ c1 c2 s t us •
  us ∈ Functional
  ∧ ¬ c2 dominates c1 ∧ c1 dominates TS_class t
  ∧ ((RelCombine ((revealRow c1 t)~ § us)(ListRel (TS_rows t))
    § Graph(updateRow c1 (TS_class t))) ▷ {x|isError x})
    § Graph destError = {}
  ⇒ cleanTable c2 t = cleanTable c2 (replaceRows
    t
    (RelList
      (ListRel (TS_rows t)
        ⊕ (RelCombine
          ((revealRow c1 t)~ § us)
          (ListRel (TS_rows t))
          § Graph (updateRow c1 (TS_class t)))
          § Graph destVal)
    ))Γ);
a(REPEAT strip_tac);
a(rewrite_tac[cleanTable_def,replaceRows_def,get_specΓMkTableSpecΓ]);
a(cases_tacΓc2 dominates TS_class tΓTHEN
  asm_rewrite_tac[tab_components,get_specΓMkTableSpecΓ,cleanColCons_def]);
a(DROP_NTH_ASM_T 2 ante_tac THEN rewrite_tac[cleanRows_def,revealRow_def]);

```

SML

```

a(lemma_tacΓ∃ l • TS_rows t = lΓTHEN_LIST
  [prove_∃_tac,POP_ASM_T rewrite_thm_tac]);
a(REV_LIST_INDUCTION_TΓlΓasm_tac);
(* *** Goal "1" *** *)
a(rewrite_tac[map_def,⊕_null_thm,rel_list_null_thm]);
(* *** Goal "2" *** *)
a(∀_tac THEN ⇒_tac);
a(fc_tac[rel_combine_null_lemma]);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[] THEN strip_tac);
a(cases_tacΓc2 dominates R_exist lastΓTHEN
  cases_tacΓc1 dominates R_exist lastΓ
  THEN asm_rewrite_tac[] THEN DROP_NTH_ASM_T 3 (fn _ => id_tac));

```

SML

```

(* *** Goal "2.1" *** *)
a(cases_tac⊢∃ up • (#(Squash (Id (Dom (ListRel (l ^ [last])
                                ⊃ {r|c1 dominates R_exist r}))))),up) ∈ us⊢);
(* *** Goal "2.1.1" *** *)
a(strip_asm_tac(list_∇_elim⊢[c1⊢,last⊢,l⊢,up⊢,us⊢,t⊢]conjunct1_lemma2)
  THEN POP_ASM_T rewrite_thm_tac);
a(lemma_tac⊢isVal(updateRow c1 (TS_class t) (up, last))⊢);
(* *** Goal "2.1.1.1" *** *)
a(strip_asm_tac(∇_elim⊢(updateRow c1 (TS_class t) (up, last))⊢val_or_error_type));
a(DROP_NTH_ASM_T 7 (strip_asm_tac o rewrite_rule[rel_ext_clauses,rel_combine_def]));
a(list_spec_nth_asm_tac 1 [⊢# l + 1⊢,⊢destError(updateRow c1 (TS_class t) (up, last))⊢]);
a(spec_nth_asm_tac 1 [⊢updateRow c1 (TS_class t) (up, last))⊢);
a(POP_ASM_T(strip_asm_tac o rewrite_rule[] o ∇_elim [⊢up, last])⊢);
(* *** Goal "2.1.1.1.1" *** *)
a(spec_nth_asm_tac 1 [⊢#(Squash (Id (Dom (ListRel (l ^ [last])
                                ⊃ {r|c1 dominates R_exist r}))))⊢];
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[inv_rel_def]);
a(strip_asm_tac(list_∇_elim⊢[l⊢,last⊢,{r|c1 dominates R_exist r}⊢]size_∧_one_thm));
(* *** Goal "2.1.1.1.2" *** *)
a(POP_ASM_T(strip_asm_tac o rewrite_rule[list_rel_def,dot_dot_def,
  length_∧_one_thm,nth_length_one_thm]));

```

SML

