Project: DRA FRONT END FILTER PROJECT

Title: Table Computation Security Proofs

Ref: DS/FMU/FEF/035 *Issue: Revision* : *1.9* *Date:* 5 June 2016

Status: Draft *Type:* Specification

Keywords:

Authors:

Name	*Location*	*Signature*	*Date*
R. D. Arthan	WIN01		
G. M. Prout	WIN01		

Authorisation for Issue:

Name	*Function*	*Signature*	*Date*
R.B. Jones	HAT Manager		

Abstract: This document completes the formal proofs relating to DS/FMU/FEF/032; it forms part of the Phase II proofs for the DRA front end filter project RSRE 1C/6130.

Distribution: HAT FEF File
Simon Wiseman

0 DOCUMENT CONTROL

0.1 Contents List

0.2 Document Cross References

[1] DS/FMU/FEF/018. *Proposal for Phase 2.* G.M. Prout, ICL Secure Systems, WIN01.

[2] DS/FMU/FEF/026. *Critical Requirements on the SWORD Query Transformations.* R.D. Arthan, ICL Secure Systems, WIN01.

0.3 Changes History

Issue *Revision* : *1.9* **(5 June 2016)** Lemmas renamed. Section bringing together proofs from fef033 and fef035 moved to fef036.

Issue 1.10 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document provides a formal proof relating to the specifications in "An Execution Model for SWORD" [2]. It constitutes part of deliverable D13 of work package 3, as given in section 3 of the Proposal for Phase 2, [1].

(The current version is a rough draft of proofs of some of the propositions produced as pilot work during the third stage of phase 2.)

1.2 Introduction

2 PRELIMINARIES

The following ProofPower instructions set up a new theory *fef035* to hold the theorems to be proved and set up a proof context in which to carry out the proofs.

```
SML
open_theory "fef033";
(force_delete_theory "fef035" handle _ => ());
new_theory"fef035";
set_pc"hol";
```

3 LEMMAS

```
SML
push_goal([],⌜∀ f l• Length (Map f l) = Length l⌝);
a(REPEAT ∀_tac);
a(list_induction_tac⌜l⌝THEN asm_rewrite_tac(map get_spec[⌜Map⌝,⌜Length⌝]));
val length_map_lemma = save_pop_thm"length_map_lemma";
```

SML

```
push_goal([], ⌜∀l i f • i ≤ # l ∧ 1 ≤ i
        ⇒ f (Nth l i) = Nth (Map f l) i ⌝);
a ∀_tac;
a(list_induction_tac⌜l⌝ THEN rewrite_tac(map get_spec[⌜Nth⌝,⌜Map⌝,⌜Length⌝])
        THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(cases_tac⌜i=1⌝THEN asm_rewrite_tac[]);
a(list_spec_nth_asm_tac 4 [⌜i−1⌝,⌜f⌝]);
(* *** Goal "2.1" *** *)
a(GET_ASM_T ⌜1 ≤ i⌝ (strip_asm_tac o rewrite_rule[get_spec⌜$≤⌝]));
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[]));
(* *** Goal "2.2" *** *)
a(GET_ASM_T ⌜1 ≤ i⌝ (strip_asm_tac o rewrite_rule[get_spec⌜$≤⌝]));
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[]));
a(strip_asm_tac (∀_elim⌜i'⌝ ℕ_cases_thm));
a(var_elim_nth_asm_tac 1);
val fun_nth_map_lemma = save_pop_thm"fun_nth_map_lemma";
```

SML

```
push_goal([], ⌜∀i f l1 l2 • i ≤ # l1 ∧ 1 ≤ i ∧ Map f l1 = Map f l2
        ⇒ f (Nth l1 i) = f (Nth l2 i) ⌝);
a(REPEAT strip_tac);
a(lemma_tac⌜# (Map f l1) = # (Map f l2)⌝
        THEN1 asm_rewrite_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule [length_map_thm]));
a(lemma_tac⌜i ≤ # l2⌝
        THEN1 POP_ASM_T (asm_rewrite_thm_tac o eq_sym_rule));
a(all_fc_tac[fun_nth_map_lemma]);
a(asm_rewrite_tac[]);
val fun_nth_map_lemma1 = save_pop_thm"fun_nth_map_lemma1";
```

SML
```
push_goal([],⌜∀ tl el gp gps •
        ProjectData tl el [] = []
∧       ProjectData tl el (Cons gp gps) =
        Map (λ r• MkDerTableRow (DTR_where r) (DTR_row r) (Map (λ e• e tl gp r) el)) gp
        ⌢ ProjectData tl el gps⌝);
a(rewrite_tac(map get_spec[⌜Map⌝,⌜ProjectData⌝,⌜Flat⌝,⌜Let⌝]) THEN REPEAT strip_tac);
val ProjectData_lemma = save_pop_thm"ProjectData_lemma";
```

SML
```
set_goal([], ⌜∀c t• DT_spec (HideDerTable c t) = DT_spec t⌝);
a(rewrite_tac (map get_spec[⌜HideDerTable⌝, ⌜DT_spec⌝, ⌜Let⌝]));
val DT_spec_HideDerTable_lemma = save_pop_thm"DT_spec_HideDerTable_lemma";
```

SML
```
set_goal([], ⌜∀c ts1 ts2•
        Map (HideDerTable c) ts1 = Map (HideDerTable c) ts2
⇒       (Map DT_spec ts1) = (Map DT_spec ts2)⌝);
a(REPEAT strip_tac);
a(LEMMA_T⌜
        Map DT_spec(Map (HideDerTable c) ts1) =
        Map DT_spec(Map (HideDerTable c) ts2)⌝
        ante_tac THEN1 asm_rewrite_tac[]);
a(lemma_tac⌜DT_spec o HideDerTable c = DT_spec⌝
        THEN1 PC_T1 "hol2" rewrite_tac[DT_spec_HideDerTable_lemma]);
a(asm_rewrite_tac[map_o_lemma]);
val map_HideDerTable_map_DT_spec_lemma =
        save_pop_thm"map_HideDerTable_map_DT_spec_lemma";
```

SML
```
push_goal([], ⌜∀c r rs•
        HideDerTableData c [] = []
∧       HideDerTableData c (Cons r rs) =
                if      c dominates DTR_row r
                then    Cons (HideDerTableRow c r) (HideDerTableData c rs)
                else    HideDerTableData c rs
⌝);
a(rewrite_tac(map get_spec[⌜HideDerTableData⌝, ⌜$⌊⌝, ⌜Map⌝, ⌜Let⌝]));
a(REPEAT strip_tac);
a(cases_tac⌜c dominates DTR_row r⌝ THEN asm_rewrite_tac[get_spec⌜Map⌝]);
val HideDerTableData_lemma = save_pop_thm"HideDerTableData_lemma";
```

SML

```
push_goal([], ⌜∀c r1 r2 rs•
        JoinRows r1 [] = []
∧       JoinRows r1 (Cons r2 rs) =
        Cons
        (MkDerTableRow
                (DTR_where r1 lub DTR_where r2)
                (DTR_row r1 lub DTR_row r2)
                (DTR_cols r1 ⌢ DTR_cols r2))
        (JoinRows r1 rs)
⌝);
a(rewrite_tac(map get_spec[⌜JoinRows⌝, ⌜Map⌝, ⌜Let⌝]));
val JoinRows_lemma = save_pop_thm"JoinRows_lemma";
```

SML

```
push_goal([], ⌜∀c blks•
        HideDerTableData c (Flat blks) = Flat (Map (HideDerTableData c) blks)
⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜blks⌝ THEN asm_rewrite_tac(map get_spec[⌜Flat⌝, ⌜Map⌝]));
(* *** Goal "1" *** *)
a(rewrite_tac(map get_spec[⌜HideDerTableData⌝, ⌜Map⌝, ⌜Let⌝, ⌜$⌊⌝]));
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(list_induction_tac⌜x⌝);
(* *** Goal "1" *** *)
a(lemma_tac⌜HideDerTableData c [] = []⌝
        THEN1 rewrite_tac(map get_spec[⌜HideDerTableData⌝, ⌜Map⌝, ⌜Let⌝, ⌜$⌊⌝]));
a(asm_rewrite_tac(map get_spec[⌜$Append⌝, ⌜Map⌝, ⌜Flat⌝]));
(* *** Goal "2.2" *** *)
a(rewrite_tac(map get_spec[⌜$Append⌝, ⌜Map⌝, ⌜Flat⌝]));
a(all_asm_ante_tac
        THEN rewrite_tac(map get_spec[⌜HideDerTableData⌝, ⌜Map⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(rewrite_tac[get_spec⌜$⌊⌝]);
a(cases_tac ⌜c dominates DTR_row x'⌝ THEN
        TOP_ASM_T rewrite_thm_tac
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝, ⌜$Append⌝]));
val HideDerTable_flat_lemma =
        save_pop_thm"HideDerTable_flat_lemma";
```

SML

```
val conj1 = ⌜∀tl0  tl1  c•
        Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1 ⇒
        HideDerTable
              c
              (MkDerTable
                  (Fst (Join tl0))
                  (Snd (Join tl0)))
            = HideDerTable
              c
              (MkDerTable
                  (Fst (Join tl1))
                  (Snd (Join tl1)))⌝;
val conj2 = ⌜∀c ts1 ts2•
        Map (HideDerTable c) ts1 = Map (HideDerTable c) ts2
⇒       HideDerTableData c (JoinData (Map DT_rows ts1)) =
        HideDerTableData c (JoinData (Map DT_rows ts2))⌝;
val conj3 = ⌜∀c ts1 ts2•
        Map (HideDerTable c) ts1 = Map (HideDerTable c) ts2
⇒       Map (HideDerTableData c) (Map DT_rows ts1)
        = Map (HideDerTableData c) (Map DT_rows ts2)⌝;
val conj4 = ⌜∀c ds1 ds2•
        Map (HideDerTableData c) ds1 = Map (HideDerTableData c) ds2
⇒       HideDerTableData c (JoinData ds1) =
        HideDerTableData c (JoinData ds2)⌝;
val conj5 = ⌜∀c rs1 rs2•
        Flat(Map (λr• HideDerTableData c (JoinRows r rs2)) rs1) =
        Flat(Map (λr• JoinRows r (HideDerTableData c rs2)) (HideDerTableData c rs1))⌝;
val conj6 = ⌜∀c r rs•
                HideDerTableData c (JoinRows r rs)
        =       if      c dominates DTR_row r
                then    JoinRows (HideDerTableRow c r) (HideDerTableData c rs)
                else []⌝;
```

SML

```
set_goal([conj2], conj1);
a(REPEAT strip_tac);
a(rewrite_tac(MkDerTable_lemma :: map get_spec[⌜Join⌝, ⌜HideDerTable⌝, ⌜MkDerTable⌝]));
a(ALL_ASM_FC_T rewrite_tac [map_HideDerTable_map_DT_spec_lemma]);
val Join_lemma1 = save_pop_thm"Join_lemma1";
```

SML
```
set_goal([conj3, conj4], conj2);
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(all_asm_fc_tac[]);
val Join_lemma2 = save_pop_thm"Join_lemma2";
```

SML
```
set_goal([], conj3);
a(REPEAT_N 2 strip_tac);
a(list_induction_tac⌜ts1⌝);
(* *** Goal "1" *** *)
a(strip_tac);
a(strip_asm_tac(∀_elim⌜ts2⌝ list_cases_thm)
        THEN asm_rewrite_tac[map_def]);
(* *** Goal "2" *** *)
a(REPEAT_N 2 strip_tac);
a(strip_asm_tac(∀_elim⌜ts2⌝ list_cases_thm)
        THEN asm_rewrite_tac[map_def]);
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(asm_ante_tac ⌜HideDerTable c x = HideDerTable c x'⌝
        THEN rewrite_tac[MkDerTable_lemma, get_spec⌜HideDerTable⌝]);
a(REPEAT strip_tac);
val Join_lemma3 = save_pop_thm"Join_lemma3";
```

SML
```
set_goal([conj5], conj4);
a(strip_tac THEN strip_tac);
a(list_induction_tac ⌜ds1⌝);
(* *** Goal "1" *** *)
a(rewrite_tac[get_spec⌜Map⌝, map_null_thm]);
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(rewrite_tac[get_spec⌜Map⌝] THEN REPEAT ∀_tac);
a(strip_asm_tac(∀_elim⌜ds2⌝list_cases_thm) THEN
        asm_rewrite_tac[get_spec⌜Map⌝, map_null_thm]);
a(REPEAT strip_tac);
a(strip_asm_tac (∀_elim⌜ds1⌝list_cases_thm));
```

