Project: DRA FRONT END FILTER PROJECT

Title: Value Computation Security Proofs

Ref: DS/FMU/FEF/033 *Issue: Revision* : *1.16* *Date:* 5 June 2016

Status: Draft *Type:* Specification

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
R. D. Arthan	WIN01		

Authorisation for Issue:

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

Abstract: This document contains the formal proofs relating to the value computations of 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.

[3] DS/FMU/FEF/031. *Execution Model Security Proofs.* R.D. Arthan, ICL Secure Systems, WIN01.

[4] DS/FMU/FEF/032. *Table Computations for SWORD.* R.D. Arthan, ICL Secure Systems, WIN01.

0.3 Changes History

Issue *Revision* : *1.16* **(5 June 2016)** Unused lemmas removed; lemmas renamed.

Issue 1.17 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 *fef033* to hold the theorems to be proved and set up a proof context in which to carry out the proofs. Some theorems are required from the theory *fef031* defined in [3] and so this is made a parent of the theory *fef033*.

SML
```
open_theory "fef032";
(force_delete_theory "fef033" handle _ => ());
new_theory"fef033";
new_parent"fef031";
set_pc"hol";
```

3 MISCELLANY

3.1 Consistency Proofs

SML
```
push_consistency_goal⌜TableComputations⌝;
a(prove_∃_tac);
a(strip_tac THEN rewrite_tac[tac_proof(([], ⌜∀x•∃z y• (y, z) = x⌝),
                strip_tac THEN ∃_tac⌜Snd x⌝ THEN ∃_tac⌜Fst x⌝ THEN rewrite_tac[])]);
save_consistency_thm⌜TableComputations⌝ (pop_thm());
```

4 LEMMAS

SML

```
set_pc"hol";
set_goal([], ⌜∀i1 i2• BoolItem i1 = BoolItem i2 ⇔ i1 = i2⌝);
a(rewrite_tac(map get_spec[⌜BoolItem⌝, ⌜ValuedItemItem⌝])
        THEN REPEAT ∀_tac THEN ⇔_tac THEN_TRY asm_rewrite_tac[]);
a(LEMMA_T ⌜VI_val(MkValuedItem sterling (BoolVal i1))
            = VI_val(MkValuedItem sterling (BoolVal i2))⌝
        (accept_tac o rewrite_rule(map get_spec[⌜MkValuedItem⌝, ⌜BoolVal⌝]))
        THEN1 asm_rewrite_tac[]);
val BoolItem_OneOne_lemma = save_pop_thm"BoolItem_OneOne_lemma";
```

SML

```
set_goal([], ⌜∀c r1 r2•
        HideDerTableRow c r1 = HideDerTableRow c r2 ⇒
        Length (DTR_cols r1) = Length (DTR_cols r2)⌝);
a(rewrite_tac(MkDerTableRow_lemma :: map get_spec[⌜HideDerTableRow⌝, ⌜Let⌝]));
a(REPEAT strip_tac);
a(LEMMA_T⌜Length(Map(λ (c', i) • if c dominates c'
                    then (c', i)
                    else (c', ValuedItemItem (MkValuedItem sterling dummyVal)))
              (DTR_cols r1))
            = Length(Map (λ (c', i) • if c dominates c'
                    then (c', i)
                    else (c', ValuedItemItem (MkValuedItem sterling dummyVal)))
              (DTR_cols r2))⌝
        (strip_asm_tac o once_rewrite_rule[length_map_thm])
        THEN1 asm_rewrite_tac[]);
val HideDerTableRow_Length_lemma = save_pop_thm"HideDerTableRow_Length_lemma";
```

SML

```
set_goal([], ⌜∀c r i•
        1 ≤ i ∧ i ≤ Length (DTR_cols r) ⇒
        Nth (DTR_cols (HideDerTableRow c r)) i =
        (Fst (Nth (DTR_cols r) i),
        if      c dominates (Fst (Nth (DTR_cols r) i))
        then    Snd(Nth (DTR_cols r) i)
        else    ValuedItemItem (MkValuedItem sterling dummyVal))⌝);
a(REPEAT ∀_tac);
a(rewrite_tac(map get_spec[⌜Let⌝, ⌜DTR_cols⌝, ⌜HideDerTableRow⌝]));
a(LEMMA_T ⌜∃cols• DTR_cols r = cols⌝
        (REPEAT_TTCL STRIP_THM_THEN rewrite_thm_tac)
        THEN1 prove_∃_tac);
a(intro_∀_tac(⌜i⌝, ⌜i⌝) THEN list_induction_tac⌜cols⌝);
(* *** Goal "1" *** *)
a(rewrite_tac(map get_spec[⌜Length⌝]));
a(REPEAT strip_tac THEN all_var_elim_asm_tac);
```

SML

```
(* *** Goal "2" *** *)
a(rewrite_tac(map get_spec[⌜Length⌝, ⌜Nth⌝]));
a(REPEAT ∀_tac);
a(cases_tac ⌜i = 1⌝ THEN asm_rewrite_tac(map get_spec[⌜Map⌝, ⌜Nth⌝])
        THEN REPEAT strip_tac);
(* *** Goal "2.1" *** *)
a(CASES_T ⌜c dominates Fst x⌝ rewrite_thm_tac);
(* *** Goal "2.2" *** *)
a(GET_ASM_T ⌜1 ≤ i⌝ (strip_asm_tac o rewrite_rule[get_spec⌜$≤⌝]));
a(LEMMA_T⌜i = i' + 1⌝ rewrite_thm_tac THEN1
        (POP_ASM_T (rewrite_thm_tac o eq_sym_rule)));
a(lemma_tac⌜1 ≤ i' ∧ i' ≤ Length cols⌝ THEN1 PC_T1 "lin_arith" asm_prove_tac[]);
a(all_asm_fc_tac[]);
val Nth_HideDerTableRow_lemma = save_pop_thm"Nth_HideDerTableRow_lemma";
```

SML

```
push_goal([], ⌜∀c• DTR_row o HideDerTableRow c = DTR_row⌝);
a(PC_T1"hol2" REPEAT strip_tac);
a(rewrite_tac(
        map get_spec[⌜$o:((′a→′c)→((′b→′a)→(′b→′c)))⌝,
        ⌜HideDerTableRow⌝, ⌜DTR_row⌝, ⌜Let⌝]));
val DTR_row_o_HideDerTableRow_lemma =
        save_pop_thm"DTR_row_o_HideDerTableRow_lemma";
```

SML
```
set_goal([], ⌜∀f a b c• f(if a then b else c) = (if a then f b else f c)⌝);
a(REPEAT strip_tac THEN cases_tac⌜a:BOOL⌝ THEN asm_rewrite_tac[]);
val fun_if_thm = save_pop_thm"fun_if_thm";
```