```

(* *** Goal "2.1.1.2" *** *)
a(LEMMA_T⊢c2 dominates R_exist
  (destVal (updateRow c1 (TS_class t) (up, last))⊢rewrite_thm_tac);
(* *** Goal "2.1.1.2.1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[updateRow_def]);
a(cases_tac⊢¬ up ∈ Functional⊢THEN
  cases_tac⊢((RelCombine up (R_data last)
    § Graph (updateField c1 (TS_class t)))
    ⊃ {x|isError x})
    § Graph destError
  = {}⊢
  THEN asm_rewrite_tac[¬isVal_giveError_thm,destVal_def,get_spec⊢MkRow⊢,
  row_components]THEN REPEAT strip_tac);
(* *** Goal "2.1.1.2.2" *** *)
a(strip_asm_tac(list_∇_elim⊢[c1⊢,c2⊢,last⊢,t⊢,up⊢]
  (rewrite_rule[cleanColCons_def]cleanRow_updateRow_lemma)));

```

SML

```

(* *** Goal "2.1.2" *** *)
a(LEMMA_TΓRelCombine
  ((Squash
    (Id
      (Dom
        (ListRel (l  $\hat{\wedge}$  [last])
           $\triangleright$  {r | c1 dominates R_exist r}))))~
     $\S$  us)
  (ListRel (l  $\hat{\wedge}$  [last]))
  =RelCombine
  ((Squash
    (Id
      (Dom
        (ListRel l
           $\triangleright$  {r | c1 dominates R_exist r}))))~
     $\S$  us)
  (ListRel l)Γrewrite_thm_tac);

```

SML

```

(* *** Goal "2.1.2.1" *** *)
a(asm_rewrite_tac[squash_ $\hat{\wedge}$ _thm]);
a(LEMMA_TΓ{(# (Squash (Id (Dom (ListRel l  $\triangleright$ 
  {r | c1 dominates R_exist r})))) + 1, # l + 1)}~
   $\S$  us = {}}Γrewrite_thm_tac);
(* *** Goal "2.1.2.1.1" *** *)
a(rewrite_tac[rel_ext_clauses,inv_rel_def] THEN REPEAT strip_tac);
a(spec_nth_asm_tac 3 ΓyΓ);
a(DROP_NTH_ASM_T 4(fn _ => id_tac));
a(strip_asm_tac(list_ $\forall$ _elimΓ[lΓ,lastΓ,{r | c1 dominates R_exist r}Γ]size_squash_plus1_thm));
a contr_tac;
a(DROP_NTH_ASM_T 5 ante_tac THEN DROP_NTH_ASM_T 3 ante_tac
  THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.1.2" *** *)
a(rewrite_tac[rel_combine_one_lemma]);
(* *** Goal "2.1.2.2" *** *)
a(strip_asm_tac(list_ $\forall$ _elimΓ[lΓ,lastΓ,c1Γ,usΓ,tΓ]conjunct1_lemma1));
a(asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[rel_combine_one_lemma,squash_∧_thm]);
a(strip_asm_tac(list_∇_elim[⌈l⌉,⌈last⌉,⌈c1⌉,⌈us⌉,⌈t⌉]conjunct1_lemma1));
a(asm_rewrite_tac[]);
(* *** Goal "2.3" *** *)
a(cases_tac[∃ up • (#(Squash (Id (Dom (ListRel (l ∧ [last])
                                ▷ {r|c1 dominates R_exist r}))))],up) ∈ us⌉);
(* *** Goal "2.3.1" *** *)
a(strip_asm_tac(list_∇_elim[⌈c1⌉,⌈last⌉,⌈l⌉,⌈up⌉,⌈us⌉,⌈t⌉]conjunct1_lemma2)
    THEN POP_ASM_T rewrite_thm_tac);
a(lemma_tac[isVal(updateRow c1 (TS_class t) (up, last))⌉]);
(* *** Goal "2.3.1.1" *** *)
a(strip_asm_tac(∇_elim[updateRow c1 (TS_class t) (up, last)]⌉val_or_error_type));
a(DROP_NTH_ASM_T 7 (strip_asm_tac o rewrite_rule[rel_ext_clauses,rel_combine_def]));
a(list_spec_nth_asm_tac 1 [⌈# l + 1⌉,⌈destError(updateRow c1 (TS_class t) (up, last))⌉]);
a(spec_nth_asm_tac 1 [updateRow c1 (TS_class t) (up, last)]⌉);
a(POP_ASM_T(strip_asm_tac o rewrite_rule[] o ∇_elim [up, last]⌉));
(* *** Goal "2.3.1.1.1" *** *)
a(spec_nth_asm_tac 1 [⌈#(Squash (Id (Dom (ListRel (l ∧ [last])
                                ▷ {r|c1 dominates R_exist r}))))⌉]);
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[inv_rel_def]);
a(strip_asm_tac(list_∇_elim[⌈l⌉,⌈last⌉,⌈{r|c1 dominates R_exist r}⌉]size_∧_one_thm));
(* *** Goal "2.3.1.1.2" *** *)
a(POP_ASM_T(strip_asm_tac o rewrite_rule[list_rel_def,dot_dot_def,
    length_∧_one_thm,nth_length_one_thm]));

```

SML