SML

```
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝, map_null_thm]);
a(REPEAT strip_tac THEN var_elim_asm_tac⌜list2 = []⌝);
a(asm_rewrite_tac(map get_spec [⌜JoinData⌝, ⌜Let⌝]));
(* *** Goal "2.2" *** *)
a(lemma_tac⌜¬ds1 = [] ∧ ¬list2 = []⌝ THEN1 asm_rewrite_tac[]);
(* *** Goal "2.2.1" *** *)
a(swap_nth_asm_concl_tac 2 THEN asm_rewrite_tac[get_spec⌜Map⌝, map_null_thm]);
(* *** Goal "2.2.2" *** *)
a(LIST_DROP_NTH_ASM_T [3, 6] (Combinators.K id_tac));
a(asm_rewrite_tac(let_def::HideDerTable_flat_lemma::map get_spec [⌜JoinData⌝]));
a(asm_rewrite_tac[map_o_lemma,
        pc_rule1"hol2"prove_rule[]⌜∀f g z•(f o (λx•g x z)) = (λx•f(g x z))⌝]);
a(ALL_ASM_FC_T rewrite_tac[]);
val Join_lemma4 = save_pop_thm"Join_lemma4";
```

SML

```
set_goal([conj6], conj5);
a(REPEAT strip_tac);
a(list_induction_tac ⌜rs1⌝);
(* *** Goal "1" *** *)
a(rewrite_tac(map get_spec[⌜HideDerTableData⌝, ⌜Flat⌝, ⌜Map⌝, ⌜Let⌝, ⌜$⍀⌝]));
(* *** Goal "2" *** *)
a(rewrite_tac[dominates_lub_lemma,HideDerTableData_lemma,get_spec⌜Flat⌝,get_spec⌜Map⌝]);
a(strip_tac);
a(cases_tac⌜c dominates DTR_row x⌝
        THEN asm_rewrite_tac(map get_spec[⌜$⌢⌝, ⌜Map⌝, ⌜Flat⌝]));
val Join_lemma5 = save_pop_thm"Join_lemma5";
```

SML

```
set_goal([], conj6);
a(REPEAT strip_tac);
a(list_induction_tac⌜rs⌝);
(* *** Goal "1" *** *)
a(rewrite_tac[HideDerTableData_lemma, JoinRows_lemma]);
a(CASES_T ⌜c dominates DTR_row r⌝ rewrite_thm_tac);
(* *** Goal "2" *** *)
a(strip_tac);
a(cases_tac ⌜c dominates DTR_row r⌝ THEN asm_rewrite_tac[]);
```

SML

```
(* *** Goal "2.1" *** *)
a(swap_nth_asm_concl_tac 2 THEN asm_rewrite_tac[] THEN swap_nth_asm_concl_tac 1);
a(DROP_NTH_ASM_T 2 ante_tac);
a(rewrite_tac[HideDerTableData_lemma, JoinRows_lemma,
        dominates_lub_lemma, get_spec⌜MkDerTableRow⌝]);
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
a(cases_tac ⌜c dominates DTR_row x⌝THEN asm_rewrite_tac[]);
a(rewrite_tac[JoinRows_lemma]);
a(asm_rewrite_tac(MkDerTableRow_lemma:: map_⌒_thm1::
        map get_spec[⌜HideDerTableRow⌝, ⌜MkDerTableRow⌝, ⌜Let⌝]));
(* *** Goal "2.2" *** *)
a(POP_ASM_T ante_tac);
a(asm_rewrite_tac[HideDerTableData_lemma, JoinRows_lemma,
        dominates_lub_lemma, get_spec⌜MkDerTableRow⌝]);
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
val Join_lemma6 = save_pop_thm"Join_lemma6";
```

SML

```
set_goal([], conj1);
a(MAP_EVERY (ante_tac o all_⇒_intro) [
        Join_lemma1,
        Join_lemma2,
        Join_lemma3,
        Join_lemma4,
        Join_lemma5,
        Join_lemma6]
        THEN taut_tac);
val Join_OK_d_lemma = save_pop_thm"Join_OK_d_lemma";
```

SML

```
push_goal([],⌜∀ c tl_0 tl_1 tel
• Elems tel ⊆ OK_TC_d c
      ∧ c dominates lubl (Fst (Split (Map (λ te• te tl_0) tel)))
      ∧ Map (HideDerTable c) tl_0 = Map (HideDerTable c) tl_1
    ⇒ Map (HideDerTable c) (Snd (Split (Map (λ te• te tl_0) tel)))
      = Map (HideDerTable c) (Snd (Split (Map (λ te• te tl_1) tel)))⌝);
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 3 ante_tac THEN DROP_NTH_ASM_T 2 ante_tac);
a(list_induction_tac⌜tel⌝);
(* *** Goal "1" *** *)
a(asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "2" *** *)
a(rewrite_tac[dominates_lub_lemma,lubl_lemma,split_thm,get_spec⌜Map⌝]);
a(REPEAT strip_tac);
```

SML

```
(* *** Goal "3" *** *)
a(PC_T1 "sets_ext" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "4" *** *)
a(REPEAT strip_tac);
a(asm_rewrite_tac[split_thm,get_spec⌜Map⌝]);
a(lemma_tac⌜x ∈ OK_TC_d c⌝);
(* *** Goal "4.1" *** *)
a(PC_T1 "sets_ext" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "4.2" *** *)
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜OK_TC_d⌝]));
a(DROP_NTH_ASM_T 3 (strip_asm_tac o rewrite_rule[dominates_lub_lemma,lubl_lemma,
        split_thm,get_spec⌜Map⌝]));
a(contr_tac THEN all_asm_fc_tac[]);
val AllTuples_lemma1 = save_pop_thm"AllTuples_lemma1";
```

SML

```
push_goal([],⌜∀ c tl0 tl1 rl0 rl1 r0 r1 sl•
        Elems sl ⊆ OK_VCd c ∩ OK_VCc c
        ∧ Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
        ∧ Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
        ∧ HideDerTableRow c r0 = HideDerTableRow c r1
        ⇒ HideDerTableRow c
            (MkDerTableRow (DTR_where r0) (DTR_row r0) (Map (λ e• e tl0 rl0 r0) sl))
        =
        HideDerTableRow c
            (MkDerTableRow (DTR_where r1) (DTR_row r1) (Map (λ e• e tl1 rl1 r1) sl))⌝);
a(REPEAT strip_tac);
a(TOP_ASM_T ante_tac THEN rewrite_tac(MkDerTableRow_lemma ::
        map get_spec[⌜HideDerTableRow⌝,⌜Let⌝,⌜MkDerTableRow⌝])
        THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 7 ante_tac);
a(list_induction_tac ⌜sl⌝);
(* *** Goal "1" *** *)
a(rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(PC_T1 "sets_ext" asm_prove_tac[get_spec⌜Elems⌝]);
```

SML

```
(* *** Goal "3" *** *)
a(REPEAT strip_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
a(lemma_tac⌜x ∈ OK_VCd c ∧ x ∈ OK_VCc c⌝);
(* *** Goal "3.1" *** *)
a(PC_T1 "sets_ext" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "3.2" *** *)
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN
        rewrite_tac(map get_spec[⌜OK_VCd⌝,⌜OK_VCc⌝]));
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(POP_ASM_T (asm_tac o eq_sym_rule));
a(cases_tac⌜c dominates Fst (x tl0 rl0 r0)⌝THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 3 discard_tac);
a(lemma_tac⌜Snd (x tl0 rl0 r0) = Snd (x tl1 rl1 r1)⌝
        THEN1 contr_tac THEN all_asm_fc_tac[]);
a(once_rewrite_tac[prove_rule[]⌜x tl0 rl0 r0 = (Fst (x tl0 rl0 r0), Snd (x tl0 rl0 r0))⌝]);
a(POP_ASM_T pure_rewrite_thm_tac);
a(DROP_NTH_ASM_T 2 (rewrite_thm_tac o eq_sym_rule));
val ProjectData_lemma1 = save_pop_thm"ProjectData_lemma1";
```

SML

```
push_goal([],⌜∀ c rl0 rl1 • Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
        ⇒ HideDerTableData c rl0 = HideDerTableData c rl1⌝);
a(REPEAT ∀_tac);
a(intro_∀_tac(⌜rl1⌝,⌜rl1⌝));
a(list_induction_tac⌜rl0⌝);
(* *** Goal "1" *** *)
a ∀_tac;
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(asm_rewrite_tac[]);
(* *** Goal "1.2" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2" *** *)
a(REPEAT ∀_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "2.1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝,HideDerTableData_lemma]);
a(cases_tac⌜c dominates DTR_row x⌝ THEN
        cases_tac⌜c dominates DTR_row x'⌝ THEN asm_rewrite_tac[]);
```

SML

```
(* *** Goal "2.2.1" *** *)
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
(* *** Goal "2.2.2" *** *)
a(rewrite_tac(MkDerTableRow_lemma :: map get_spec
        [⌜HideDerTableRow⌝,⌜MkDerTableRow⌝,⌜Let⌝]));
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.3" *** *)
a(rewrite_tac(MkDerTableRow_lemma :: map get_spec
        [⌜HideDerTableRow⌝,⌜MkDerTableRow⌝,⌜Let⌝]));
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.4" *** *)
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
val HideDerTableData_lemma1 = save_pop_thm"HideDerTableData_lemma1";
```

4 CONJECTURES FOR *AllTuples* OKNESS PROOFS

4.1 *Where*

SML

```
val Where_conj1 = ⌜∀tl rl c e •
        c dominates lubl (Map DTR_row (Where c tl rl e))⌝;
```

SML

```
val Where_conj2 = ⌜∀ tl0 tl1 rl0 rl1 c e
• e ∈ OK_VCd c ∩ OK_VCc c
      ∧ Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
      ∧ HideDerTableData c rl0 = HideDerTableData c rl1
    ⇒ Map (HideDerTableRow c) (Where c tl0 rl0 e)
      = Map (HideDerTableRow c) (Where c tl1 rl1 e)⌝;
```

4.2 *Group*

We require separate conjectures for the proofs of OK_TC_d and OK_TC_c.

4.2.1 Data OKness Conjecture

SML

```
val Group_conj1 = ⌜∀tl0  tl1 rl0 rl1 c e ml nl •
        e ∈ OK_VCd c ∩ OK_VCc c
∧       c dominates lubl (Map DTR_row rl0)
∧       c dominates lubl (Map DTR_row rl1)
∧       Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
∧       Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
∧       ¬ Map (Map (HideDerTableRow c)) (Snd(Group c tl0 rl0 ml nl e))
        =
        Map (Map (HideDerTableRow c)) (Snd(Group c tl1 rl1 ml nl e))
⇒       ¬ c dominates Fst(Group c tl0 rl0 ml nl e)⌝ ;
```

The specification of *Group* is subdivided to facilitate the proof of the conjecture about *Group*.

HOL Constant

```
GroupA      : DerTableRow LIST
            → ℕ LIST
            → ℕ LIST
            → (Class × (DerTableRow LIST LIST))
```

```
∀rl gbsterling gbclass•
    GroupA rl gbsterling gbclass =
    let     gpby row = (ListNth gbsterling (Map Snd (DTR_cols row)),
                        ListNth gbclass (Map Fst(DTR_cols row)))
    in let  gbc row = lubl(ListNth gbsterling (Map Fst(DTR_cols row)))
    in      (lubl (Map gbc rl), MakeGroups gpby rl)
```

HOL Constant

```
GroupB      : DerTable LIST
            → DerTableRow LIST LIST
            → VALUE_COMP
            → (Class × (DerTableRow LIST LIST))
```

```
∀tl gps having•
    GroupB tl gps having =
    let     has_test gp = ((CommonValue having) tl gp Arbitrary)
    in let  wanted_gps = gps ↾ {gp | ItemBool(Snd (has_test gp))}
    in      (lubl(Map (Fst o has_test) gps), wanted_gps)
```

The following lemma demonstrates that *Group* can be reconstituted from *GroupA* and *GroupB*.

SML

```
set_goal([], ⌜
∀cc tl rl gbsterling gbclass having•
    Group cc tl rl gbsterling gbclass having =
    let     (c1, gps) = GroupA rl gbsterling gbclass
    in let  (c2, wanted_gps) = GroupB tl gps having
    in      ((if cc dominates c1 then c2 else c1), wanted_gps)
⌝);
a(REPEAT strip_tac THEN
    rewrite_tac(map get_spec[⌜Group⌝, ⌜GroupA⌝, ⌜GroupB⌝, ⌜Let⌝]));
val Group_lemma1 = save_pop_thm"Group_lemma1";
```

Next, conjectures about *GroupA* and *GroupB* are given.