SML
```
push_goal([], ⌜∀c rl•
        rl ↾ {r|c dominates DTR_row r ∧ c dominates DTR_where r} = []
        ⇔
        (Map (HideDerTableRow c) (rl ↾ {r|c dominates DTR_row r}))
                ↾ {r|c dominates DTR_where r} = []⌝);
a(REPEAT ∀_tac);
a(list_induction_tac ⌜rl⌝ THEN asm_rewrite_tac(
        map get_spec[⌜Let⌝, ⌜Map⌝, ⌜DTR_row⌝, ⌜HideDerTableRow⌝, ⌜$↾⌝]));
(* *** Goal "1" *** *)
a(strip_tac THEN cases_tac ⌜c dominates DTR_row x⌝ THEN asm_rewrite_tac(
        map get_spec[⌜Let⌝, ⌜Map⌝, ⌜DTR_row⌝, ⌜HideDerTableRow⌝, ⌜$↾⌝]));
a(cases_tac⌜c dominates DTR_where x⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(strip_tac THEN cases_tac ⌜c dominates DTR_row x⌝ THEN asm_rewrite_tac(
        map get_spec[⌜Let⌝, ⌜Map⌝, ⌜DTR_row⌝, ⌜HideDerTableRow⌝, ⌜$↾⌝]));
a(cases_tac⌜c dominates DTR_where x⌝ THEN asm_rewrite_tac[]);
val ↾_null_map_hide_lemma = save_pop_thm"↾_null_map_hide_lemma";
```

SML
```
push_goal([], ⌜∀c rl1 rl2•
        Map (HideDerTableRow c) rl1 = Map(HideDerTableRow c) rl2
        ⇒
        Map (HideDerTableRow c) (rl1 ↾ {r | c dominates DTR_where r})
        =
        Map (HideDerTableRow c) (rl2 ↾ {r | c dominates DTR_where r})⌝);
a(REPEAT_N 2 strip_tac);
a(list_induction_tac ⌜rl1⌝ THEN asm_rewrite_tac(
        map_null_thm::map get_spec[⌜Map⌝, ⌜$↾⌝]) THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜$↾⌝]);
```

SML

```
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN strip_asm_tac(∀_elim⌜rl2⌝list_cases_thm)
        THEN all_var_elim_asm_tac1 THEN asm_rewrite_tac[get_spec⌜Map⌝]);
a(rewrite_tac(MkDerTableRow_lemma::
        map get_spec[⌜HideDerTableRow⌝, ⌜Let⌝, ⌜Map⌝, ⌜$↾⌝])
        THEN REPEAT strip_tac);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[]);
a(cases_tac⌜c dominates DTR_where x'⌝ THEN asm_rewrite_tac[get_spec⌜Map⌝]);
a(rewrite_tac(MkDerTableRow_lemma::
        map get_spec[⌜HideDerTableRow⌝, ⌜Let⌝, ⌜Map⌝, ⌜$↾⌝])
        THEN REPEAT strip_tac);
val map_hide_map_hide_↾_lemma = save_pop_thm"map_hide_map_hide_↾_lemma";
```

SML

```
set_goal([], ⌜∀l a b• l ↾ (a ∩ b) = (l ↾ a) ↾ b⌝);
a(REPEAT strip_tac);
a(list_induction_tac ⌜l⌝ THEN asm_rewrite_tac(
        map get_spec[⌜$↾⌝]));
a(strip_tac THEN PC_T1 "sets_ext1" rewrite_tac[]);
a(cases_tac⌜x ∈ a⌝ THEN asm_rewrite_tac[get_spec⌜$↾⌝]);
val ↾_∩_lemma = save_pop_thm"↾_∩_lemma";
```

SML

```
push_goal([], ⌜∀xy list•
        Split [] = ([], []) ∧
        Split (Cons xy list) =
        (Cons (Fst xy) (Fst (Split list)), Cons (Snd xy) (Snd (Split list)))⌝);
a(rewrite_tac[get_spec⌜Split⌝] THEN REPEAT strip_tac);
a(lemma_tac ⌜∃x y•xy = (x, y)⌝ THEN1
        (∃_tac⌜Fst xy⌝ THEN ∃_tac⌜Snd xy⌝) THEN
        asm_rewrite_tac[get_spec⌜Split⌝]);
val split_thm = save_pop_thm"split_thm";
```

SML

```
push_goal([], ⌜∀list•Length list = 0 ⇔ list = []⌝);
a(REPEAT strip_tac THEN_LIST [id_tac, asm_rewrite_tac[get_spec⌜Length⌝]]);
a(POP_ASM_T ante_tac THEN list_induction_tac⌜list⌝
        THEN rename_tac[] THEN asm_rewrite_tac[get_spec⌜Length⌝]);
val length_0_thm = save_pop_thm"length_0_thm";
```

SML

```
push_goal([], ⌜∀list• Length list = 1 ⇔ ∃x• list = [x]⌝);
a(REPEAT strip_tac THEN_LIST [id_tac, asm_rewrite_tac[get_spec⌜Length⌝]]);
a(POP_ASM_T ante_tac THEN list_induction_tac⌜list⌝
        THEN rename_tac[] THEN asm_rewrite_tac[get_spec⌜Length⌝, length_0_thm]);
a(REPEAT strip_tac THEN prove_∃_tac);
val length_1_thm = save_pop_thm"length_1_thm";
```

SML

```
push_goal([], ⌜∀list•
        Fst(Split list) = Map Fst list ∧
        Snd(Split list)= Map Snd list ⌝);
a(strip_tac);
a(list_induction_tac ⌜list⌝ THEN
        asm_rewrite_tac(split_thm::map get_spec[⌜Map⌝]));
val fst_snd_split_thm = save_pop_thm"fst_snd_split_thm";
```

SML

```
push_goal([], ⌜∀c cl• lubl [] = lattice_bottom ∧ lubl (Cons c cl) = c lub lubl cl⌝);
a(REPEAT ∀_tac THEN rewrite_tac(map get_spec[⌜lubl⌝, ⌜Fold⌝]));
val lubl_lemma = save_pop_thm"lubl_lemma";
```

SML

```
push_goal([], ⌜∀c• c lub lattice_bottom = c⌝);
a(REPEAT strip_tac THEN lemma_tac
        ⌜c dominates c lub lattice_bottom ∧
                c lub lattice_bottom dominates c⌝
        THEN_LIST [rewrite_tac[get_spec⌜$lub⌝], all_fc_tac[get_spec⌜$lub⌝]]);
a(lemma_tac
        ⌜c dominates c ∧ c dominates lattice_bottom⌝
        THEN_LIST [rewrite_tac[get_spec⌜$lub⌝], all_fc_tac[get_spec⌜$lub⌝]]);
val lub_lattice_bottom_thm = save_pop_thm"lub_lattice_bottom_thm";
```

SML

```
push_goal([], ⌜∀cc cil•
        cc dominates lubl (Map Fst cil)
        ⇒
        Map (λ (c, i)• (c, (if cc dominates c then i else Arbitrary))) cil
        = cil
⌝);
a(REPEAT ∀_tac);
a(list_induction_tac ⌜cil⌝
        THEN asm_rewrite_tac(dominates_lub_lemma::lubl_lemma :: map get_spec[⌜Map⌝]));
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
val dominates_lubl_map_hide_lemma = save_pop_thm "dominates_lubl_map_hide_lemma";
```

The following primitive recursive reformulation of *CaseVal* could perhaps do the case analysis more economically. Nonetheless it serves its purpose.