```

(* *** Goal "2.3.1.2" *** *)
a(LEMMA_T[⌈¬ c2 dominates R_exist
    (destVal (updateRow c1 (TS_class t) (up, last))⌉]rewrite_thm_tac);
a(POP_ASM_T ante_tac THEN rewrite_tac[updateRow_def]);
a(cases_tac[⌈¬ up ∈ Functional⌉]THEN
    cases_tac[⌈(RelCombine up (R_data last)
        % Graph (updateField c1 (TS_class t)))
        ▷ {x|isError x}
        % Graph destError
        = {}⌉]
    THEN asm_rewrite_tac[⌈¬isVal_giveError_thm,destVal_def,get_spec[⌈MkRow⌉,
    row_components]⌉]THEN REPEAT strip_tac);

```

SML

```

(* *** Goal "2.3.2" *** *)
a(LEMMA_TΓRelCombine ((Squash
                        (Id
                         (Dom
                          (ListRel (l  $\wedge$  [last])
                                    $\triangleright$  {r | c1 dominates R_exist r}))))~
                        % us)
  (ListRel (l  $\wedge$  [last]))
=RelCombine ((Squash
              (Id
               (Dom
                (ListRel l
                  $\triangleright$  {r | c1 dominates R_exist r}))))~
              % us)
  (ListRel l)Γrewrite_thm_tac);

```

SML

```

(* *** Goal "2.3.2.1" *** *)
a(asm_rewrite_tac[squash_ $\wedge$ _thm]);
a(LEMMA_TΓ{(# (Squash (Id (Dom (ListRel l  $\triangleright$ 
                        {r | c1 dominates R_exist r})))) + 1, # l + 1)}~
      % us = {}}Γrewrite_thm_tac);

```

SML

```

(* *** Goal "2.3.2.1.1" *** *)
a(rewrite_tac[rel_ext_clauses,inv_rel_def] THEN REPEAT strip_tac);
a(spec_nth_asm_tac 3  $\lceil y \rceil$ );
a(DROP_NTH_ASM_T 4(fn => id_tac));
a(strip_asm_tac(list_∀_elim[ $\lceil l \rceil$ , $\lceil last \rceil$ , $\lceil \{r|c_1 \text{ dominates } R.\text{exist } r \} \rceil$ ]size_squash_plus1_thm));
a contr_tac;
a(DROP_NTH_ASM_T 5 ante_tac THEN DROP_NTH_ASM_T 3 ante_tac
  THEN asm_rewrite_tac[]);
(* *** Goal "2.3.2.1.2" *** *)
a(rewrite_tac[rel_combine_one_lemma]);
(* *** Goal "2.3.2.2" *** *)
a(strip_asm_tac(list_∀_elim[ $\lceil l \rceil$ , $\lceil last \rceil$ , $\lceil c_1 \rceil$ , $\lceil us \rceil$ , $\lceil t \rceil$ ]conjunct1_lemma1));
a(asm_rewrite_tac[]);
(* *** Goal "2.4" *** *)
a(asm_rewrite_tac[rel_combine_one_lemma,squash_∧_thm]);
a(strip_asm_tac(list_∀_elim[ $\lceil l \rceil$ , $\lceil last \rceil$ , $\lceil c_1 \rceil$ , $\lceil us \rceil$ , $\lceil t \rceil$ ]conjunct1_lemma1));
a(asm_rewrite_tac[]);
val updateRows_lemma = save_pop_thm"updateRows_lemma";

```

HOL output