SML

```
val GroupA_conj1 = ⌜∀rl0 rl1 c gbsterling gbclass •
        c dominates lubl (Map DTR_row rl0)
∧       c dominates lubl (Map DTR_row rl1)
∧       Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
∧       ¬ Map (Map (HideDerTableRow c)) (Snd(GroupA rl0 gbsterling gbclass))
        =
        Map (Map (HideDerTableRow c)) (Snd(GroupA rl1 gbsterling gbclass))
⇒       ¬ c dominates Fst(GroupA rl0 gbsterling gbclass)⌝ ;
```

SML

```
val GroupB_conj1 = ⌜∀tl0  tl1 gps0 gps1 c having •
        having ∈ OK_VCd c ∩ OK_VCc c
∧       Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
∧       Map (Map (HideDerTableRow c)) gps0 = Map (Map (HideDerTableRow c)) gps1
∧       ¬ Map (Map (HideDerTableRow c)) (Snd(GroupB tl0 gps0 having))
        =
        Map (Map (HideDerTableRow c)) (Snd(GroupB tl1 gps1 having))
⇒       ¬ c dominates Fst(GroupB tl0 gps0 having)⌝ ;
```

Finally, under the assumptions *GroupA_conj* and *GroupB_conj*, *Group_conj* is proven.

SML

```
set_goal([GroupA_conj1, GroupB_conj1], Group_conj1);
a(rewrite_tac[Group_lemma1, let_def, dominates_lub_lemma]
        THEN REPEAT strip_tac);
a(cases_tac⌜c dominates Fst (GroupA rl0 ml nl)⌝ THEN asm_rewrite_tac[]);
a(GET_NTH_ASM_T 10 bc_thm_tac);
a(∃_tac⌜Snd (GroupA rl1 ml nl)⌝ THEN ∃_tac⌜tl1⌝);
a(contr_tac THEN all_asm_fc_tac[]);
val Group_lemma2 = save_pop_thm"Group_lemma2";
```

4.2.2 Classification OKness Conjecture

First, conjectures for *GroupA* and *GroupB*.

SML

```
val GroupA_conj2 = ⌜∀ rl0 rl1 c gbsterling gbclass
          • Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
              ⇒ Fst (GroupA rl0 gbsterling gbclass)
               = Fst (GroupA rl1 gbsterling gbclass)⌝;
```

SML
```
val GroupB_conj2 = ⌜∀ tl0 tl1 gps0 gps1 c having
• having ∈ OK_VC_c c
      ∧ Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
      ∧ Map (Map (HideDerTableRow c)) gps0
        = Map (Map (HideDerTableRow c)) gps1
    ⇒  Fst (GroupB tl0 gps0 having) = Fst (GroupB tl1 gps1 having)⌝;
```

Finally, the overall *Ok3ness* conjecture for *Group*.

SML
```
val Group_conj2 = ⌜∀ tl0 tl1 rl0 rl1 c e ml nl
    • e ∈ OK_VC_c c
          ∧ Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
          ∧ Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
          ⇒ Fst (Group c tl0 rl0 ml nl e) = Fst (Group c tl1 rl1 ml nl e)⌝;
```

4.3 *ProjectData*

SML
```
val ProjectData_conj = ⌜∀tl0  tl1 gps0 gps1 c sl •
        Elems sl ⊆ OK_VC_d c ∩ OK_VC_c c
∧       Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
∧       Map (Map (HideDerTableRow c)) gps0 = Map (Map (HideDerTableRow c)) gps1
⇒
        Map (HideDerTableRow c) (ProjectData tl0 sl gps0)
        =
        Map (HideDerTableRow c) (ProjectData tl1 sl gps1)⌝;
```

5 PROOFS OF *AllTuples* CONJECTURES

5.1 *Where*

Where is first defined in terms of new operations W and H to give a primitive recursive formulation.

HOL Constant

$$
\begin{array}{l}
\quad \mathbf{W} \quad : \mathit{Class} \\
\qquad \rightarrow \mathit{DerTable\ LIST} \\
\qquad \rightarrow \mathit{DerTableRow\ LIST} \\
\qquad \rightarrow \mathit{DerTableRow\ LIST} \\
\qquad \rightarrow \mathit{VALUE\_COMP} \\
\qquad \rightarrow \mathit{DerTableRow\ LIST} \\
\hline
\forall c\ tl\ rl_1\ r\ rl_2\ e \bullet \\
\quad W\ c\ tl\ rl_1\ []\ e = [] \\
\wedge \quad W\ c\ tl\ rl_1\ (\mathit{Cons}\ r\ rl_2)\ e = \\
\quad (\mathit{let}\ w = \mathit{DTR\_where}\ r\ \mathit{lub}\ \mathit{Fst}\ (e\ tl\ rl_1\ r) \\
\quad \mathit{in\ let}\ h = ((\mathit{ItemBool}\ (\mathit{Snd}\ (e\ tl\ rl_1\ r)) \vee \neg\ c\ \mathit{dominates}\ w) \\
\qquad\qquad \wedge\ c\ \mathit{dominates}\ \mathit{DTR\_row}\ r) \\
\quad \mathit{in\ if}\ h\ \mathit{then}\ \mathit{Cons}\ (\mathit{MkDerTableRow}\ w\ (\mathit{DTR\_row}\ r)\ (\mathit{DTR\_cols}\ r)) \\
\qquad\qquad (W\ c\ tl\ rl_1\ rl_2\ e) \\
\qquad \mathit{else}\ (W\ c\ tl\ rl_1\ rl_2\ e))
\end{array}
$$

HOL Constant

$$
\begin{array}{l}
\quad \mathbf{H} \quad : \mathit{Class} \\
\qquad \rightarrow \mathit{DerTableRow\ LIST} \\
\qquad \rightarrow \mathit{DerTableRow\ LIST} \\
\hline
\forall c\ rl \bullet H\ c\ rl = rl \upharpoonright \{r | c\ \mathit{dominates}\ \mathit{DTR\_row}\ r\}
\end{array}
$$

The following lemma relates *Where* to W.

SML

```
push_goal([],⌜∀ c tl rl e • Where c tl rl e = W c tl (H c rl) (H c rl) e⌝);
a(rewrite_tac(map get_spec[⌜Where⌝,⌜W⌝,⌜H⌝,⌜Let⌝]) THEN REPEAT strip_tac);
a(lemma_tac⌜∀ rs •Map
              Snd
              (Map
                   (λ r
                    • ((ItemBool (Snd (e tl (rs ↑ {r|c dominates DTR_row r}) r))
                              ∨ ¬ c
                                  dominates DTR_where r
                                    lub Fst (e tl (rs ↑ {r|c dominates DTR_row r}) r)),
                       MkDerTableRow
                              (DTR_where r
                                  lub Fst (e tl (rs ↑ {r|c dominates DTR_row r}) r))
                              (DTR_row r)
                              (DTR_cols r)))
                   (rl ↑ {r|c dominates DTR_row r})
                 ↑ {(t, r)|t})
          = W c tl (rs ↑ {r|c dominates DTR_row r}) (rl ↑ {r|c dominates DTR_row r}) e⌝
       THEN_LIST[id_tac,asm_rewrite_tac[]]);
```

SML

```
a(list_induction_tac⌜rl⌝);
(* *** Goal "1" *** *)
a(rewrite_tac(map get_spec[⌜W⌝,⌜Map⌝,⌜$↑⌝]));
(* *** Goal "2" *** *)
a(rewrite_tac(map get_spec[⌜W⌝,⌜Map⌝,⌜$↑⌝,⌜Let⌝]));
a(REPEAT strip_tac);
a(cases_tac⌜c dominates DTR_row x⌝THEN asm_rewrite_tac
      (map get_spec[⌜W⌝,⌜Map⌝,⌜$↑⌝,⌜Let⌝]));
a(cases_tac⌜ItemBool (Snd (e tl (rs ↑ {r|c dominates DTR_row r}) x))
                  ∨ ¬ c dominates DTR_where x
                      lub Fst (e tl (rs ↑ {r|c dominates DTR_row r}) x)⌝
      THEN asm_rewrite_tac[get_spec⌜Map⌝]);
val Where_W_lemma = save_pop_thm"Where_W_lemma";
```

The conjectures about *Where* are now proven.

SML

```
push_goal([],Where_conj1);
a(rewrite_tac[Where_W_lemma,get_spec⌜H⌝]THEN REPEAT ∀_tac);
a(lemma_tac⌜∀ rs • c dominates lubl
        (Map DTR_row (W c tl (rs ⍭ {r|c dominates DTR_row r})
                    (rl ⍭ {r|c dominates DTR_row r}) e))⌝
        THEN_LIST[id_tac,asm_rewrite_tac[]]);
a(list_induction_tac⌜rl⌝);
(* *** Goal "1" *** *)
a(rewrite_tac(lubl_lemma :: map get_spec[⌜Map⌝,⌜W⌝,⌜$⍭⌝,⌜$dominates⌝]));
(* *** Goal "2" *** *)
a(rewrite_tac[get_spec⌜$⍭⌝] THEN REPEAT ∀_tac);
a(cases_tac⌜c dominates DTR_row x⌝THEN
        asm_rewrite_tac(map get_spec[⌜Map⌝,⌜W⌝,⌜Let⌝]));
a(cases_tac⌜ItemBool (Snd (e tl (rs ⍭ {r|c dominates DTR_row r}) x))
                        ∨ ¬ c dominates DTR_where x
                         lub Fst (e tl (rs ⍭ {r|c dominates DTR_row r}) x)⌝
        THEN asm_rewrite_tac(lubl_lemma :: dominates_lub_lemma :: map
                get_spec[⌜Map⌝,⌜MkDerTableRow⌝]));
val Where_dominates_lemma = save_pop_thm"Where_dominates_lemma";
```

Before a proof of $Where_conj2$ is given, some subsidiary lemmas are proven.

SML

```
push_goal([],⌜∀ tl0 tl1 rl0 rl1 r0 r1 c e
• e ∈ OK_VCd c ∩ OK_VCc c
    ∧ Map (HideDerTable c) tl0 = Map (HideDerTable c) tl1
    ∧ Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
    ∧ HideDerTableRow c r0 = HideDerTableRow c r1
  ⇒ (((ItemBool (Snd (e tl0 rl0 r0)) ∨ ¬ c dominates DTR_where r0 lub Fst (e tl0 rl0 r0))
        ∧ c dominates DTR_row r0) ⇔
      ((ItemBool (Snd (e tl1 rl1 r1)) ∨ ¬ c dominates DTR_where r1 lub Fst (e tl1 rl1 r1))
        ∧ c dominates DTR_row r1))⌝);
a(REPEAT ∀_tac THEN ⇒_tac);
a(GET_NTH_ASM_T 1 ante_tac THEN
        rewrite_tac(MkDerTableRow_lemma ::
        map get_spec[⌜HideDerTableRow⌝,⌜Let⌝,⌜MkDerTableRow⌝])      THEN ⇒_tac);
```

SML

```
a(POP_ASM_T discard_tac);
a(DROP_NTH_ASM_T 7 ante_tac THEN DROP_NTH_ASM_T 6 ante_tac THEN
        rewrite_tac(map get_spec[⌜OK_VC_d⌝,⌜OK_VC_c⌝]));
a(⇒_tac THEN all_asm_fc_tac[]);
a(DROP_NTH_ASM_T 2 discard_tac THEN ⇒_tac);
a(cases_tac⌜c dominates Fst (e tl_0 rl_0 r_0)⌝);
(* *** Goal "1" *** *)
a(list_spec_nth_asm_tac 2 [⌜tl_0⌝,⌜tl_1⌝,⌜rl_0⌝,⌜rl_1⌝,⌜r_0⌝,⌜r_1⌝]);
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(lemma_tac⌜¬ c dominates Fst (e tl_1 rl_1 r_1)⌝);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 3 (asm_rewrite_thm_tac o eq_sym_rule));
(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[dominates_lub_lemma]THEN REPEAT strip_tac);
val Where_lemma = save_pop_thm"Where_lemma";
```

SML

```
push_goal([],⌜∀tl_0  tl_1 rl_0 rl_1  rl1 rl2 c e •
        e ∈ OK_VC_d c ∩ OK_VC_c c
∧       Map (HideDerTable c) tl_0 = Map (HideDerTable c) tl_1
∧       Map (HideDerTableRow c) rl_0 = Map (HideDerTableRow c) rl_1
∧       Map (HideDerTableRow c) rl1 = Map (HideDerTableRow c) rl2
        ⇒
        Map (HideDerTableRow c) (W c tl_0 rl1 rl_0 e) =
        Map (HideDerTableRow c) (W c tl_1 rl2 rl_1 e)⌝);
a(REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
a(intro_∀_tac(⌜rl1⌝,⌜rl1⌝) THEN intro_∀_tac(⌜rl2⌝,⌜rl2⌝) THEN intro_∀_tac(⌜rl_1⌝,⌜rl_1⌝));
```

SML

```
a(list_induction_tac⌜rl_0⌝);
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl_1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[get_spec⌜W⌝]);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
a(REPEAT strip_tac);
a(rewrite_tac(map get_spec[⌜W⌝,⌜Let⌝]));
a(CASES_T⌜(ItemBool (Snd (e tl0 rl1 x)) ∨
        ¬ c dominates DTR_where x lub Fst (e tl0 rl1 x))
                ∧ c dominates DTR_row x⌝asm_tac);
```