SML

```
set_goal([], ⌜∀cc tst cev cevs ev•
        CaseVal cc tst [] ev =
        (λtl rl r•
        let     (tc, ti) = tst tl rl r
        in let  (ec, ei) = ev tl rl r
        in      if      cc dominates tc
                then    (ec, ei)
                else    (tc, ei))
∧       CaseVal cc tst (Cons cev cevs) ev =
        (λtl rl r•
        let     (cr, ir) = CaseVal cc tst cevs ev tl rl r
        in let  (tc, ti) = tst tl rl r
        in let  (ce, cv) = cev
        in let  (cec, cei) = ce tl rl r
        in let  (cvc, cvi) = cv tl rl r
        in      if      ti = cei
                then    if      cc dominates tc ∧ cc dominates cec
                        then    (cvc, cvi)
                        else if ¬cc dominates tc
                        then    (tc, cvi)
                        else    (cec, cvi)
                else    if      cc dominates tc ∧ cc dominates cec
                        then    (cr, ir)
                        else if ¬cc dominates tc
                        then    (tc, ir)
                        else    (cec, ir))
⌝);
```

SML

```
a(PC_T1 "predicates1" rewrite_tac[]);
a(rewrite_tac(map get_spec[⌜$dom⌝, ⌜CaseVal⌝, ⌜CheckTest⌝,
        ⌜CaseValValue⌝, ⌜CheckList⌝, ⌜Let⌝, ⌜Map⌝]) THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(cases_tac ⌜cc dominates Fst (tst x x' x'')⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(cases_tac ⌜cc dominates Fst (tst x x' x'')⌝ THEN
        cases_tac ⌜cc dominates Fst (Fst cev x x' x'')⌝ THEN
        cases_tac ⌜Snd (tst x x' x'') = Snd (Fst cev x x' x'')⌝ THEN
        asm_rewrite_tac[]);
val CaseVal_lemma = save_pop_thm "CaseVal_lemma";
```

5 PROOF THAT OK_TC_d GIVES *OkTableComputation*

SML

```
set_pc"hol";
set_goal([], ⌜∀c• OK_TC_d c ⊆ OkTableComputation c⌝);
a(PC_T1"hol2"
        rewrite_tac(map get_spec[⌜OK_TC_d⌝, ⌜OkTableComputation⌝, ⌜RiskInputs⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 2 (strip_asm_tac o eq_sym_rule));
a(DROP_NTH_ASM_T 2 (strip_asm_tac o conv_rule(RAND_C eq_sym_conv)));
a(all_asm_fc_tac[]);
val OK_TC_d_lemma = save_pop_thm"OK_TC_d_lemma";
```

6 DATA OKNESS PROOFS

6.1 Constant Expression

SML

```
set_goal([], ⌜∀c ci• DenoteConstant ci ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_d⌝, ⌜DenoteConstant⌝]) THEN REPEAT strip_tac);
val DenoteConstant_OK_d_lemma = save_pop_thm"DenoteConstant_OK_d_lemma";
```

6.2 *Contents*

SML

```
set_goal([], ⌜∀c i• Contents i ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_d⌝, ⌜Contents⌝])
        THEN REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN all_fc_tac[HideDerTableRow_Length_lemma]);
a(cases_tac ⌜1 ≤ i ∧ i ≤ # (DTR_cols r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜i ≤ # (DTR_cols r_0)⌝ THEN1 asm_rewrite_tac[]);
a(ALL_FC_T (MAP_EVERY(asm_tac o ∀_elim⌜c⌝)) [Nth_HideDerTableRow_lemma]);
a(contr_tac);
a(swap_nth_asm_concl_tac 4 THEN LIST_DROP_NTH_ASM_T [3, 8] rewrite_tac);
a(cases_tac ⌜Fst (Nth (DTR_cols r_1) i) = Fst (Nth (DTR_cols r_0) i)⌝
        THEN asm_rewrite_tac[]);
a(conv_tac (RAND_C eq_sym_conv) THEN asm_rewrite_tac[]);
val Contents_OK_d_lemma = save_pop_thm"Contents_OK_d_lemma";
```

6.3 *Classification*

SML

```
set_goal([], ⌜∀c i• Classification i ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_d⌝, ⌜Classification⌝])
        THEN REPEAT strip_tac);
a(POP_ASM_T ante_tac THEN all_fc_tac[HideDerTableRow_Length_lemma]);
a(cases_tac ⌜1 ≤ i ∧ i ≤ # (DTR_cols r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜i ≤ # (DTR_cols r_0)⌝ THEN1 asm_rewrite_tac[]);
a(ALL_FC_T (MAP_EVERY(asm_tac o ∀_elim⌜c⌝)) [Nth_HideDerTableRow_lemma]);
a(contr_tac);
a(swap_nth_asm_concl_tac 4 THEN LIST_DROP_NTH_ASM_T [3, 8] rewrite_tac);
a(cases_tac ⌜Fst (Nth (DTR_cols r_1) i) = Fst (Nth (DTR_cols r_0) i)⌝
        THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
val Classification_OK_d_lemma = save_pop_thm"Classification_OK_d_lemma";
```

6.4 *Countall*

SML

```
set_goal([], ⌜∀c• CountAll ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_d⌝, ⌜CountAll⌝,⌜Let⌝]) THEN REPEAT strip_tac);
a(lemma_tac⌜¬(# rl_0) = (# rl_1)⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(DROP_NTH_ASM_T 2 (fn x => id_tac));
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN POP_ASM_T ante_tac);
a(intro_∀_tac(⌜r_1⌝,⌜r_1⌝));
a(intro_∀_tac(⌜r_0⌝,⌜r_0⌝));
a(intro_∀_tac(⌜rl_1⌝,⌜rl_1⌝));
a(list_induction_tac⌜rl_0⌝ THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(strip_asm_tac (∀_elim⌜rl_1⌝ list_cases_thm));
(* *** Goal "1.1" *** *)
a(DROP_NTH_ASM_T 2 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "1.2" *** *)
a(swap_asm_concl_tac
        ⌜Map (HideDerTableRow c) [] = Map (HideDerTableRow c) rl_1⌝
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
```