```

updateRows_lemma =
  ⊢ ∀ c1 c2 s t us
    • us ∈ Functional
      ∧ ¬ c2 dominates c1
      ∧ c1 dominates TS_class t
      ∧ ((RelCombine ((revealRow c1 t)~ § us) (ListRel (TS_rows t))
        § Graph (updateRow c1 (TS_class t)))
        ▷ {x|isError x})
        § Graph destError
      = {}
    ⇒ cleanTable c2 t
      = cleanTable
        c2
        (replaceRows
          t
          (RelList
            (ListRel (TS_rows t)
              ⊕ (RelCombine
                ((revealRow c1 t)~ § us)
                (ListRel (TS_rows t))
                § Graph (updateRow c1 (TS_class t)))
                § Graph destVal))))

```

SML

```

push_goal([], ⌈∀ c1 c2 s i us
  • (¬ c2 dominates c1
    ∧ tabExists c1 (tabFromEffect (UpdateEffect (i,us))) (repState s)
    ∧ c1 dominates (TS_class ((getTable(tabFromEffect (UpdateEffect (i,us))))(repState s))))
  ⇒ hideR (c2, repState s)
    = hideR(c2, Fst(updateQuery
      (c1, destUpdate(UpdateEffect (i,us)), repState s, getTable
        (tabFromEffect (UpdateEffect (i,us))) (repState s))))⌋);
a(REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN rewrite_tac
  [destInsert_def, tabFromEffect_def,
   getTable_def, rewrite_rule[dom_def] tabExists_def]);
a(REPEAT strip_tac);
a(strip_asm_tac (pure_rewrite_rule[get_spec⌈isState⌋, get_spec⌈StateS⌋, ↔_def, ∩_def]
  (∀_elim⌈s⌋ isState_lemma)));
a(strip_asm_tac(list_∀_elim[⌈repState s⌋, ⌈Front i⌋, ⌈y⌋] at_thm1));

```

SML

```

a(LIST_DROP_NTH_ASM_T[4,5,6,7](MAP_EVERY ante_tac)
  THEN TOP_ASM_T rewrite_thm_tac);
a(REPEAT =>_tac THEN DROP_NTH_ASM_T 7 ante_tac THEN
  rewrite_tac[←_def,get_specΓIdeL⊔,get_specΓDirectoryS⊔,∩_def,×_def,
  get_specΓUniverse⊔,rel_ext_clauses,get_specΓ$P⊔] THEN strip_tac);
a(asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac[→_def,∩_def] THEN REPEAT =>_tac);
a(strip_asm_tac(list_∇_elimΓDir_tables y⊔,ΓLast i⊔,Γy'⊔] at_thm1));
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[] THEN =>_tac);
a(REPEAT ∇_tac);
a(rewrite_tac[updateQuery_def]);
a(conv_tac(MAP_C let_conv));
a(cases_tacΓus ∈ Functional⊔ THEN
  cases_tacΓDom (∪ (Ran us)) ⊆ {n|∃ c • c ∈ visibleCols c1 y' ∧ CS_posn c = n}⊔
  THEN asm_rewrite_tac[]);
a(cases_tacΓ((RelCombine ((revealRow c1 y')⊔ ; us)(ListRel (TS_rows y'))
  ; Graph(updateRow c1 (TS_class y')))) ▷ {x|isError x})
  ; Graph destError = {}⊔ THEN asm_rewrite_tac[]);
a(asm_rewrite_tac[changeSpec_def,hideR_def,rel_ext_clauses]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "1" *** *)
a(rewrite_tac[⊕_single]);
a(CASES_TΓx = Front i⊔asm_tac);
(* *** Goal "1.1" *** *)
a(∃_tacΓMkDirectory(Dir_tables y
  ⊕ {(Last i,
    replaceRows
      y'
      (RelList
        (ListRel (TS_rows y'))
        ⊕ (RelCombine
          ((revealRow c1 y')⊔ ; us)
          (ListRel (TS_rows y'))
          ; Graph
            (updateRow c1 (TS_class y'))))
          ; Graph destVal))})
  (Dir_exist y)
  (Dir_class y)⊔ THEN asm_rewrite_tac[get_specΓMkDirectory⊔]);
a(lemma_tacΓy = z⊔);

```

SML