SML

```
(* *** Goal "2.2.1" *** *)
a(LEMMA_T⌜(ItemBool (Snd (e tl1 rl2 x')) ∨
        ¬ c dominates DTR_where x' lub Fst (e tl1 rl2 x'))
                ∧ c dominates DTR_row x'⌝rewrite_thm_tac);
(* *** Goal "2.2.1.1" *** *)
a(LEMMA_T⌜e ∈ OK_VC_d c ∩ OK_VC_c c⌝asm_tac THEN_LIST[REPEAT strip_tac,id_tac]);
a(ante_tac(list_∀_elim[⌜tl0⌝,⌜tl1⌝,⌜rl1⌝,⌜rl2⌝,⌜x⌝,⌜x'⌝,⌜c⌝,⌜e⌝]Where_lemma)
        THEN asm_rewrite_tac[]);
(* *** Goal "2.2.1.2" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[get_spec⌜Map⌝]);
a(all_asm_fc_tac[]);
a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T discard_tac);
a(GET_NTH_ASM_T 2 ante_tac THEN rewrite_tac
        (MkDerTableRow_lemma :: map get_spec[⌜HideDerTableRow⌝,⌜Let⌝,⌜MkDerTableRow⌝])
        THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 10 ante_tac  THEN rewrite_tac[get_spec⌜OK_VC_c⌝]);
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(asm_rewrite_tac[]);
```

SML
```
(* *** Goal "2.2.2" *** *)
a(LEMMA_T⌜¬((ItemBool (Snd (e tl1 rl2 x′)) ∨
        ¬ c dominates DTR_where x′ lub Fst (e tl1 rl2 x′))
                ∧ c dominates DTR_row x′)⌝rewrite_thm_tac);
(* *** Goal "2.2.2.1" *** *)
a(LEMMA_T⌜e ∈ OK_VCd c ∩ OK_VCc c⌝asm_tac THEN_LIST[REPEAT strip_tac,id_tac]);
a(ante_tac(list_∀_elim[⌜tl0⌝,⌜tl1⌝,⌜rl1⌝,⌜rl2⌝,⌜x⌝,⌜x′⌝,⌜c⌝,⌜e⌝]Where_lemma)
        THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(all_asm_fc_tac[]);
val W_lemma = save_pop_thm"W_lemma";
```

SML
```
push_goal([],Where_conj2);
a(REPEAT strip_tac);
a(rewrite_tac[Where_W_lemma]);
a(bc_tac[W_lemma]THEN_TRY PC_T1 "sets_ext" asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac(map get_spec[⌜Let⌝,⌜HideDerTableData⌝,⌜H⌝]));
val Where_OKd_lemma = save_pop_thm"Where_OKd_lemma";
```

5.2 *Group*

First, some lemmas about *MakeGroups* and the columns of a *DTR_row*.

SML
```
push_goal([],⌜∀ xs gpby • ¬ [] ∈ Elems (MakeGroups gpby xs)⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜xs⌝);
(* *** Goal "1" *** *)
a(rewrite_tac(map get_spec[⌜Elems⌝,⌜MakeGroups⌝]));
(* *** Goal "2" *** *)
a(strip_tac THEN rewrite_tac(map get_spec[⌜Elems⌝,⌜MakeGroups⌝]));
a(lemma_tac⌜∃ gps • MakeGroups gpby xs = gps⌝THEN1 prove_∃_tac);
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
a(POP_ASM_T discard_tac);
a(list_induction_tac⌜gps⌝);
```

SML

```
(* *** Goal "2.1" *** *)
a(rewrite_tac(map get_spec[⌜Elems⌝,⌜PutInGroup⌝]));
(* *** Goal "2.2" *** *)
a(PC_T1 "sets_ext" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "2.3" *** *)
a(rewrite_tac(map get_spec[⌜Elems⌝,⌜PutInGroup⌝]));
a(REPEAT strip_tac);
a(cases_tac⌜gpby x = gpby (Head x')⌝THEN asm_rewrite_tac[]);
(* *** Goal "2.3.1" *** *)
a(asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "2.3.2" *** *)
a(asm_prove_tac[get_spec⌜Elems⌝]);
val MakeGroups_lemma1 = save_pop_thm"MakeGroups_lemma1";
```

SML

```
set_goal([],⌜∀ r1 r2 c •
        HideDerTableRow c r1 = HideDerTableRow c r2
     ⇒ Map Fst (DTR_cols r1) =  Map Fst (DTR_cols r2)⌝);
a(REPEAT strip_tac);
a(POP_ASM_T  (strip_asm_tac o rewrite_rule(MkDerTableRow_lemma :: map get_spec
        [⌜HideDerTableRow⌝,⌜Let⌝,⌜MkDerTableRow⌝]))));
a(LIST_DROP_NTH_ASM_T [2,3](MAP_EVERY discard_tac));
a(POP_ASM_T ante_tac);
a(lemma_tac⌜∃l1 • DTR_cols r1 = l1⌝THEN1 prove_∃_tac);
a(lemma_tac⌜∃l2 • DTR_cols r2 = l2⌝THEN1 prove_∃_tac);
a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T rewrite_thm_tac );
a(intro_∀_tac(⌜l2⌝,⌜l2⌝));
a(list_induction_tac⌜l1⌝);
```

SML

```
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l2⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l2⌝list_cases_thm));
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T ante_tac);
a(rewrite_tac[get_spec⌜Map⌝]);
a(REPEAT ⇒_tac);
a(all_asm_fc_tac[]);
a(asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 3 ante_tac);
a(cases_tac⌜c dominates Fst x⌝THEN cases_tac⌜c dominates Fst x'⌝THEN asm_rewrite_tac[]);
(* *** Goal "2.2.1" *** *)
a(⇒_T rewrite_thm_tac);
(* *** Goal "2.2.2" *** *)
a(⇒_T (strip_asm_tac o once_rewrite_rule[prove_rule[]⌜x = (Fst x,Snd x)⌝]));
(* *** Goal "2.2.3" *** *)
a(⇒_T (strip_asm_tac o once_rewrite_rule[prove_rule[]⌜x' = (Fst x',Snd x')⌝]));
val GroupA_cols_lemma1 = save_pop_thm"GroupA_cols_lemma1";
```

SML

```
push_goal([],⌜∀ r1 r2 c gbsterling gbclass •
        c dominates lubl (ListNth gbsterling (Map Fst (DTR_cols r1)))
∧       HideDerTableRow c r1 = HideDerTableRow c r2
      ⇒ ListNth gbsterling (Map Snd (DTR_cols r1))
          = ListNth gbsterling (Map Snd (DTR_cols r2))⌝);
a(REPEAT strip_tac);
a(TOP_ASM_T  (strip_asm_tac o rewrite_rule(MkDerTableRow_lemma :: map get_spec
        [⌜HideDerTableRow⌝,⌜Let⌝,⌜MkDerTableRow⌝])));
a(all_asm_fc_tac[HideDerTableRow_Length_lemma]);
a(all_asm_fc_tac[GroupA_cols_lemma1]);
a(LIST_DROP_NTH_ASM_T [4,5,6](MAP_EVERY discard_tac));
a(DROP_NTH_ASM_T 4 ante_tac);
a(LIST_INDUCTION_T⌜gbsterling⌝asm_tac);
```

SML

```
(* *** Goal "1" *** *)
a(rewrite_tac[get_spec⌜ListNth⌝]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[get_spec⌜ListNth⌝,length_map_lemma,lubl_lemma,dominates_lub_lemma]);
a(∀_tac THEN CASES_T⌜1 ≤ x ∧ x ≤ # (DTR_cols r2)⌝asm_tac THEN
        asm_rewrite_tac[]);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
a(REPEAT strip_tac THEN_TRY all_asm_fc_tac[]);
a(lemma_tac⌜x ≤ # (DTR_cols r1)⌝THEN1 asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 5 (strip_asm_tac));
a(lemma_tac⌜c dominates Nth (Map Fst (DTR_cols r1)) x⌝THEN1 asm_rewrite_tac[]);
```

SML

```
a(strip_asm_tac(list_∀_elim[⌜DTR_cols r1⌝,⌜x⌝,
        ⌜Fst: Class × (ValuedItem + ONE) → Class⌝]fun_nth_map_lemma));
a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T (rewrite_thm_tac o eq_sym_rule)
        THEN ⇒_tac);
a(strip_asm_tac(list_∀_elim[⌜DTR_cols r2⌝,⌜x⌝,
        ⌜Fst: Class × (ValuedItem + ONE) → Class⌝]fun_nth_map_lemma));
a(DROP_NTH_ASM_T 7 ante_tac THEN POP_ASM_T (rewrite_thm_tac o eq_sym_rule)
        THEN ⇒_tac);
a(strip_asm_tac(list_∀_elim[⌜DTR_cols r1⌝,⌜x⌝,
        ⌜Snd: Class × (ValuedItem + ONE) → ValuedItem + ONE⌝]fun_nth_map_lemma));
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
a(strip_asm_tac(list_∀_elim[⌜DTR_cols r2⌝,⌜x⌝,
        ⌜Snd: Class × (ValuedItem + ONE) → ValuedItem + ONE⌝]fun_nth_map_lemma));
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
```

SML

```
a(LIST_DROP_NTH_ASM_T [1,2,3,5,9,10](MAP_EVERY ante_tac));
a(lemma_tac⌜∃l1 • DTR_cols r1 = l1⌝THEN1 prove_∃_tac);
a(lemma_tac⌜∃l2 • DTR_cols r2 = l2⌝THEN1 prove_∃_tac);
a(POP_ASM_T rewrite_thm_tac THEN POP_ASM_T rewrite_thm_tac );
a(LIST_DROP_NTH_ASM_T [2,3,4](MAP_EVERY discard_tac));
a(POP_ASM_T ante_tac);
a(intro_∀_tac(⌜l2⌝,⌜l2⌝));
a(intro_∀_tac(⌜x⌝,⌜x⌝));
a(list_induction_tac⌜l1⌝);
```

SML

```
(* *** Goal "2.1.1" *** *)
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 4(strip_asm_tac o rewrite_rule[length_def]));
a(DROP_NTH_ASM_T 7 ante_tac THEN POP_ASM_T  rewrite_thm_tac);
(* *** Goal "2.1.2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l2⌝list_cases_thm));
(* *** Goal "2.1.2.1" *** *)
a(DROP_NTH_ASM_T 6 ante_tac THEN asm_rewrite_tac[get_spec⌜Length⌝]);
(* *** Goal "2.1.2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(LIST_DROP_NTH_ASM_T [1,2,3,4,5,6](MAP_EVERY ante_tac));
a(rewrite_tac(map get_spec[⌜Length⌝,⌜Map⌝,⌜Nth⌝]));
a(cases_tac⌜x'=1⌝THEN asm_rewrite_tac[]);
```

SML

```
(* *** Goal "2.1.2.2.1" *** *)
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[]);
a(⇒_T rewrite_thm_tac);
(* *** Goal "2.1.2.2.2" *** *)
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 10 (ante_tac o list_∀_elim[⌜x'-1⌝,⌜list2⌝]));
a(asm_rewrite_tac[]);
a(GET_ASM_T ⌜1 ≤ x'⌝ (strip_asm_tac o rewrite_rule[get_spec⌜$≤⌝]));
a(var_elim_nth_asm_tac 1);
a(asm_rewrite_tac[]);
a(strip_asm_tac (∀_elim⌜i⌝ ℕ_cases_thm));
a(POP_ASM_T rewrite_thm_tac);
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[] THEN
        REPEAT strip_tac THEN all_asm_fc_tac[]);
val GroupA_cols_lemma2 = save_pop_thm"GroupA_cols_lemma2";
```

SML

```
push_goal([],⌜∀ rl0 rl1 g •
        Map g rl0 = Map g rl1
⇒       Map (Map g) (MakeGroups g rl0) = Map (Map g) (MakeGroups g rl1)⌝);
a ∀_tac;
a(list_induction_tac⌜rl0⌝);
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T ante_tac THEN
        rewrite_tac[get_spec⌜Map⌝,get_spec⌜MakeGroups⌝]);
a(REPEAT ⇒_tac);
a(all_asm_fc_tac[]);
a(DROP_NTH_ASM_T 4 discard_tac);
```