SML

```
(* *** Goal "2" *** *)
a(strip_asm_tac (∀_elim⌜rl1⌝ list_cases_thm));
(* *** Goal "2.1" *** *)
a(swap_asm_concl_tac
        ⌜Map (HideDerTableRow c) (Cons x rl0) = Map (HideDerTableRow c) rl1⌝
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(rewrite_tac[dominates_lub_lemma,lubl_lemma,get_spec⌜Map⌝]);
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 4 (strip_asm_tac o rewrite_rule[get_spec⌜Map⌝]));
a(DROP_NTH_ASM_T 6 (ante_tac o list_∀_elim[⌜list2⌝,⌜x⌝,⌜x'⌝]));
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 5 (strip_asm_tac o rewrite_rule[get_spec⌜Length⌝]));
val CountAll_OKd_lemma = save_pop_thm"CountAll_OKd_lemma";
```

6.5 Monadic

SML

```
set_goal([], ⌜∀c f vc• vc ∈ OK_VCd c ⇒ MonOp f vc ∈ OK_VCd c⌝);
a(rewrite_tac(map get_spec[⌜OK_VCd⌝, ⌜MonOp⌝, ⌜Let⌝]) THEN REPEAT strip_tac);
a(lemma_tac⌜¬Snd (vc tl0 rl0 r0) = Snd (vc tl1 rl1 r1)⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(all_asm_fc_tac[]);
val MonOp_OKd_lemma = save_pop_thm"MonOp_OKd_lemma";
```

6.6 Binary

SML

```
set_goal([],
        ⌜∀cc cil1 cil2•
        Map (λ(c, i)• (c, if cc dominates c then i else Arbitrary)) cil1 =
        Map (λ(c, i)• (c, if cc dominates c then i else Arbitrary)) cil2
        ⇒       Fst(ComputeAnd cc cil1) = Fst(ComputeAnd cc cil2)
        ∧       (cc dominates Fst(ComputeAnd cc cil1)
                ⇒       Snd(ComputeAnd cc cil1) = Snd(ComputeAnd cc cil2))
⌝);
a(rewrite_tac(∀_elim⌜Fst⌝ fun_if_thm :: ∀_elim⌜Snd⌝ fun_if_thm ::
        map get_spec[⌜ComputeAnd⌝, ⌜Map⌝, ⌜Let⌝])
        THEN REPEAT ∀_tac THEN strip_tac);
a(lemma_tac⌜cil1 ↾ {(c, i)|cc dominates c ∧ ¬ ItemBool i}
                = cil2 ↾ {(c, i)|cc dominates c ∧ ¬ ItemBool i}⌝);
```

SML

```
(* *** Goal "1" *** *)
a(all_asm_ante_tac THEN intro_∀_tac(⌜cil2⌝, ⌜cil2⌝)
        THEN list_induction_tac⌜cil1⌝
        THEN asm_rewrite_tac(map get_spec[⌜$|⌝, ⌜Map⌝]));
(* *** Goal "1.1" *** *)
a(rewrite_tac[map_null_thm]);
a(REPEAT strip_tac THEN asm_rewrite_tac[get_spec⌜$|⌝]);
(* *** Goal "1.2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜cil2⌝list_cases_thm) THEN all_var_elim_asm_tac1
        THEN POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(asm_rewrite_tac[get_spec⌜$|⌝]);
a(cases_tac⌜cc dominates Fst x'⌝ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
a(cases_tac⌜¬ ItemBool (Snd x')⌝ THEN asm_rewrite_tac[]);
a(PC_T1 "prop_eq_pair" asm_prove_tac[]);
```

SML

```
(* *** Goal "2" *** *)
a(asm_rewrite_tac[]);
a(lemma_tac⌜Map Fst cil1 = Map Fst cil2⌝);
(* *** Goal "2.1" *** *)
a(LEMMA_T⌜
        Map Fst(Map (λ (c, i)•
                (c, (if cc dominates c then i else Arbitrary))) cil1) =
        Map Fst(Map (λ (c, i)•
                (c, (if cc dominates c then i else Arbitrary))) cil2)⌝
        (ante_tac o rewrite_rule[map_o_lemma])
        THEN1 asm_rewrite_tac[]);
a(LEMMA_T ⌜
        (Fst o (λ (c, i:Item) • (c, (if cc dominates c then i else Arbitrary))))
        = Fst⌝ rewrite_thm_tac);
a(PC_T1"hol2" rewrite_tac[]);
```

SML

```
(* *** Goal "2.2" *** *)
a(cases_tac⌜cil2 ↑ {(c, i)|cc dominates c ∧ ¬ItemBool i} = []⌝ THEN
        asm_rewrite_tac[]);
a(strip_tac THEN LEMMA_T ⌜cil1 = cil2⌝ rewrite_thm_tac);
a(lemma_tac⌜cc dominates lubl (Map Fst cil1)⌝ THEN1 asm_rewrite_tac[]);
a(all_fc_tac[dominates_lubl_map_hide_lemma]);
a(swap_nth_asm_concl_tac 8 THEN asm_rewrite_tac[]);
val ComputeAnd_lemma = save_pop_thm"ComputeAnd_lemma";
```

SML

```
set_goal([],
        ⌜∀c vcl• Elems vcl ⊆ OK_VC_d c ∧ Elems vcl ⊆ OK_VC_c c
        ⇒ BinOpAnd c vcl ∈ OK_VC_d c⌝);
a(rewrite_tac(
        map get_spec[⌜BinOpAnd⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac THEN swap_nth_asm_concl_tac 1
        THEN POP_ASM_T ante_tac THEN strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_0 rl_0 r_0) vcl) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_1 rl_1 r_1) vcl)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeAnd_lemma]]);
a(POP_ASM_T discard_tac THEN
        asm_ante_tac ⌜Elems vcl ⊆ OK_VC_c c⌝ THEN
        asm_ante_tac ⌜Elems vcl ⊆ OK_VC_d c⌝
        THEN list_induction_tac⌜vcl⌝);
```

SML

```
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "3" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "4" *** *)
a(asm_rewrite_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]) THEN REPEAT_N 3 strip_tac);
a(LEMMA_T⌜x ∈ OK_VC_d c ∧ x ∈ OK_VC_c c⌝
        (strip_asm_tac o rewrite_rule(map get_spec[⌜OK_VC_d⌝, ⌜OK_VC_c⌝])) THEN1
        PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[]);
a(cases_tac⌜c dominates Fst (x tl_1 rl_1 r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (x tl_0 rl_0 r_0)⌝ THEN1 asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val BinOpAnd_OK_d_lemma = save_pop_thm"BinOpAnd_OK_d_lemma";
```