```

(* *** Goal "1.1.1" *** *)
a(DROP_NTH_ASM_T 18 ante_tac THEN TOP_ASM_T (rewrite_thm_tac o eq_sym_rule)
  THEN strip_tac);
a(DROP_NTH_ASM_T 18(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∇_elim[⌈x⌋,⌈y⌋,⌈z⌋]));
(* *** Goal "1.1.2" *** *)
a(asm_rewrite_tac[cleanDirectory_def]);
a(cases_tac⌈c2 dominates Dir_class z⌋ THEN asm_rewrite_tac
  [get_spec⌈MkDirectory⌋,dir_components]);
a(rewrite_tac[rel_ext_clauses,⊕_single]);
a(REPEAT ∇_tac);
a(cases_tac⌈x' = Last i⌋ THEN asm_rewrite_tac[]);
a(↔_T strip_asm_tac);

```

SML

```

(* *** Goal "1.1.2.1" *** *)
a(lemma_tac⌈y' = z'⌋);
(* *** Goal "1.1.2.1.1" *** *)
a(DROP_NTH_ASM_T 18 ante_tac THEN DROP_NTH_ASM_T 15 ante_tac
  THEN asm_rewrite_tac[] THEN REPEAT ⇒_tac);
a(DROP_NTH_ASM_T 2(asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∇_elim[⌈Last i⌋,⌈y'⌋,⌈z'⌋]));
(* *** Goal "1.1.2.1.2" *** *)
a(prove_∃_tac);
a(DROP_NTH_ASM_T 14 ante_tac THEN DROP_NTH_ASM_T 11 ante_tac
  THEN asm_rewrite_tac[] THEN REPEAT ⇒_tac);
a(bc_tac[updateRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2.2" *** *)
a(∃_tac⌈y'⌋ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 5(asm_rewrite_thm_tac o eq_sym_rule));
a(conv_tac eq_sym_conv);
a(bc_tac[updateRows_lemma] THEN_TRY asm_rewrite_tac[]);
(* *** Goal "1.2" *** *)
a(∃_tac⌈z⌋ THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2" *** *)
a(CASES_TΓ x = Front i∇asm_tac);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[⊕_single,get_specΓMkDirectory∇,
  dir_components]THEN strip_tac);
a(∃_tacΓ y∇ THEN asm_rewrite_tac[]);
a(GET_NTH_ASM_T 2 (rewrite_thm_tac o eq_sym_rule));
a(GET_NTH_ASM_T 6 rewrite_thm_tac);
a(rewrite_tac[cleanDirectory_def,rel_ext_clauses]);
a(cases_tacΓ c2 dominates Dir_class y∇ THEN asm_rewrite_tac
  [get_specΓMkDirectory∇,dir_components]);
a(rewrite_tac[rel_ext_clauses,⊕_single]);
a(REPEAT ∇_tac);
a(cases_tacΓ x' = Last i∇ THEN asm_rewrite_tac[]);
a(⇔_T strip_asm_tac);

```

SML

```

(* *** Goal "2.1.1" *** *)
a(∃_tacΓ y'∇ THEN asm_rewrite_tac[]);
a(conv_tac eq_sym_conv);
a(bc_tac[updateRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2" *** *)
a prove_∃_tac;
a(lemma_tacΓ z' = y'∇);
(* *** Goal "2.1.2.1" *** *)
a(DROP_NTH_ASM_T 16 (asm_tac o rewrite_rule[functional_def]));
a(POP_ASM_T (strip_asm_tac o list_∇_elim[ΓLast i∇,Γz'∇,Γy'∇]));
(* *** Goal "2.1.2.2" *** *)
a(asm_rewrite_tac[]);
a(bc_tac[updateRows_lemma] THEN asm_rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[⊕_single,get_specΓMkDirectory∇,
  dir_components]THEN strip_tac);
a(∃_tacΓ z∇ THEN asm_rewrite_tac[]);
val updateQuery_lemma = save_pop_thm"updateQuery_lemma";

```

HOL output