SML

```
a(lemma_tac⌜∃l0 • MakeGroups g rl0 = l0⌝ THEN1 prove_∃_tac);
a(lemma_tac⌜∃l1 • MakeGroups g list2 = l1⌝ THEN1 prove_∃_tac);
a(ante_tac(list_∀_elim[⌜rl0⌝,⌜g⌝]MakeGroups_lemma1));
a(ante_tac(list_∀_elim[⌜list2⌝,⌜g⌝]MakeGroups_lemma1));
a(DROP_NTH_ASM_T 3 ante_tac THEN
        POP_ASM_T rewrite_thm_tac THEN POP_ASM_T rewrite_thm_tac);
a(intro_∀_tac(⌜l1⌝,⌜l1⌝));
a(list_induction_tac⌜l0⌝);
(* *** Goal "2.2.1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l1⌝list_cases_thm));
(* *** Goal "2.2.1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[get_spec⌜PutInGroup⌝,get_spec⌜Map⌝]);
a(asm_rewrite_tac[]);
(* *** Goal "2.2.1.2" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2.2.2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l1⌝list_cases_thm));
(* *** Goal "2.2.2.1" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2.2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(LIST_DROP_NTH_ASM_T [1,2,3](MAP_EVERY ante_tac) THEN
        rewrite_tac(map get_spec[⌜Map⌝,⌜Elems⌝,⌜PutInGroup⌝]));
a(REPEAT strip_tac);
a(all_asm_fc_tac[]);
a(strip_asm_tac(∀_elim⌜x''⌝list_cases_thm));
(* *** Goal "2.2.2.2.1" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2" *** *)
a(strip_asm_tac(∀_elim⌜x'''⌝list_cases_thm));
```

SML

```
(* *** Goal "2.2.2.2.2.1" *** *)
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(var_elim_nth_asm_tac 1);
a(LIST_DROP_NTH_ASM_T [2,3,4,5,8](MAP_EVERY discard_tac));
a(DROP_NTH_ASM_T 3 (strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
a(rewrite_tac[]);
a(cases_tac⌜g x = g x''''⌝THEN cases_tac⌜g x' = g x''''⌝
        THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2.2.2.2.2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
val MakeGroups_lemma2 = save_pop_thm"MakeGroups_lemma2";
```

SML

```
push_goal([],⌜∀ rl0 rl1 g h •
        Map g rl0 = Map g rl1
∧       Map h rl0 = Map h rl1
⇒       Map (Map h) (MakeGroups g rl0) = Map (Map h) (MakeGroups g rl1)⌝);
a ∀_tac;
a(list_induction_tac⌜rl0⌝);
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T ante_tac THEN  POP_ASM_T ante_tac THEN
        rewrite_tac[get_spec⌜Map⌝,get_spec⌜MakeGroups⌝]);
a(REPEAT ⇒_tac);
a(all_asm_fc_tac[]);
a(DROP_NTH_ASM_T 6 discard_tac);
a(lemma_tac⌜Map (Map g) (MakeGroups g rl0) = Map (Map g) (MakeGroups g list2)⌝
        THEN1 all_asm_fc_tac[MakeGroups_lemma2]);
a(lemma_tac⌜∃l0 • MakeGroups g rl0 = l0⌝THEN1 prove_∃_tac);
a(lemma_tac⌜∃l1 • MakeGroups g list2 = l1⌝THEN1 prove_∃_tac);
```

SML

```
a(ante_tac(list_∀_elim[⌜rl0⌝,⌜g⌝]MakeGroups_lemma1));
a(ante_tac(list_∀_elim[⌜list2⌝,⌜g⌝]MakeGroups_lemma1));
a(DROP_NTH_ASM_T 4 ante_tac THEN DROP_NTH_ASM_T 3 ante_tac THEN
        POP_ASM_T rewrite_thm_tac THEN POP_ASM_T rewrite_thm_tac);
a(intro_∀_tac(⌜l1⌝,⌜l1⌝));
a(list_induction_tac⌜l0⌝);
(* *** Goal "2.2.1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l1⌝list_cases_thm));
(* *** Goal "2.2.1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[get_spec⌜PutInGroup⌝,get_spec⌜Map⌝]);
a(asm_rewrite_tac[]);
(* *** Goal "2.2.1.2" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2.2.2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜l1⌝list_cases_thm));
(* *** Goal "2.2.2.1" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2.2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(LIST_DROP_NTH_ASM_T [1,2,3,4](MAP_EVERY ante_tac) THEN
        rewrite_tac(map get_spec[⌜Map⌝,⌜Elems⌝,⌜PutInGroup⌝]));
a(REPEAT strip_tac);
a(all_asm_fc_tac[]);
a(strip_asm_tac(∀_elim⌜x''⌝list_cases_thm));
(* *** Goal "2.2.2.2.1" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2" *** *)
a(strip_asm_tac(∀_elim⌜x'''⌝list_cases_thm));
```

SML

```
(* *** Goal "2.2.2.2.2.1" *** *)
a(DROP_NTH_ASM_T 7 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(var_elim_nth_asm_tac 1);
a(LIST_DROP_NTH_ASM_T [2,3,4,5,10](MAP_EVERY discard_tac));
a(DROP_NTH_ASM_T 3 (strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
a(DROP_NTH_ASM_T 6 (strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
a(rewrite_tac[]);
a(cases_tac⌜g x = g x''''⌝THEN cases_tac⌜g x' = g x''''⌝
        THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2.2.2.2.2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2.2.2.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
val MakeGroups_lemma3 = save_pop_thm"MakeGroups_lemma3";
```

SML

```
push_goal([],⌜∀ rl0 rl1 c gbsterling gbclass •
        c dominates lubl (Map (λ row• lubl
                (ListNth gbsterling (Map Fst (DTR_cols row)))) rl0)
∧       Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
⇒       Map (λ row • (ListNth gbsterling (Map Snd (DTR_cols row)),
                ListNth gbclass (Map Fst (DTR_cols row)))) rl0 =
        Map (λ row • (ListNth gbsterling (Map Snd (DTR_cols row)),
                ListNth gbclass (Map Fst (DTR_cols row)))) rl1⌝);
a(REPEAT strip_tac);
a(LIST_DROP_NTH_ASM_T [1,2](MAP_EVERY ante_tac)
        THEN intro_∀_tac(⌜rl1⌝,⌜rl1⌝));
a(list_induction_tac⌜rl0⌝);
```

SML

```
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
```

SML

```
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(LIST_DROP_NTH_ASM_T [1,2](MAP_EVERY ante_tac) THEN
        rewrite_tac[get_spec⌜Map⌝,lubl_lemma,dominates_lub_lemma]);
a(REPEAT ⇒_tac);
a(all_asm_fc_tac[]);
a(DROP_NTH_ASM_T 6 discard_tac);
a(POP_ASM_T rewrite_thm_tac);
a(all_asm_fc_tac[GroupA_cols_lemma1]);
a(all_asm_fc_tac[GroupA_cols_lemma2]);
a(asm_rewrite_tac[]);
val GroupA_lemma = save_pop_thm"GroupA_lemma";
```

5.2.1 Data OKness lemmas

First, proofs of *GroupA_conj1* and *GroupB_conj1*.

SML
```
push_goal([], GroupA_conj1);
a(rewrite_tac[get_spec⌜GroupA⌝, let_def]
        THEN REPEAT strip_tac);
a(swap_nth_asm_concl_tac 1);
a(all_asm_fc_tac[GroupA_lemma]);
a(bc_tac[MakeGroups_lemma3]THEN_TRY asm_rewrite_tac[]);
val GroupA_OK_d_lemma = save_pop_thm"GroupA_OK_d_lemma";
```

SML
```
push_goal([], GroupB_conj1);
a(rewrite_tac[get_spec⌜GroupB⌝, let_def]
        THEN REPEAT strip_tac);
a(swap_nth_asm_concl_tac 1);
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
a(intro_∀_tac(⌜gps1⌝,⌜gps1⌝));
a(list_induction_tac⌜gps0⌝);
```

SML
```
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜gps1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(asm_rewrite_tac[]THEN rewrite_tac[get_spec⌜$↾⌝]);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜gps1⌝list_cases_thm));
```

SML

```
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
a(rewrite_tac(lubl_lemma :: dominates_lub_lemma :: map get_spec[⌜Map⌝,⌜$|⌝]));
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(all_asm_fc_tac[CommonValue_OK_d_lemma]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜OK_VC_d⌝]));
a(all_asm_fc_tac[CommonValue_OK_c_lemma]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜OK_VC_c⌝]));
a(LEMMA_T⌜HideDerTableRow c Arbitrary = HideDerTableRow c Arbitrary⌝asm_tac
        THEN_LIST[rewrite_tac[],all_asm_fc_tac[]]);
a(lemma_tac⌜Snd (CommonValue having tl_0 x Arbitrary)
        = Snd (CommonValue having tl_1 x' Arbitrary)⌝
        THEN1 contr_tac THEN all_asm_fc_tac[]);
a(cases_tac⌜ItemBool (Snd (CommonValue having tl_1 x' Arbitrary))⌝
        THEN asm_rewrite_tac[get_spec⌜Map⌝]);
val GroupB_OK_d_lemma = save_pop_thm"GroupB_OK_d_lemma";
```

Finally, a proof of the Data OKness *Group* conjecture, the main proof of this section.

SML

```
push_goal([], Group_conj1);
a(MAP_EVERY (ante_tac o all_⇒_intro) [
        Group_lemma2,
        GroupA_OK_d_lemma,
        GroupB_OK_d_lemma]
        THEN taut_tac);
val Group_OK_d_lemma = save_pop_thm"Group_OK_d_lemma";
```

5.2.2 Classification OKness lemmas

First, proofs of *GroupA_conj2* and *GroupB_conj2*.

SML

```
push_goal([],GroupA_conj2);
a(rewrite_tac[get_spec⌜GroupA⌝, let_def]
        THEN REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN intro_∀_tac(⌜rl_1⌝,⌜rl_1⌝));
a(list_induction_tac⌜rl_0⌝);
```

SML

```
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
(* *** Goal "1.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝list_cases_thm));
```

SML

```
(* *** Goal "2.1" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T ante_tac THEN
        rewrite_tac[dominates_lub_lemma,lubl_lemma,get_spec⌜Map⌝]);
a(REPEAT strip_tac);
a(all_asm_fc_tac[]);
a(all_asm_fc_tac[GroupA_cols_lemma1]);
a(asm_rewrite_tac[]);
val GroupA_OKc_lemma = save_pop_thm"GroupA_OKc_lemma";
```

SML

```
push_goal([],GroupB_conj2);
a(rewrite_tac[get_spec⌜GroupB⌝, let_def]
        THEN REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN intro_∀_tac(⌜gps1⌝,⌜gps1⌝));
a(list_induction_tac⌜gps0⌝);
```

SML

```
(* *** Goal "1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜gps_1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "1.2" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜gps_1⌝list_cases_thm));
```

SML

```
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(POP_ASM_T ante_tac);
a(rewrite_tac[lubl_lemma,dominates_lub_lemma,get_spec⌜Map⌝]);
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(all_asm_fc_tac[CommonValue_OK_c_lemma]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜OK_VC_c⌝]));
a(LEMMA_T⌜HideDerTableRow c Arbitrary = HideDerTableRow c Arbitrary⌝asm_tac
        THEN_LIST[rewrite_tac[],all_asm_fc_tac[]]);
a(asm_rewrite_tac[]);
val GroupB_OK_c_lemma = save_pop_thm"GroupB_OK_c_lemma";
```

Finally, a proof of the Classification OKness *Group* conjecture, the main proof of this section.