SML

```
set_goal([],
        ⌜∀cc cil1 cil2•
        Map (λ(c, i)• (c, if cc dominates c then i else Arbitrary)) cil1 =
        Map (λ(c, i)• (c, if cc dominates c then i else Arbitrary)) cil2
        ⇒       Fst(ComputeOr cc cil1) = Fst(ComputeOr cc cil2)
        ∧       (cc dominates Fst(ComputeOr cc cil1)
                ⇒       Snd(ComputeOr cc cil1) = Snd(ComputeOr cc cil2))
⌝);
a(rewrite_tac(∀_elim⌜Fst⌝ fun_if_thm :: ∀_elim⌜Snd⌝ fun_if_thm ::
        map get_spec[⌜ComputeOr⌝, ⌜Map⌝, ⌜Let⌝])
        THEN REPEAT ∀_tac THEN strip_tac);
a(lemma_tac⌜cil1 ↾ {(c, i)|cc dominates c ∧ ItemBool i}
                = cil2 ↾ {(c, i)|cc dominates c ∧ ItemBool i}⌝);
```

SML

```
(* *** Goal "1" *** *)
a(all_asm_ante_tac THEN intro_∀_tac(⌜cil2⌝, ⌜cil2⌝)
        THEN list_induction_tac⌜cil1⌝
        THEN asm_rewrite_tac(map get_spec[⌜$|⌝, ⌜Map⌝]));
(* *** Goal "1.1" *** *)
a(rewrite_tac[map_null_thm]);
a(REPEAT strip_tac THEN asm_rewrite_tac[get_spec⌜$|⌝]);
(* *** Goal "1.2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜cil2⌝list_cases_thm) THEN all_var_elim_asm_tac1
        THEN POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
a(REPEAT strip_tac THEN all_asm_fc_tac[]);
a(asm_rewrite_tac[get_spec⌜$|⌝]);
a(cases_tac⌜cc dominates Fst x'⌝ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[]);
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
a(cases_tac⌜¬ ItemBool (Snd x')⌝ THEN asm_rewrite_tac[]);
a(PC_T1 "prop_eq_pair" asm_prove_tac[]);
```

SML

```
(* *** Goal "2" *** *)
a(asm_rewrite_tac[]);
a(lemma_tac⌜Map Fst cil1 = Map Fst cil2⌝);
(* *** Goal "2.1" *** *)
a(LEMMA_T⌜
        Map Fst(Map (λ (c, i)•
                (c, (if cc dominates c then i else Arbitrary))) cil1) =
        Map Fst(Map (λ (c, i)•
                (c, (if cc dominates c then i else Arbitrary))) cil2)⌝
        (ante_tac o rewrite_rule[map_o_lemma])
        THEN1 asm_rewrite_tac[]);
a(LEMMA_T ⌜
        (Fst o (λ (c, i:Item) • (c, (if cc dominates c then i else Arbitrary))))
        = Fst⌝ rewrite_thm_tac);
a(PC_T1"hol2" rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(cases_tac⌜cil2 ↑ {(c, i)|cc dominates c ∧ ItemBool i} = []⌝ THEN
        asm_rewrite_tac[]);
a(strip_tac THEN LEMMA_T ⌜cil1 = cil2⌝ rewrite_thm_tac);
a(lemma_tac⌜cc dominates lubl (Map Fst cil1)⌝ THEN1 asm_rewrite_tac[]);
a(all_fc_tac[dominates_lubl_map_hide_lemma]);
a(swap_nth_asm_concl_tac 8 THEN asm_rewrite_tac[]);
val ComputeOr_lemma = save_pop_thm"ComputeOr_lemma";
```

SML

```
set_goal([],
        ⌜∀c vcl• Elems vcl ⊆ OK_VC_d c ∧ Elems vcl ⊆ OK_VC_c c
        ⇒ BinOpOr c vcl ∈ OK_VC_d c⌝);
a(rewrite_tac(
        map get_spec[⌜BinOpOr⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac THEN swap_nth_asm_concl_tac 1
        THEN POP_ASM_T ante_tac THEN strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_0 rl_0 r_0) vcl) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_1 rl_1 r_1) vcl)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeOr_lemma]]);
a(POP_ASM_T discard_tac THEN
        asm_ante_tac ⌜Elems vcl ⊆ OK_VC_c c⌝ THEN
        asm_ante_tac ⌜Elems vcl ⊆ OK_VC_d c⌝
        THEN list_induction_tac⌜vcl⌝);
```

SML

```
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "3" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "4" *** *)
a(asm_rewrite_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]) THEN REPEAT_N 3 strip_tac);
a(LEMMA_T⌜x ∈ OK_VC_d c ∧ x ∈ OK_VC_c c⌝
        (strip_asm_tac o rewrite_rule(map get_spec[⌜OK_VC_d⌝, ⌜OK_VC_c⌝])) THEN1
        PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[]);
a(cases_tac⌜c dominates Fst (x tl_1 rl_1 r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (x tl_0 rl_0 r_0)⌝ THEN1 asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val BinOpOr_OK_d_lemma = save_pop_thm"BinOpOr_OK_d_lemma";
```

SML

```
set_goal([], ⌜∀c f vc1 vc2• vc1 ∈ OK_VC_d c ∧ vc2 ∈ OK_VC_d c
        ⇒ BinOp f vc1 vc2 ∈ OK_VC_d c⌝);
a(rewrite_tac(dominates_lub_lemma :: map get_spec[⌜OK_VC_d⌝, ⌜BinOp⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜Snd (vc2 tl_0 rl_0 r_0) = Snd (vc2 tl_1 rl_1 r_1)⌝
        THEN1 (contr_tac THEN all_asm_fc_tac[]));
a(lemma_tac⌜¬Snd (vc1 tl_0 rl_0 r_0) = Snd (vc1 tl_1 rl_1 r_1)⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(all_asm_fc_tac[]);
val BinOp_OK_d_lemma = save_pop_thm"BinOp_OK_d_lemma";
```

6.7 Triadic

SML

```
set_goal([], ⌜∀c f vc1 vc2 vc3•
        vc1 ∈ OK_VC_d c ∧ vc2 ∈ OK_VC_d c ∧ vc3 ∈ OK_VC_d c ⇒
        TriOp f vc1 vc2 vc3 ∈ OK_VC_d c⌝);
a(rewrite_tac(dominates_lub_lemma :: map get_spec[⌜OK_VC_d⌝, ⌜TriOp⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜Snd (vc2 tl_0 rl_0 r_0) = Snd (vc2 tl_1 rl_1 r_1) ∧
        Snd (vc3 tl_0 rl_0 r_0) = Snd (vc3 tl_1 rl_1 r_1)⌝
        THEN1 (contr_tac THEN all_asm_fc_tac[]));
a(lemma_tac⌜¬Snd (vc1 tl_0 rl_0 r_0) = Snd (vc1 tl_1 rl_1 r_1)⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(all_asm_fc_tac[]);
val TriOp_OK_d_lemma = save_pop_thm"TriOp_OK_d_lemma";
```

6.8 Conversions

Omitted. see [4].