```

updateQuery_lemma =
| ⊢ ∀ c1 c2 s i us
| • ¬ c2 dominates c1
|   ∧ tabExists c1 (tabFromEffect (UpdateEffect (i, us))) (repState s)
|   ∧ c1
|     dominates TS_class
|       (getTable (tabFromEffect (UpdateEffect (i, us))) (repState s))
⇒ hideR (c2, repState s)
= hideR
  (c2,
   Fst
    (updateQuery
     (c1, destUpdate (UpdateEffect (i, us)), repState s,
      getTable
        (tabFromEffect (UpdateEffect (i, us)))
        (repState s))))

```

5.4 Proof of Conjunct 1

SML

```

push_goal([], ⌈∀ c1 c2 s e
  • ¬ hideR (c2, repState s) = hideR (c2, Fst (updateStateR (c1, e, repState s)))
  ⇒ c2 dominates c1⌋);
a(REPEAT strip_tac);
a(swap_nth_asm_concl_tac 1);
a(LEMMA_T ⌈e = (Fst e, Snd e)⌋ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], rewrite_tac[updateStateR_def]]);
a(strip_asm_tac(⌈∀_elim ⌈Fst e⌋ query_type) THEN asm_rewrite_tac[]
  THEN cases_tac ⌈¬ Snd e = []⌋ THEN asm_rewrite_tac[]
  THEN cases_tac ⌈tabExists c1 (tabFromEffect (Fst e)) (repState s)⌋
  THEN asm_rewrite_tac[]
  THEN cases_tac ⌈¬ c1 dominates TS_class(getTable (tabFromEffect (Fst e))
    (repState s))⌋
  THEN asm_rewrite_tac[]);

```

SML

```

| (* *** 3 subgoals – state unchanged for Select *** *)
| (* *** Goal "1" *** *)
| (** Insert **)
| a(POP_ASM_T (fn _ => id_tac) THEN POP_ASM_T ante_tac);
| a(DROP_NTH_ASM_T 5 (strip_asm_tac o rewrite_rule[isInsert_def]));
| a(TOP_ASM_T rewrite_thm_tac);
| a(LEMMA_TΓ i = (Fst i, Snd i)Γ pure_once_asm_rewrite_thm_tac
|   THEN_LIST[rewrite_tac[], ⇒_T asm_tac]);
| a(bc_tac[insertQuery_lemma] THEN asm_rewrite_tac[]);
| (* *** Goal "2" *** *)
| (** Delete **)
| a(POP_ASM_T (fn _ => id_tac) THEN POP_ASM_T ante_tac);
| a(DROP_NTH_ASM_T 5 (strip_asm_tac o rewrite_rule[isDelete_def]));
| a(TOP_ASM_T rewrite_thm_tac);
| a(LEMMA_TΓ d = (Fst d, Snd d)Γ pure_once_asm_rewrite_thm_tac
|   THEN_LIST[rewrite_tac[], ⇒_T asm_tac]);
| a(bc_tac[deleteQuery_lemma] THEN asm_rewrite_tac[]);

```

SML

```

| (* *** Goal "3" *** *)
| (** Update **)
| a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
| a(DROP_NTH_ASM_T 5 (strip_asm_tac o rewrite_rule[isUpdate_def]));
| a(TOP_ASM_T rewrite_thm_tac);
| a(LEMMA_TΓ u = (Fst u, Snd u)Γ pure_once_asm_rewrite_thm_tac
|   THEN_LIST[rewrite_tac[], REPEAT ⇒_tac]);
| a(bc_tac[updateQuery_lemma] THEN asm_rewrite_tac[]);
| val conjunct1 = save_pop_thm "conjunct1";

```

HOL output

```

| conjunct1 =
| ⊢ ∀ c1 c2 s e
|   • ¬ hideR (c2, repState s)
|     = hideR (c2, Fst (updateStateR (c1, e, repState s)))
|     ⇒ c2 dominates c1

```

6 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

SML