SML

```
push_goal([],Group_conj2);
a(rewrite_tac[Group_lemma1, let_def]THEN REPEAT strip_tac);
a(all_asm_fc_tac[GroupA_OK_c_lemma]);
a(lemma_tac⌜Fst (GroupA rl_1 ml nl) = Fst (GroupA rl_0 ml nl)⌝ THEN1 asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 2 discard_tac);
a(cases_tac⌜c dominates Fst (GroupA rl_0 ml nl)⌝ THEN asm_rewrite_tac[]);
a(bc_tac[GroupB_OK_c_lemma] THEN_TRY asm_rewrite_tac[]);
a(∃_tac⌜c⌝THEN asm_rewrite_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac(map get_spec[⌜Let⌝,⌜GroupA⌝]));
a(strip_tac THEN bc_tac[MakeGroups_lemma3] THEN_TRY asm_rewrite_tac[]);
a(all_asm_fc_tac[GroupA_lemma]);
a(asm_rewrite_tac[]);
val Group_OK_c_lemma = save_pop_thm"Group_OK_c_lemma";
```

5.3 *ProjectData*

SML

```
push_goal([],ProjectData_conj);
a(REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN intro_∀_tac(⌜gps1⌝,⌜gps1⌝));
a(list_induction_tac⌜gps0⌝);
(* *** Goal "1" *** *)
a ∀_tac;
a(strip_asm_tac (∀_elim⌜gps1⌝list_cases_thm));
(* *** Goal "1.1" *** *)
a(asm_rewrite_tac(map get_spec[⌜Let⌝,⌜ProjectData⌝,⌜Map⌝]));
(* *** Goal "1.2" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2" *** *)
a(REPEAT ∀_tac);
a(strip_asm_tac (∀_elim⌜gps1⌝list_cases_thm));
(* *** Goal "2.1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(⇒_T (strip_asm_tac o rewrite_rule(map get_spec[⌜Map⌝])));
a(all_asm_fc_tac[]);
a(rewrite_tac[ProjectData_lemma]);
```

SML

```
a(lemma_tac⌜∀ x1 x2 x' •
        Map (HideDerTableRow c) x = Map (HideDerTableRow c) x'
∧       Map (HideDerTableRow c) x1 = Map (HideDerTableRow c) x2
⇒
        Map
              (HideDerTableRow c)
              (Map (λ r• MkDerTableRow (DTR_where r) (DTR_row r)
                        (Map (λ e• e tl_0 x1 r) sl)) x
                 ⌢ ProjectData tl_0 sl gps_0)
            = Map
              (HideDerTableRow c)
              (Map (λ r• MkDerTableRow (DTR_where r) (DTR_row r)
                        (Map (λ e• e tl_1 x2 r) sl)) x'
                 ⌢ ProjectData tl_1 sl list2)⌝THEN_LIST[id_tac, all_asm_fc_tac[]]);
a(DROP_NTH_ASM_T 3 discard_tac);
a(list_induction_tac⌜x⌝);
```

SML

```
(* *** Goal "2.2.1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac (∀_elim⌜x'⌝list_cases_thm));
(* *** Goal "2.2.1.1" *** *)
a(asm_rewrite_tac(map get_spec[⌜Map⌝,⌜$⌢⌝]));
(* *** Goal "2.2.1.2" *** *)
a(var_elim_nth_asm_tac 1);
a(DROP_NTH_ASM_T 2(strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
(* *** Goal "2.2.2" *** *)
a(REPEAT ∀_tac);
a(strip_asm_tac(∀_elim⌜x''⌝list_cases_thm));
(* *** Goal "2.2.2.1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2.2.2.2" *** *)
a(asm_rewrite_tac(map get_spec[⌜Map⌝,⌜$⌢⌝]));
a(REPEAT strip_tac THEN all_asm_fc_tac[ProjectData_lemma1]);
val ProjectData_OK_d_lemma = save_pop_thm"ProjectData_OK_d_lemma";
```

6 DATA OKNESS PROOFS

6.1 *TableContents*

SML

```
set_goal([], ⌜∀c i• TableContents i ∈ OK_TC_d c⌝);
a(rewrite_tac(map get_spec[⌜OK_TC_d⌝, ⌜TableContents⌝])
        THEN REPEAT strip_tac);
a(POP_ASM_T ante_tac);
a(lemma_tac⌜# (Map (HideDerTable c) tl_1)
            = # (Map (HideDerTable c) tl_0)⌝
        THEN1 asm_rewrite_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule [length_map_thm]));
a(cases_tac ⌜1 ≤ i ∧ i ≤ # tl_0⌝ THEN asm_rewrite_tac[]);
a(REPEAT strip_tac);
a(all_fc_tac[fun_nth_map_lemma1]);
val TableContents_OK_d_lemma = save_pop_thm"TableContents_OK_d_lemma";
```

6.2 *AllTuples*

SML

```
push_goal([], ⌜∀c esl tel e1 ml nl e2 •
        Elems (Map Fst esl) ⊆ OK_VC_d c ∩ OK_VC_c c ∧
        Elems tel ⊆ OK_TC_d c ∧
        e1 ∈ OK_VC_d c ∩ OK_VC_c c∧
        e2 ∈ OK_VC_d c ∩ OK_VC_c c ⇒
        AllTuples c esl tel e1 ml nl e2 ∈ OK_TC_d c⌝);
a(rewrite_tac[get_spec⌜OK_TC_d⌝]);
a(REPEAT strip_tac);
a(swap_nth_asm_concl_tac 1);
a(lemma_tac⌜c dominates lubl (Fst (Split (Map (λ te• te tl_0) tel)))⌝);
(* *** Goal "1" *** *)
a(swap_nth_asm_concl_tac 1);
a(asm_rewrite_tac(map get_spec [⌜Let⌝,⌜AllTuples⌝]));
```

SML

```
(* *** Goal "2" *** *)
a(lemma_tac⌜Map
              (HideDerTable c)
              (Snd (Split (Map (λ te• te tl0) tel)))
            = Map
              (HideDerTable c)
              (Snd (Split (Map (λ te• te tl1) tel)))⌝
        THEN1 bc_tac[AllTuples_lemma1]THEN_TRY asm_rewrite_tac[]);
a(all_fc_tac[Join_OKd_lemma]);
a(POP_ASM_T discard_tac);
a(POP_ASM_T ante_tac);
a(POP_ASM_T discard_tac);
a(DROP_NTH_ASM_T 2 ante_tac);
a(asm_rewrite_tac(MkDerTable_lemma :: map get_spec
        [⌜AllTuples⌝,⌜Let⌝,⌜HideDerTable⌝,⌜Project⌝,⌜DT_spec⌝]));
a(REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
```

SML

```
a(lemma_tac⌜Map
              (HideDerTableRow c)
              (Where c tl0 (Snd (Join (Snd (Split (Map (λ te• te tl0) tel))))) e1)
            = Map
              (HideDerTableRow c)
              (Where c tl1 (Snd (Join (Snd (Split (Map (λ te• te tl1) tel))))) e1)⌝
         THEN_LIST[all_asm_fc_tac[Where_OKd_lemma],id_tac]);
a(lemma_tac⌜c dominates lubl (Map DTR_row
        (Where c tl0 (Snd (Join (Snd (Split (Map (λ te• te tl0) tel))))) e1))
∧       c dominates lubl (Map DTR_row
        (Where c tl1 (Snd (Join (Snd (Split (Map (λ te• te tl1) tel))))) e1))⌝
                THEN1 rewrite_tac[Where_dominates_lemma]);
```

SML

```
a(lemma_tac⌜Map (Map (HideDerTableRow c))
 (Snd (Group c tl0 (Where c tl0 (Snd (Join (Snd (
        Split (Map (λ te• te tl0) tel))))) e1) ml nl e2))
            = Map (Map (HideDerTableRow c))
 (Snd (Group c tl1 (Where c tl1 (Snd (Join (Snd (
        Split (Map (λ te• te tl1) tel))))) e1) ml nl e2))⌝
        THEN1 contr_tac THEN all_asm_fc_tac[Group_OKd_lemma]);
a(all_asm_fc_tac[ProjectData_OKd_lemma]);
a(bc_tac[HideDerTableData_lemma1]THEN asm_rewrite_tac[]);
val AllTuples_OKd_lemma = save_pop_thm"AllTuples_OKd_lemma";
```

7 CLASSIFICATION OKNESS PROOFS

7.1 *TableContents*

SML

```
set_goal([], ⌜∀c i• TableContents i ∈ OK_TC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_TC_c⌝, ⌜TableContents⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜# (Map (HideDerTable c) tl_1)
            = # (Map (HideDerTable c) tl_0)⌝
        THEN1 asm_rewrite_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule [length_map_thm]));
a(cases_tac ⌜1 ≤ i ∧ i ≤ # tl_0⌝ THEN asm_rewrite_tac[]);
val TableContents_OK_c_lemma = save_pop_thm"TableContents_OK_c_lemma";
```

7.2 *AllTuples*

SML

```
push_goal([],⌜∀ c tl_0 tl_1 tel
• Elems tel ⊆ OK_TC_c c
      ∧ Map (HideDerTable c) tl_0 = Map (HideDerTable c) tl_1
    ⇒ lubl (Fst (Split (Map (λ te• te tl_0) tel)))
      = lubl (Fst (Split (Map (λ te• te tl_1) tel)))⌝);
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 2 ante_tac);
a(list_induction_tac⌜tel⌝);
(* *** Goal "1" *** *)
a(asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "2" *** *)
a(PC_T1 "sets_ext" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "3" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac
        [fst_snd_split_thm,map_o_lemma,lubl_lemma,get_spec⌜Map⌝,get_spec⌜Elems⌝]);
a(REPEAT strip_tac);
a(lemma_tac⌜x ∈ OK_TC_c c⌝THEN1 PC_T1 "sets_ext" asm_prove_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜OK_TC_c⌝]));
a(all_asm_fc_tac[]);
a(asm_rewrite_tac[]);
val AllTuples_lemma2 = save_pop_thm"AllTuples_lemma2";
```

SML

```
push_goal([], ⌜∀c esl tel e1 ml nl e2 •
        e1 ∈ OK_VC_c c ∩ OK_VC_d c ∧
        Elems tel ⊆ OK_TC_d c ∩ OK_TC_c c ∧
        e2 ∈ OK_VC_c c ⇒
        AllTuples c esl tel e1 ml nl e2 ∈ OK_TC_c c⌝);
a(rewrite_tac[get_spec⌜OK_TC_c⌝]);
a(REPEAT strip_tac);
a(lemma_tac⌜Elems tel ⊆ OK_TC_c c ∧ Elems tel ⊆ OK_TC_d c⌝
        THEN1 PC_T1 "sets_ext" asm_prove_tac[]);
a(all_asm_fc_tac[AllTuples_lemma2]);
a(rewrite_tac(map get_spec[⌜Let⌝,⌜AllTuples⌝]));
```

SML

```
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
a(cases_tac⌜c dominates lubl (Fst (Split (Map (λ te• te tl_0) tel)))⌝
        THEN asm_rewrite_tac[]);
a(bc_tac[Group_OK_c_lemma]THEN_TRY asm_rewrite_tac[]);
a(bc_tac[Where_OK_d_lemma]THEN_TRY PC_T1 "sets_ext" asm_rewrite_tac[]);
a(all_fc_tac[AllTuples_lemma1]);
a(all_asm_fc_tac[Join_OK_d_lemma]);
a(POP_ASM_T discard_tac);
a(POP_ASM_T (strip_asm_tac o rewrite_rule(MkDerTable_lemma ::
        map get_spec[⌜HideDerTable⌝,⌜MkDerTable⌝])));
val AllTuples_OK_c_lemma = save_pop_thm"AllTuples_OK_c_lemma";
```

8 CLOSING DOWN

9 THE THEORY fef035

9.1 Parents

fef033

9.2 Children

fef036

9.3 Constants

GroupA *DerTableRow LIST*
→ *Errors*
→ *Errors*
→ *Class* × *DerTableRow LIST LIST*

GroupB *DerTable LIST*
→ *DerTableRow LIST LIST*
→ *VALUE_COMP*
→ *Class* × *DerTableRow LIST LIST*

W *Class*
→ *DerTable LIST*
→ *DerTableRow LIST*
→ *DerTableRow LIST*
→ *VALUE_COMP*
→ *DerTableRow LIST*

H *Class* → *DerTableRow LIST* → *DerTableRow LIST*

9.4 Definitions

GroupA
```
⊢ ∀ rl gbsterling gbclass
  • GroupA rl gbsterling gbclass
      = (let gpby row
            = (ListNth
                 gbsterling
                 (Map Snd (DTR_cols row)),
               ListNth gbclass (Map Fst (DTR_cols row)))
         in let gbc row
               = lubl
                 (ListNth
                    gbsterling
                    (Map Fst (DTR_cols row)))
            in (lubl (Map gbc rl), MakeGroups gpby rl))
```

GroupB
```
⊢ ∀ tl gps having
  • GroupB tl gps having
      = (let has_test gp
            = CommonValue having tl gp Arbitrary
```

```
                    in let wanted_gps
                         = gps ↾ {gp|ItemBool (Snd (has_test gp))}
                       in (lubl (Map (Fst o has_test) gps),
                         wanted_gps))
W                ⊢ ∀ c tl rl1 r rl2 e
                   • W c tl rl1 [] e = []
                     ∧ W c tl rl1 (Cons r rl2) e
                       = (let w = DTR_where r lub Fst (e tl rl1 r)
                         in let h
                              ⇔ (ItemBool (Snd (e tl rl1 r))
                                 ∨ ¬ c dominates w)
                                ∧ c dominates DTR_row r
                           in if h
                           then
                             Cons
                               (MkDerTableRow
                                  w
                                  (DTR_row r)
                                  (DTR_cols r))
                               (W c tl rl1 rl2 e)
                           else W c tl rl1 rl2 e)
H                ⊢ ∀ c rl• H c rl = rl ↾ {r|c dominates DTR_row r}
```

9.5 Theorems

length_map_lemma

```
⊢ ∀ f l• # (Map f l) = # l
```

fun_nth_map_lemma

```
⊢ ∀ l i f
  • i ≤ # l ∧ 1 ≤ i ⇒ f (Nth l i) = Nth (Map f l) i
```

fun_nth_map_lemma1

```
⊢ ∀ i f l1 l2
  • i ≤ # l1 ∧ 1 ≤ i ∧ Map f l1 = Map f l2
    ⇒ f (Nth l1 i) = f (Nth l2 i)
```

ProjectData_lemma

```
⊢ ∀ tl el gp gps
  • ProjectData tl el [] = []
    ∧ ProjectData tl el (Cons gp gps)
      = Map
          (λ r
            • MkDerTableRow
                (DTR_where r)
                (DTR_row r)
                (Map (λ e• e tl gp r) el))
          gp
        @ ProjectData tl el gps
```