6.9 Sterling

Omitted. see [4].

6.10 Dinary

Omitted. see [4].

6.11 Declarations

Omitted. see [4].

6.12 Case Expressions

The proof below could probably be reworked to run rather quicker by taking care to move the lemmas proved by *tac1* ... *tac4* before the case split.

SML

```
set_goal([], ⌜∀c te cel ee•
        te ∈ OK_VC_d c ∧
        Elems (Map Fst cel) ⊆ OK_VC_d c ∧
        Elems (Map Snd cel) ⊆ OK_VC_d c ∧
        ee ∈ OK_VC_d c ⇒
        CaseVal c te cel ee ∈ OK_VC_d c
⌝);
a(REPEAT ∀_tac);
a(list_induction_tac⌜cel⌝ THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(rewrite_tac(CaseVal_lemma:: map get_spec[⌜OK_VC_d⌝, ⌜Let⌝])
        THEN REPEAT ∀_tac);
a(rewrite_tac[∀_elim⌜Snd⌝ fun_if_thm]);
a(CASES_T⌜c dominates Fst (te tl_0 rl_0 r_0)⌝ rewrite_thm_tac);
a(CASES_T⌜c dominates Fst (te tl_1 rl_1 r_1)⌝ rewrite_thm_tac);
(* *** Goal "1.1" (duplicates "1.2") *** *)
a(REPEAT strip_tac THEN
        DROP_ASM_T ⌜ee ∈ OK_VC_d c⌝
        (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VC_d⌝])th])
        THEN REPEAT strip_tac);
(*
```

SML

```
*)
(* *** Goal "2" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "3" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "4" *** *)
a(rewrite_tac(∀_elim⌜Snd⌝ fun_if_thm::∀_elim⌜Fst⌝ fun_if_thm::CaseVal_lemma::
        map get_spec[⌜OK_VC_d⌝, ⌜Let⌝])
        THEN REPEAT ∀_tac);
a(MAP_EVERY (fn t => CASES_T t (fn th => rewrite_tac[th] THEN strip_asm_tac th))
        [⌜Snd (te tl_0 rl_0 r_0) = Snd (Fst x tl_0 rl_0 r_0)⌝,
        ⌜c dominates Fst (te tl_0 rl_0 r_0)⌝,
        ⌜c dominates Fst (Fst x tl_0 rl_0 r_0)⌝]);
(*
```

```
SML
*)
(* *** Goal "4.1" *** *)
val tac1 = REPEAT strip_tac THEN
        LEMMA_T ⌜Snd x ∈ OK_VC_d c⌝
        (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VC_d⌝])th])
        THEN1 PC_T1 "sets_ext" asm_prove_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]);
val tac2 = REPEAT strip_tac THEN
        cases_tac⌜Snd (te tl_0 rl_0 r_0) = Snd (te tl_1 rl_1 r_1)⌝
        THEN_LIST [
        asm_ante_tac ⌜¬ Snd (te tl_1 rl_1 r_1) = Snd (Fst x tl_1 rl_1 r_1)⌝
        THEN POP_ASM_T (asm_rewrite_thm_tac o eq_sym_rule)
        THEN REPEAT strip_tac THEN
        LEMMA_T ⌜Fst x ∈ OK_VC_d c⌝
                (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VC_d⌝])th])
        THEN1 PC_T1 "sets_ext" asm_prove_tac(map get_spec[⌜Elems⌝, ⌜Map⌝])
        ,
        DROP_ASM_T ⌜te ∈ OK_VC_d c⌝
        (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VC_d⌝])th])];
a(MAP_EVERY (fn t => CASES_T t (fn th => rewrite_tac[th] THEN strip_asm_tac th))
        [⌜Snd (te tl_1 rl_1 r_1) = Snd (Fst x tl_1 rl_1 r_1)⌝,
        ⌜c dominates Fst (Fst x tl_1 rl_1 r_1)⌝,
        ⌜c dominates Fst (te tl_1 rl_1 r_1)⌝]
        THEN_LIST[tac1, tac1, tac1, tac1, tac2, tac2, tac2, tac2]);
(*
```

```
SML
*)
(* *** Goal "4.2" *** *)
val tac3 = REPEAT strip_tac THEN
        cases_tac⌜Snd (te tl0 rl0 r0) = Snd (te tl1 rl1 r1)⌝
        THEN_LIST [
        asm_ante_tac ⌜¬ Snd (te tl0 rl0 r0) = Snd (Fst x tl0 rl0 r0)⌝
        THEN asm_rewrite_tac[]
        THEN STRIP_T (asm_tac o conv_rule(RAND_C eq_sym_conv)) THEN
        LEMMA_T ⌜Fst x ∈ OK_VCd c⌝
                (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VCd⌝])th])
        THEN1 PC_T1 "sets_ext" asm_prove_tac(map get_spec[⌜Elems⌝, ⌜Map⌝])
        ,
        DROP_ASM_T ⌜te ∈ OK_VCd c⌝
        (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VCd⌝])th])];
val tac4 = REPEAT strip_tac THEN
        DROP_ASM_T ⌜CaseVal c te cel ee ∈ OK_VCd c⌝
        (fn th => all_fc_tac[rewrite_rule(map get_spec[⌜OK_VCd⌝])th]);
a(MAP_EVERY (fn t => CASES_T t (fn th => rewrite_tac[th] THEN strip_asm_tac th))
        [⌜Snd (te tl1 rl1 r1) = Snd (Fst x tl1 rl1 r1)⌝,
         ⌜c dominates Fst (Fst x tl1 rl1 r1)⌝,
         ⌜c dominates Fst (te tl1 rl1 r1)⌝]
        THEN_LIST[tac3, tac3, tac3, tac3, tac4, tac4, tac4, tac4]);
val CaseVal_OKd_lemma = save_pop_thm"CaseVal_OKd_lemma";
```

SML

```
set_goal([], ⌜∀c cel ee•
        Elems (Map Fst cel) ⊆ OK_VC_d c ∧
        Elems (Map Snd cel) ⊆ OK_VC_d c ∧
        ee ∈ OK_VC_d c ⇒
        Case c cel ee ∈ OK_VC_d c
⌝);
a(rewrite_tac(map get_spec[⌜Let⌝,⌜Case⌝]));
a(REPEAT ∀_tac);
a(list_induction_tac⌜cel⌝ THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN
        rewrite_tac(map get_spec[⌜OK_VC_d⌝, ⌜Let⌝,⌜Map⌝,⌜CaseC⌝,⌜CaseValue⌝])
        THEN REPEAT strip_tac);
(* *** Goal "2" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "3" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
```