```

| pop_pc();

```

7 THE THEORY fef012

7.1 Parents

fef011

7.2 Children

fef013

7.3 Theorems

destItem_consistent

destClass_consistent

destData_consistent

\vdash *Consistent*

$(\lambda (destItem', destClass', destData')$

• $\forall i c d$

• $destItem' (ItemUpdate i) = i$

$\wedge destClass' (ClassUpdate c) = c$

$\wedge destData' (DataUpdate d) = d)$

\neg giveVal_eq_giveError_thm

$\vdash \forall v e \bullet \neg giveVal v = giveError e$

\neg giveError_eq_giveVal_thm

$\vdash \forall e v \bullet \neg giveError e = giveVal v$

rel_combine_one_lemma

$\vdash \forall l last s u$

• *RelCombine*

$(Squash (Id (Dom (ListRel l \triangleright s))) \sim \% u)$

$(ListRel (l @ [last]))$

$= RelCombine$

$(Squash (Id (Dom (ListRel l \triangleright s))) \sim \% u)$

$(ListRel l)$

rel_combine_null_lemma

$\vdash \forall l last c t us$

• $((RelCombine$

$(Squash$

$(Id$

$(Dom$

$(ListRel (l @ [last])$

$\triangleright \{r$

$|c$

$dominates R_exist$

$r\})) \sim$

$\% us)$

$(ListRel (l @ [last]))$

$\% Graph (updateRow c (TS_class t))$

$$\begin{aligned}
& \triangleright \{x \mid \text{isError } x\} \\
& \% \text{ Graph destError} \\
& = \{\} \\
\Rightarrow & ((\text{RelCombine} \\
& \quad (\text{Squash} \\
& \quad \quad (\text{Id} \\
& \quad \quad \quad (\text{Dom} \\
& \quad \quad \quad \quad (\text{ListRel } l \\
& \quad \quad \quad \quad \quad \triangleright \{r \\
& \quad \quad \quad \quad \quad \quad |c \\
& \quad \quad \quad \quad \quad \quad \quad \text{dominates } R_exist \\
& \quad \quad \quad \quad \quad \quad \quad \quad r\})) \sim \\
& \quad \quad \quad \quad \quad \quad \quad \%) \\
& \quad \quad \quad (\text{ListRel } l) \\
& \quad \quad \%) \text{ Graph } (\text{updateRow } c \text{ (TS_class } t))) \\
& \quad \triangleright \{x \mid \text{isError } x\} \\
& \quad \% \text{ Graph destError} \\
& = \{\}
\end{aligned}$$

dom_rel_combine_null_⊆_lemma $\vdash \forall l \ s \ u \ c \ t$ • *Dom*

$$\begin{aligned}
& ((\text{RelCombine} \\
& \quad (\text{Squash} \\
& \quad \quad (\text{Id} \\
& \quad \quad \quad (\text{Dom} \\
& \quad \quad \quad \quad (\text{ListRel } l \\
& \quad \quad \quad \quad \quad \triangleright \{r \\
& \quad \quad \quad \quad \quad \quad |c \\
& \quad \quad \quad \quad \quad \quad \quad \text{dominates } R_exist \\
& \quad \quad \quad \quad \quad \quad \quad \quad r\})) \sim \\
& \quad \quad \quad \quad \quad \quad \quad \%) \\
& \quad \quad \quad (\text{ListRel } l) \\
& \quad \quad \%) \text{ Graph } (\text{updateRow } c \text{ (TS_class } t))) \\
& \quad \%) \text{ Graph destVal} \\
& \subseteq \text{Dom } (\text{ListRel } l)
\end{aligned}$$

destVal_fun_thm $\vdash \text{Graph destVal} \in \text{Functional}$ **updateRow_fun_thm** $\vdash \forall c \ t \bullet \text{Graph } (\text{updateRow } c \text{ (TS_class } t)) \in \text{Functional}$ **conjunct1_fun_lemma** $\vdash \forall l \ s \ u \ c \ t$ • $u \in \text{Functional}$

$$\begin{aligned}