DT_spec_HideDerTable_lemma

```
⊢ ∀ c t• DT_spec (HideDerTable c t) = DT_spec t
```

map_HideDerTable_map_DT_spec_lemma

$$\vdash \forall\ c\ ts1\ ts2$$
$$\bullet\ Map\ (HideDerTable\ c)\ ts1 = Map\ (HideDerTable\ c)\ ts2$$
$$\Rightarrow Map\ DT\_spec\ ts1 = Map\ DT\_spec\ ts2$$

HideDerTableData_lemma

$$\vdash \forall\ c\ r\ rs$$
$$\bullet\ HideDerTableData\ c\ [] = []$$
$$\wedge\ HideDerTableData\ c\ (Cons\ r\ rs)$$
$$= (if\ c\ dominates\ DTR\_row\ r$$
$$then$$
$$Cons$$
$$(HideDerTableRow\ c\ r)$$
$$(HideDerTableData\ c\ rs)$$
$$else\ HideDerTableData\ c\ rs)$$

JoinRows_lemma

$$\vdash \forall\ c\ r1\ r2\ rs$$
$$\bullet\ JoinRows\ r1\ [] = []$$
$$\wedge\ JoinRows\ r1\ (Cons\ r2\ rs)$$
$$= Cons$$
$$(MkDerTableRow$$
$$(DTR\_where\ r1\ lub\ DTR\_where\ r2)$$
$$(DTR\_row\ r1\ lub\ DTR\_row\ r2)$$
$$(DTR\_cols\ r1\ @\ DTR\_cols\ r2))$$
$$(JoinRows\ r1\ rs)$$

HideDerTable_flat_lemma

$$\vdash \forall\ c\ blks$$
$$\bullet\ HideDerTableData\ c\ (Flat\ blks)$$
$$= Flat\ (Map\ (HideDerTableData\ c)\ blks)$$

Join_lemma1 $\forall\ c\ ts1\ ts2$

$$\bullet\ Map\ (HideDerTable\ c)\ ts1 = Map\ (HideDerTable\ c)\ ts2$$
$$\Rightarrow HideDerTableData\ c\ (JoinData\ (Map\ DT\_rows\ ts1))$$
$$= HideDerTableData$$
$$c$$
$$(JoinData\ (Map\ DT\_rows\ ts2))$$
$$\vdash \forall\ tl_0\ tl_1\ c$$
$$\bullet\ Map\ (HideDerTable\ c)\ tl_0$$
$$= Map\ (HideDerTable\ c)\ tl_1$$
$$\Rightarrow HideDerTable$$
$$c$$
$$(MkDerTable$$
$$(Fst\ (Join\ tl_0))$$
$$(Snd\ (Join\ tl_0)))$$
$$= HideDerTable$$
$$c$$
$$(MkDerTable$$
$$(Fst\ (Join\ tl_1))$$
$$(Snd\ (Join\ tl_1)))$$

Join_lemma2 $\forall\ c\ ts1\ ts2$

```
• Map (HideDerTable c) ts1 = Map (HideDerTable c) ts2
    ⇒ Map (HideDerTableData c) (Map DT_rows ts1)
      = Map (HideDerTableData c) (Map DT_rows ts2),
  ∀ c ds1 ds2
    • Map (HideDerTableData c) ds1
        = Map (HideDerTableData c) ds2
      ⇒ HideDerTableData c (JoinData ds1)
        = HideDerTableData c (JoinData ds2)
⊢ ∀ c ts1 ts2
  • Map (HideDerTable c) ts1
      = Map (HideDerTable c) ts2
    ⇒ HideDerTableData
        c
        (JoinData (Map DT_rows ts1))
      = HideDerTableData
        c
        (JoinData (Map DT_rows ts2))
```

Join_lemma3

```
⊢ ∀ c ts1 ts2
  • Map (HideDerTable c) ts1 = Map (HideDerTable c) ts2
    ⇒ Map (HideDerTableData c) (Map DT_rows ts1)
      = Map (HideDerTableData c) (Map DT_rows ts2)
```

Join_lemma4

```
∀ c rs1 rs2
• Flat
    (Map
       (λ r• HideDerTableData c (JoinRows r rs2))
       rs1)
  = Flat
    (Map
       (λ r• JoinRows r (HideDerTableData c rs2))
       (HideDerTableData c rs1))
⊢ ∀ c ds1 ds2
  • Map (HideDerTableData c) ds1
      = Map (HideDerTableData c) ds2
    ⇒ HideDerTableData c (JoinData ds1)
      = HideDerTableData c (JoinData ds2)
```

Join_lemma5

```
∀ c r rs
• HideDerTableData c (JoinRows r rs)
  = (if c dominates DTR_row r
     then
       JoinRows
         (HideDerTableRow c r)
         (HideDerTableData c rs)
     else [])
⊢ ∀ c rs1 rs2
  • Flat
      (Map
         (λ r
            • HideDerTableData c (JoinRows r rs2))
```

$rs1)$
$= Flat$
$(Map$
$(\lambda\ r$
$\bullet\ JoinRows\ r\ (HideDerTableData\ c\ rs2))$
$(HideDerTableData\ c\ rs1))$

Join_lemma6 $\vdash \forall\ c\ r\ rs$
$\bullet\ HideDerTableData\ c\ (JoinRows\ r\ rs)$
$= (if\ c\ dominates\ DTR\_row\ r$
$then$
$JoinRows$
$(HideDerTableRow\ c\ r)$
$(HideDerTableData\ c\ rs)$
$else\ [])$

Join_OK_d_lemma
$\vdash \forall\ tl_0\ tl_1\ c$
$\bullet\ Map\ (HideDerTable\ c)\ tl_0$
$= Map\ (HideDerTable\ c)\ tl_1$
$\Rightarrow HideDerTable$
c
$(MkDerTable$
$(Fst\ (Join\ tl_0))$
$(Snd\ (Join\ tl_0)))$
$= HideDerTable$
c
$(MkDerTable$
$(Fst\ (Join\ tl_1))$
$(Snd\ (Join\ tl_1)))$

AllTuples_lemma1
$\vdash \forall\ c\ tl_0\ tl_1\ tel$
$\bullet\ Elems\ tel \subseteq OK\_TC_d\ c$
$\wedge\ c$
$dominates\ lubl$
$(Fst\ (Split\ (Map\ (\lambda\ te\bullet\ te\ tl_0)\ tel)))$
$\wedge\ Map\ (HideDerTable\ c)\ tl_0$
$= Map\ (HideDerTable\ c)\ tl_1$
$\Rightarrow Map$
$(HideDerTable\ c)$
$(Snd\ (Split\ (Map\ (\lambda\ te\bullet\ te\ tl_0)\ tel)))$
$= Map$
$(HideDerTable\ c)$
$(Snd\ (Split\ (Map\ (\lambda\ te\bullet\ te\ tl_1)\ tel)))$

ProjectData_lemma1
$\vdash \forall\ c\ tl_0\ tl_1\ rl_0\ rl_1\ r_0\ r_1\ sl$
$\bullet\ Elems\ sl \subseteq OK\_VC_d\ c \cap OK\_VC_c\ c$
$\wedge\ Map\ (HideDerTable\ c)\ tl_0$
$= Map\ (HideDerTable\ c)\ tl_1$
$\wedge\ Map\ (HideDerTableRow\ c)\ rl_0$

$= Map\ (HideDerTableRow\ c)\ rl_1$
$\wedge\ HideDerTableRow\ c\ r_0 = HideDerTableRow\ c\ r_1$
$\Rightarrow HideDerTableRow$
c
$(MkDerTableRow$
$(DTR\_where\ r_0)$
$(DTR\_row\ r_0)$
$(Map\ (\lambda\ e\bullet\ e\ tl_0\ rl_0\ r_0)\ sl))$
$= HideDerTableRow$
c
$(MkDerTableRow$
$(DTR\_where\ r_1)$
$(DTR\_row\ r_1)$
$(Map\ (\lambda\ e\bullet\ e\ tl_1\ rl_1\ r_1)\ sl))$

HideDerTableData_lemma1
$\vdash \forall\ c\ rl_0\ rl_1$
$\bullet\ Map\ (HideDerTableRow\ c)\ rl_0$
$= Map\ (HideDerTableRow\ c)\ rl_1$
$\Rightarrow HideDerTableData\ c\ rl_0$
$= HideDerTableData\ c\ rl_1$

Group_lemma1 $\vdash \forall\ cc\ tl\ rl\ gbsterling\ gbclass\ having$
$\bullet\ Group\ cc\ tl\ rl\ gbsterling\ gbclass\ having$
$= (let\ (c1,\ gps) = GroupA\ rl\ gbsterling\ gbclass$
$in\ let\ (c2,\ wanted\_gps) = GroupB\ tl\ gps\ having$
$in\ ((if\ cc\ dominates\ c1\ then\ c2\ else\ c1),$
$wanted\_gps))$

Group_lemma2 $\forall\ rl_0\ rl_1\ c\ gbsterling\ gbclass$
$\bullet\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_0)$
$\wedge\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_1)$
$\wedge\ Map\ (HideDerTableRow\ c)\ rl_0$
$= Map\ (HideDerTableRow\ c)\ rl_1$
$\wedge\ \neg\ Map$
$(Map\ (HideDerTableRow\ c))$
$(Snd\ (GroupA\ rl_0\ gbsterling\ gbclass))$
$= Map$
$(Map\ (HideDerTableRow\ c))$
$(Snd\ (GroupA\ rl_1\ gbsterling\ gbclass))$
$\Rightarrow \neg\ c$
$dominates\ Fst$
$(GroupA\ rl_0\ gbsterling\ gbclass),$
$\forall\ tl_0\ tl_1\ gps_0\ gps_1\ c\ having$
$\bullet\ having \in OK\_VC_d\ c \cap OK\_VC_c\ c$
$\wedge\ Map\ (HideDerTable\ c)\ tl_0$
$= Map\ (HideDerTable\ c)\ tl_1$
$\wedge\ Map\ (Map\ (HideDerTableRow\ c))\ gps_0$
$= Map\ (Map\ (HideDerTableRow\ c))\ gps_1$
$\wedge\ \neg\ Map$
$(Map\ (HideDerTableRow\ c))$

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Snd\ (GroupB\ tl_0\ gps_0\ having)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Map\ (HideDerTableRow\ c)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Snd\ (GroupB\ tl_1\ gps_1\ having)) \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \neg\ c \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad dominates\ Fst\ (GroupB\ tl_0\ gps_0\ having) \\
\quad\quad\quad\quad\quad\quad \vdash \forall\ tl_0\ tl_1\ rl_0\ rl_1\ c\ e\ ml\ nl \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_0) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_1) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge \neg\ Map \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Map\ (HideDerTableRow\ c)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Snd\ (Group\ c\ tl_0\ rl_0\ ml\ nl\ e)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Map\ (HideDerTableRow\ c)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (Snd\ (Group\ c\ tl_1\ rl_1\ ml\ nl\ e)) \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \neg\ c\ dominates\ Fst\ (Group\ c\ tl_0\ rl_0\ ml\ nl\ e)
\end{array}
$$

Where_W_lemma

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad \vdash \forall\ c\ tl\ rl\ e \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ Where\ c\ tl\ rl\ e = W\ c\ tl\ (H\ c\ rl)\ (H\ c\ rl)\ e
\end{array}
$$

Where_dominates_lemma

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad \vdash \forall\ tl\ rl\ c\ e \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ c\ dominates\ lubl\ (Map\ DTR\_row\ (Where\ c\ tl\ rl\ e))
\end{array}
$$

Where_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ rl_0\ rl_1\ r_0\ r_1\ c\ e \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ HideDerTableRow\ c\ r_0 = HideDerTableRow\ c\ r_1 \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow ((ItemBool\ (Snd\ (e\ tl_0\ rl_0\ r_0)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \vee \neg\ c \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad dominates\ DTR\_where\ r_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad lub\ Fst\ (e\ tl_0\ rl_0\ r_0)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ c\ dominates\ DTR\_row\ r_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (ItemBool\ (Snd\ (e\ tl_1\ rl_1\ r_1)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \vee \neg\ c \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad dominates\ DTR\_where\ r_1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad lub\ Fst\ (e\ tl_1\ rl_1\ r_1)) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ c\ dominates\ DTR\_row\ r_1)
\end{array}
$$

W_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ rl_0\ rl_1\ rl1\ rl2\ c\ e \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = Map\ (HideDerTable\ c)\ tl_1
\end{array}
$$

$$
\begin{array}{l}
\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\quad\quad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl1 \\
\quad\quad = Map\ (HideDerTableRow\ c)\ rl2 \\
\quad \Rightarrow Map\ (HideDerTableRow\ c)\ (W\ c\ tl_0\ rl1\ rl_0\ e) \\
\quad\quad = Map\ (HideDerTableRow\ c)\ (W\ c\ tl_1\ rl2\ rl_1\ e)
\end{array}
$$