SML

```
(* *** Goal "4" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN
        rewrite_tac(map get_spec[ ⌜Let⌝,⌜Map⌝,⌜CaseC⌝,⌜CaseValue⌝,⌜OK_VC_d⌝])
        THEN REPEAT strip_tac);
a(DROP_NTH_ASM_T 5 (ante_tac o list_∀_elim
        [⌜tl_0⌝,⌜tl_1⌝,⌜rl_0⌝,⌜rl_1⌝,⌜r_0⌝,⌜r_1⌝])
        THEN asm_rewrite_tac[]);
a(POP_ASM_T ante_tac
        THEN cases_tac⌜¬ c dominates Fst (Fst x tl_0 rl_0 r_0)⌝
        THEN cases_tac⌜ItemBool (Snd (Fst x tl_0 rl_0 r_0))⌝
        THEN cases_tac⌜ItemBool (Snd (Fst x tl_1 rl_1 r_1))⌝

        THEN asm_rewrite_tac[]
        THEN REPEAT strip_tac);
(* 4.1 and 4.2 the same except for asm 1 *)
```

SML

```
(* *** Goal "4.1" *** *)
a(POP_ASM_T (fn x => id_tac));
set_labelled_goal"4.2";
(* *** Goal "4.2" *** *)
a(POP_ASM_T (fn x => id_tac));
a(lemma_tac⌜Snd x ∈ OK_VC_d c⌝
        THEN_LIST[PC_T1 "sets_ext" asm_prove_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]),
                all_asm_fc_tac[get_spec⌜OK_VC_d⌝]]);
(* *** Goal "4.3" *** *)
(* 4.3 and 4.4 the same except for asm 1 *)
a(POP_ASM_T (fn x => id_tac));
set_labelled_goal"4.4";
```

SML

```
(* *** Goal "4.4" *** *)
a(POP_ASM_T (fn x => id_tac));
a(lemma_tac⌜Fst x ∈ OK_VC_d c⌝
        THEN1 PC_T1 "sets_ext" asm_prove_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]));
a(lemma_tac⌜¬ Snd (Fst x tl_0 rl_0 r_0) = Snd (Fst x tl_1 rl_1 r_1)⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(all_asm_fc_tac[get_spec⌜OK_VC_d⌝]);
(* *** Goal "4.5" *** *)
a(lemma_tac⌜Fst x ∈ OK_VC_d c⌝
        THEN1 PC_T1 "sets_ext" asm_prove_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]));
a(lemma_tac⌜¬ Snd (Fst x tl_0 rl_0 r_0) = Snd (Fst x tl_1 rl_1 r_1)⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(all_asm_fc_tac[get_spec⌜OK_VC_d⌝]);
val Case_OK_d_lemma = save_pop_thm"Case_OK_d_lemma";
```

6.13 Set Functions

The main lemma (the second of the two below) is an example of strengthening an induction hypothesis almost beyond belief. The point is that, in the goal as stated, both rl_0 and rl_1 appear in two different ways, viz. as an argument to the "recursive" call on *vc* and as the operand for what we're doing. To get a strong enough induction hypothesis we have to quantify separately over the the row lists appearing in these different roles.

SML

```
set_goal([], ⌜∀c f vc• vc ∈ OK_VC_d c ⇒ SetFuncAll f vc ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜SetFuncAll⌝, ⌜OK_VC_d⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜¬ Snd(Split(Map(vc tl_0 rl_0) rl_0)) =
                Snd(Split(Map (vc tl_1 rl_1) rl_1))⌝
        THEN1 PC_T1 "prop_eq" asm_prove_tac[]);
a(lemma_tac ⌜∀rl1 rl2 rl_1•
        Map (HideDerTableRow c) rl1 = Map (HideDerTableRow c) rl2 ∧
        Map (HideDerTableRow c) rl_0 = Map (HideDerTableRow c) rl_1
            ⇒ ¬ (Snd (Split (Map (vc tl_0 rl1) rl_0)))
                = (Snd (Split (Map (vc tl_1 rl2) rl_1)))
            ⇒ ¬ c dominates lubl (Fst (Split (Map (vc tl_0 rl1) rl_0)))⌝
        THEN_LIST[id_tac, all_asm_fc_tac[]]);
a(LIST_DROP_NTH_ASM_T [1, 2, 3, 4] (Combinators.K id_tac));
a(list_induction_tac⌜rl_0⌝ THEN REPEAT strip_tac);
```

SML

```
(* *** Goal "1" *** *)
a(strip_asm_tac (∀_elim⌜rl_1⌝ list_cases_thm));
(* *** Goal "1.1" *** *)
a(swap_asm_concl_tac
        ⌜¬ (Snd (Split (Map (vc tl_0 rl1) [])))
                = (Snd (Split (Map (vc tl_1 rl2) rl_1)))⌝
        THEN asm_rewrite_tac(map get_spec[⌜Split⌝, ⌜Map⌝]));
(* *** Goal "1.2" *** *)
a(swap_asm_concl_tac
        ⌜Map (HideDerTableRow c) [] = Map (HideDerTableRow c) rl_1⌝
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
```

SML

```
(* *** Goal "2" *** *)
a(strip_asm_tac (∀_elim⌜rl1⌝ list_cases_thm));
(* *** Goal "2.1" *** *)
a(swap_asm_concl_tac
        ⌜Map (HideDerTableRow c) (Cons x rl0) = Map (HideDerTableRow c) rl1⌝
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(rewrite_tac[fst_snd_split_thm, lubl_lemma,
        get_spec⌜Map⌝, dominates_lub_lemma]);
a(swap_asm_concl_tac
        ⌜¬ (Snd (Split (Map (vc tl0 rl1) (Cons x rl0))))
                = (Snd (Split (Map (vc tl1 rl2) (Cons x' list2))))⌝
        THEN asm_rewrite_tac[fst_snd_split_thm, lubl_lemma,
        get_spec⌜Map⌝, dominates_lub_lemma]);
a(GET_NTH_ASM_T 3 (strip_asm_tac o rewrite_rule[map_def]));
a(DROP_NTH_ASM_T 7 (strip_asm_tac o rewrite_rule[fst_snd_split_thm]));
a(contr_tac THEN all_asm_fc_tac[]);
val SetFuncAll_OKd_lemma = save_pop_thm"SetFuncAll_OKd_lemma";
```