\Rightarrow & (\text{RelCombine} \\
& \quad (\text{Squash} \\
& \quad \quad (\text{Id} \\
& \quad \quad \quad (\text{Dom} \\
& \quad \quad \quad \quad (\text{ListRel } l \\
& \quad \quad \quad \quad \quad \triangleright \{r
\end{aligned}$$

$$\begin{array}{c}
|c \\
\text{dominates } R_exist \\
r\}})) \sim \\
\% u) \\
(ListRel\ l) \\
\% Graph\ (updateRow\ c\ (TS_class\ t))) \\
\% Graph\ destVal \\
\in\ Functional \\
\text{conjunct1_lemma1} \\
\vdash\ \forall\ l\ last\ c\ u\ t \\
\bullet\ u \in Functional \\
\Rightarrow RelList \\
(ListRel\ (l\ @\ [last]) \\
\oplus\ (RelCombine \\
(Squash \\
(Id \\
(Dom \\
(ListRel\ l \\
\triangleright\ \{r \\
|c \\
\text{dominates } R_exist \\
r\}})) \sim \\
\% u) \\
(ListRel\ l) \\
\% Graph\ (updateRow\ c\ (TS_class\ t))) \\
\% Graph\ destVal \\
= RelList \\
(ListRel\ l \\
\oplus\ (RelCombine \\
(Squash \\
(Id \\
(Dom \\
(ListRel\ l \\
\triangleright\ \{r \\
|c \\
\text{dominates } R_exist\ r\}})) \sim \\
\% u) \\
(ListRel\ l) \\
\% Graph\ (updateRow\ c\ (TS_class\ t))) \\
\% Graph\ destVal) \\
@ [last]
\end{array}$$
conjunct1_lemma2

$$\begin{array}{c}
\vdash\ \forall\ c\ last\ l\ u\ us\ t \\
\bullet\ us \in Functional \\
\wedge\ c\ \text{dominates } R_exist\ last \\
\wedge\ (\# \\
(Squash \\
(Id
\end{array}$$

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

insertRows_lemma

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

$$\begin{aligned}
& \text{deleteQuery} \\
& (c_1, \\
& \quad \text{destDelete} \\
& \quad \quad (\text{DeleteEffect } (i, ns)), \\
& \quad \text{repState } s, \\
& \quad \text{getTable} \\
& \quad \quad (\text{tabFromEffect} \\
& \quad \quad \quad (\text{DeleteEffect } (i, ns))) \\
& \quad \quad (\text{repState } s))
\end{aligned}$$
replaceData_updateField_lemma

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

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

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

$$\begin{aligned}
& (\text{revealRow } c_1 \ t \ \sim \ \% \ us) \\
& (\text{ListRel } (TS_rows \ t)) \\
& \% \ \text{Graph} \\
& (\text{updateRow } c_1 \ (TS_class \ t)) \\
& \% \ \text{Graph } \text{destVal}))
\end{aligned}$$
updateQuery_lemma

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

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

8 INDEX

<i>cleanRow_updateRow_lemma</i>	44
<i>conjunct1_fun_lemma</i>	11
<i>conjunct1_lemma1</i>	12
<i>conjunct1_lemma2</i>	17
<i>conjunct1</i>	56
<i>deleteQuery_def</i>	5
<i>deleteQuery_lemma</i>	39
<i>deleteRows_lemma</i>	35
<i>destError_def</i>	5
<i>destItem_def</i>	5
<i>destVal_def</i>	5
<i>destVal_fun_thm</i>	10
<i>dom_rel_combine_null_⊆_lemma</i>	9
<i>fef012</i>	4
<i>insertQuery_lemma</i>	23
<i>insertRows_lemma</i>	19
<i>rel_combine_null_lemma</i>	8
<i>rel_combine_one_lemma</i>	6
<i>replaceData_updateField_lemma</i>	40
<i>updateQuery_lemma</i>	54
<i>updateRows_lemma</i>	50
<i>updateRow_fun_thm</i>	10
\neg <i>giveError_eq_giveVal_thm</i>	6
\neg <i>giveVal_eq_giveError_thm</i>	6