Where_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ rl_0\ rl_1\ c\ e \\
\quad \bullet\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad\quad \wedge\ HideDerTableData\ c\ rl_0 \\
\quad\quad\quad = HideDerTableData\ c\ rl_1 \\
\quad\quad \Rightarrow Map\ (HideDerTableRow\ c)\ (Where\ c\ tl_0\ rl_0\ e) \\
\quad\quad\quad = Map\ (HideDerTableRow\ c)\ (Where\ c\ tl_1\ rl_1\ e)
\end{array}
$$

MakeGroups_lemma1

$$\vdash \forall\ xs\ gpby \bullet \neg\ [] \in Elems\ (MakeGroups\ gpby\ xs)$$

GroupA_cols_lemma1

$$
\begin{array}{l}
\vdash \forall\ r_1\ r_2\ c \\
\quad \bullet\ HideDerTableRow\ c\ r_1 = HideDerTableRow\ c\ r_2 \\
\quad\quad \Rightarrow Map\ Fst\ (DTR\_cols\ r_1) = Map\ Fst\ (DTR\_cols\ r_2)
\end{array}
$$

GroupA_cols_lemma2

$$
\begin{array}{l}
\vdash \forall\ r_1\ r_2\ c\ gbsterling\ gbclass \\
\quad \bullet\ c \\
\quad\quad\quad dominates\ lubl \\
\quad\quad\quad\quad (ListNth \\
\quad\quad\quad\quad\quad gbsterling \\
\quad\quad\quad\quad\quad (Map\ Fst\ (DTR\_cols\ r_1))) \\
\quad\quad \wedge\ HideDerTableRow\ c\ r_1 = HideDerTableRow\ c\ r_2 \\
\quad\quad \Rightarrow ListNth\ gbsterling\ (Map\ Snd\ (DTR\_cols\ r_1)) \\
\quad\quad\quad = ListNth\ gbsterling\ (Map\ Snd\ (DTR\_cols\ r_2))
\end{array}
$$

MakeGroups_lemma2

$$
\begin{array}{l}
\vdash \forall\ rl_0\ rl_1\ g \\
\quad \bullet\ Map\ g\ rl_0 = Map\ g\ rl_1 \\
\quad\quad \Rightarrow Map\ (Map\ g)\ (MakeGroups\ g\ rl_0) \\
\quad\quad\quad = Map\ (Map\ g)\ (MakeGroups\ g\ rl_1)
\end{array}
$$

MakeGroups_lemma3

$$
\begin{array}{l}
\vdash \forall\ rl_0\ rl_1\ g\ h \\
\quad \bullet\ Map\ g\ rl_0 = Map\ g\ rl_1 \wedge Map\ h\ rl_0 = Map\ h\ rl_1 \\
\quad\quad \Rightarrow Map\ (Map\ h)\ (MakeGroups\ g\ rl_0) \\
\quad\quad\quad = Map\ (Map\ h)\ (MakeGroups\ g\ rl_1)
\end{array}
$$

GroupA_lemma

$$
\begin{array}{l}
\vdash \forall\ rl_0\ rl_1\ c\ gbsterling\ gbclass \\
\quad \bullet\ c \\
\quad\quad\quad dominates\ lubl \\
\quad\quad\quad\quad (Map \\
\quad\quad\quad\quad\quad (\lambda\ row \\
\quad\quad\quad\quad\quad\quad \bullet\ lubl \\
\quad\quad\quad\quad\quad\quad\quad (ListNth
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\quad gbsterling \\
\qquad\qquad\qquad\qquad\quad (Map\ Fst\ (DTR\_cols\ row)))) \\
\qquad\qquad\qquad\quad rl_0) \\
\qquad\qquad\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\qquad\qquad\qquad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\qquad\qquad \Rightarrow Map \\
\qquad\qquad\qquad (\lambda\ row \\
\qquad\qquad\qquad\quad \bullet\ (ListNth \\
\qquad\qquad\qquad\qquad\qquad gbsterling \\
\qquad\qquad\qquad\qquad\qquad (Map\ Snd\ (DTR\_cols\ row)), \\
\qquad\qquad\qquad\qquad ListNth \\
\qquad\qquad\qquad\qquad\qquad gbclass \\
\qquad\qquad\qquad\qquad\qquad (Map\ Fst\ (DTR\_cols\ row)))) \\
\qquad\qquad\qquad rl_0 \\
\qquad\qquad\quad = Map \\
\qquad\qquad\qquad (\lambda\ row \\
\qquad\qquad\qquad\quad \bullet\ (ListNth \\
\qquad\qquad\qquad\qquad\qquad gbsterling \\
\qquad\qquad\qquad\qquad\qquad (Map\ Snd\ (DTR\_cols\ row)), \\
\qquad\qquad\qquad\qquad ListNth \\
\qquad\qquad\qquad\qquad\qquad gbclass \\
\qquad\qquad\qquad\qquad\qquad (Map\ Fst\ (DTR\_cols\ row)))) \\
\qquad\qquad\qquad rl_1
\end{array}
$$

GroupA_OK$_d$_lemma

$$
\begin{array}{l}
\qquad\qquad \vdash \forall\ rl_0\ rl_1\ c\ gbsterling\ gbclass \\
\qquad\qquad\quad \bullet\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_0) \\
\qquad\qquad\qquad\quad \wedge\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_1) \\
\qquad\qquad\qquad\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\qquad\qquad\qquad\qquad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\qquad\qquad\qquad\quad \wedge\ \neg\ Map \\
\qquad\qquad\qquad\qquad\quad (Map\ (HideDerTableRow\ c)) \\
\qquad\qquad\qquad\qquad\quad (Snd\ (GroupA\ rl_0\ gbsterling\ gbclass)) \\
\qquad\qquad\qquad\qquad\ = Map \\
\qquad\qquad\qquad\qquad\quad (Map\ (HideDerTableRow\ c)) \\
\qquad\qquad\qquad\qquad\quad (Snd\ (GroupA\ rl_1\ gbsterling\ gbclass)) \\
\qquad\qquad\quad \Rightarrow \neg\ c \\
\qquad\qquad\qquad dominates\ Fst \\
\qquad\qquad\qquad\quad (GroupA\ rl_0\ gbsterling\ gbclass)
\end{array}
$$

GroupB_OK$_d$_lemma

$$
\begin{array}{l}
\qquad\qquad \vdash \forall\ tl_0\ tl_1\ gps_0\ gps_1\ c\ having \\
\qquad\qquad\quad \bullet\ having \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\qquad\qquad\qquad\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\qquad\qquad\qquad\qquad = Map\ (HideDerTable\ c)\ tl_1 \\
\qquad\qquad\qquad\quad \wedge\ Map\ (Map\ (HideDerTableRow\ c))\ gps_0 \\
\qquad\qquad\qquad\qquad = Map\ (Map\ (HideDerTableRow\ c))\ gps_1 \\
\qquad\qquad\qquad\quad \wedge\ \neg\ Map \\
\qquad\qquad\qquad\qquad\quad (Map\ (HideDerTableRow\ c)) \\
\qquad\qquad\qquad\qquad\quad (Snd\ (GroupB\ tl_0\ gps_0\ having))
\end{array}
$$

$$
\begin{array}{l}
\quad = Map \\
\quad\quad (Map\ (HideDerTableRow\ c)) \\
\quad\quad (Snd\ (GroupB\ tl_1\ gps_1\ having)) \\
\Rightarrow \neg\ c\ dominates\ Fst\ (GroupB\ tl_0\ gps_0\ having)
\end{array}
$$

Group_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ rl_0\ rl_1\ c\ e\ ml\ nl \\
\bullet\ e \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad \wedge\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_0) \\
\quad \wedge\ c\ dominates\ lubl\ (Map\ DTR\_row\ rl_1) \\
\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\quad\quad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\quad \wedge \neg\ Map \\
\quad\quad\quad (Map\ (HideDerTableRow\ c)) \\
\quad\quad\quad (Snd\ (Group\ c\ tl_0\ rl_0\ ml\ nl\ e)) \\
\quad\quad = Map \\
\quad\quad\quad (Map\ (HideDerTableRow\ c)) \\
\quad\quad\quad (Snd\ (Group\ c\ tl_1\ rl_1\ ml\ nl\ e)) \\
\Rightarrow \neg\ c\ dominates\ Fst\ (Group\ c\ tl_0\ rl_0\ ml\ nl\ e)
\end{array}
$$

GroupA_OK$_c$_lemma

$$
\begin{array}{l}
\vdash \forall\ rl_0\ rl_1\ c\ gbsterling\ gbclass \\
\bullet\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\quad\quad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\quad\Rightarrow Fst\ (GroupA\ rl_0\ gbsterling\ gbclass) \\
\quad\quad = Fst\ (GroupA\ rl_1\ gbsterling\ gbclass)
\end{array}
$$

GroupB_OK$_c$_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ gps_0\ gps_1\ c\ having \\
\bullet\ having \in OK\_VC_c\ c \\
\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad \wedge\ Map\ (Map\ (HideDerTableRow\ c))\ gps_0 \\
\quad\quad = Map\ (Map\ (HideDerTableRow\ c))\ gps_1 \\
\quad\Rightarrow Fst\ (GroupB\ tl_0\ gps_0\ having) \\
\quad\quad = Fst\ (GroupB\ tl_1\ gps_1\ having)
\end{array}
$$

Group_OK$_c$_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ rl_0\ rl_1\ c\ e\ ml\ nl \\
\bullet\ e \in OK\_VC_c\ c \\
\quad \wedge\ Map\ (HideDerTableRow\ c)\ rl_0 \\
\quad\quad = Map\ (HideDerTableRow\ c)\ rl_1 \\
\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad\Rightarrow Fst\ (Group\ c\ tl_0\ rl_0\ ml\ nl\ e) \\
\quad\quad = Fst\ (Group\ c\ tl_1\ rl_1\ ml\ nl\ e)
\end{array}
$$

ProjectData_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ tl_0\ tl_1\ gps_0\ gps_1\ c\ sl \\
\bullet\ Elems\ sl \subseteq OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0
\end{array}
$$

$$
\begin{array}{l}
\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\wedge\ Map\ (Map\ (HideDerTableRow\ c))\ gps_0 \\
\quad = Map\ (Map\ (HideDerTableRow\ c))\ gps_1 \\
\Rightarrow Map \\
\quad (HideDerTableRow\ c) \\
\quad (ProjectData\ tl_0\ sl\ gps_0) \\
\quad = Map \\
\quad (HideDerTableRow\ c) \\
\quad (ProjectData\ tl_1\ sl\ gps_1)
\end{array}
$$

TableContents_OK$_d$_lemma

$$\vdash \forall\ c\ i\bullet\ TableContents\ i \in OK\_TC_d\ c$$

AllTuples_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ esl\ tel\ e1\ ml\ nl\ e2 \\
\bullet\ Elems\ (Map\ Fst\ esl) \subseteq OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad \wedge\ Elems\ tel \subseteq OK\_TC_d\ c \\
\quad \wedge\ e1 \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad \wedge\ e2 \in OK\_VC_d\ c \cap OK\_VC_c\ c \\
\quad \Rightarrow AllTuples\ c\ esl\ tel\ e1\ ml\ nl\ e2 \in OK\_TC_d\ c
\end{array}
$$

TableContents_OK$_c$_lemma

$$\vdash \forall\ c\ i\bullet\ TableContents\ i \in OK\_TC_c\ c$$

AllTuples_lemma2

$$
\begin{array}{l}
\vdash \forall\ c\ tl_0\ tl_1\ tel \\
\bullet\ Elems\ tel \subseteq OK\_TC_c\ c \\
\quad \wedge\ Map\ (HideDerTable\ c)\ tl_0 \\
\quad\quad = Map\ (HideDerTable\ c)\ tl_1 \\
\quad \Rightarrow lubl\ (Fst\ (Split\ (Map\ (\lambda\ te\bullet\ te\ tl_0)\ tel))) \\
\quad\quad = lubl\ (Fst\ (Split\ (Map\ (\lambda\ te\bullet\ te\ tl_1)\ tel)))
\end{array}
$$

AllTuples_OK$_c$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ esl\ tel\ e1\ ml\ nl\ e2 \\
\bullet\ e1 \in OK\_VC_c\ c \cap OK\_VC_d\ c \\
\quad \wedge\ Elems\ tel \subseteq OK\_TC_d\ c \cap OK\_TC_c\ c \\
\quad \wedge\ e2 \in OK\_VC_c\ c \\
\quad \Rightarrow AllTuples\ c\ esl\ tel\ e1\ ml\ nl\ e2 \in OK\_TC_c\ c
\end{array}
$$

10 INDEX