SML

```
set_goal([], ⌜∀c f vc• vc ∈ OK_VCd c ⇒ SetFuncDistinct f vc ∈ OK_VCd c⌝);
a(REPEAT strip_tac);
a(lemma_tac⌜SetFuncDistinct f vc = SetFuncAll (f o Elems) vc⌝
        THEN_LIST [rewrite_tac(map get_spec[⌜SetFuncDistinct⌝, ⌜SetFuncAll⌝, ⌜Let⌝]),
                POP_ASM_T rewrite_thm_tac]);
a(bc_tac[SetFuncAll_OKd_lemma]THEN asm_rewrite_tac[]);
val SetFuncDistinct_OKd_lemma = save_pop_thm"SetFuncDistinct_OKd_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ∧ vc ∈ OK_VC_c c
        ⇒ SetFuncAllAnd c vc ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜SetFuncAllAnd⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac THEN swap_nth_asm_concl_tac 1
        THEN POP_ASM_T ante_tac THEN strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl_0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl_1)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeAnd_lemma]]);
a(LEMMA_T⌜
        ∀rl0 rl1•
        Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
        ⇒
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl1)⌝
        (fn th => all_fc_tac[th]));
a(strip_tac);
a(list_induction_tac⌜rl0⌝ THEN asm_rewrite_tac[map_null_thm, get_spec⌜Map⌝]);
(* rewrite solves base case *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝ list_cases_thm) THEN all_var_elim_asm_tac1);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
a(strip_tac THEN ALL_ASM_FC_T rewrite_tac[]);
a(cases_tac⌜c dominates Fst (vc tl_1 rl_1 x')⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (vc tl_0 rl_0 x)⌝
        THEN1 ALL_ASM_FC_T asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val SetFuncAllAnd_OK_d_lemma = save_pop_thm"SetFuncAllAnd_OK_d_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ∧ vc ∈ OK_VC_c c ⇒ SetFuncAllOr c vc ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜SetFuncAllOr⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac THEN swap_nth_asm_concl_tac 1
        THEN POP_ASM_T ante_tac THEN strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl_0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl_1)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeOr_lemma]]);
a(LEMMA_T⌜
        ∀rl0 rl1•
        Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
        ⇒
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl1)⌝
        (fn th => all_fc_tac[th]));
a(strip_tac);
a(list_induction_tac⌜rl0⌝ THEN asm_rewrite_tac[map_null_thm, get_spec⌜Map⌝]);
(* rewrite solves base case *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝ list_cases_thm) THEN all_var_elim_asm_tac1);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
a(strip_tac THEN ALL_ASM_FC_T rewrite_tac[]);
a(cases_tac⌜c dominates Fst (vc tl_1 rl_1 x')⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (vc tl_0 rl_0 x)⌝
        THEN1 ALL_ASM_FC_T asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val SetFuncAllOr_OK_d_lemma = save_pop_thm"SetFuncAllOr_OK_d_lemma";
```

6.14 Count Functions

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ⇒ CountNonNull vc ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜CountNonNull⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(bc_tac[SetFuncAll_OK_d_lemma]THEN asm_rewrite_tac[]);
val CountNonNull_OK_d_lemma = save_pop_thm"CountNonNull_OK_d_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ⇒ CountDistinct vc ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜CountDistinct⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(bc_tac[SetFuncDistinct_OK_d_lemma]THEN asm_rewrite_tac[]);
val CountDistinct_OK_d_lemma = save_pop_thm"CountDistinct_OK_d_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ⇒ CommonValue vc ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜CommonValue⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(ALL_FC_T rewrite_tac [SetFuncAll_OK_d_lemma]);
val CommonValue_OK_d_lemma = save_pop_thm"CommonValue_OK_d_lemma";
```

6.15 *ExistsTuples*

SML

```
set_goal([], ⌜∀c tc• tc ∈ OK_TC_d c ∧ tc ∈ OK_TC_c c ⇒ ExistsTuples c tc ∈ OK_VC_d c⌝);
a(rewrite_tac(BoolItem_OneOne_lemma :: dominates_lub_lemma ::
        ∀_elim⌜Fst⌝fun_if_thm :: ∀_elim⌜Snd⌝fun_if_thm ::
        map get_spec[⌜OK_TC_d⌝, ⌜OK_TC_c⌝, ⌜OK_VC_d⌝, ⌜ExistsTuples⌝, ⌜Let⌝]));
a(REPEAT strip_tac THEN POP_ASM_T ante_tac);
a(all_asm_fc_tac[]);
a(asm_rewrite_tac[]);
a(cases_tac ⌜c dominates Fst (tc tl_1)⌝ THEN asm_rewrite_tac[BoolItem_OneOne_lemma]);
a(cases_tac ⌜HideDerTable c (Snd (tc tl_0)) = HideDerTable c (Snd (tc tl_1))⌝);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac(MkDerTable_lemma::
        map get_spec[⌜HideDerTable⌝, ⌜HideDerTableData⌝, ⌜Let⌝]));
a(strip_tac);
a(once_rewrite_tac[↑_null_map_hide_lemma]);
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(all_asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
val ExistsTuples_OK_d_lemma = save_pop_thm"ExistsTuples_OK_d_lemma";
```

6.16 *SingleValue*

SML

```
set_goal([], ⌜∀c tc• tc ∈ OK_TC_d c ∧ tc ∈ OK_TC_c c ⇒ SingleValue c tc ∈ OK_VC_d c⌝);
a(rewrite_tac(BoolItem_OneOne_lemma :: dominates_lub_lemma ::
        ∀_elim⌜Fst⌝fun_if_thm :: ∀_elim⌜Snd⌝fun_if_thm ::
        map get_spec[⌜OK_TC_d⌝, ⌜OK_TC_c⌝, ⌜OK_VC_d⌝, ⌜SingleValue⌝, ⌜Let⌝]));
a(REPEAT strip_tac THEN POP_ASM_T ante_tac);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[]);
a(cases_tac ⌜c dominates Fst (tc tl_1)⌝ THEN asm_rewrite_tac[]);
a(cases_tac ⌜¬HideDerTable c (Snd (tc tl_0)) = HideDerTable c (Snd (tc tl_1))⌝);
(* *** Goal "1" *** *)
a(all_asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
```

SML

```
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac(MkDerTable_lemma::
        pc_rule1 "sets_ext1" prove_rule[]
        ⌜{r|c dominates DTR_row r ∧ c dominates DTR_where r} =
        {r|c dominates DTR_row r} ∩ {r|c dominates DTR_where r}⌝::
        map get_spec[⌜HideDerTable⌝, ⌜HideDerTableData⌝, ⌜Let⌝]));
a(rewrite_tac[↾_∩_lemma] THEN strip_tac);
a(all_fc_tac[map_hide_map_hide_↾_lemma]);
a(LEMMA_T⌜
        #(Map (HideDerTableRow c)
                ((DT_rows (Snd (tc tl_0)) ↾ {r|c dominates DTR_row r})
                        ↾ {r|c dominates DTR_where r})) =
        #(Map (HideDerTableRow c)
                ((DT_rows (Snd (tc tl_1)) ↾ {r|c dominates DTR_row r})
                        ↾ {r|c dominates DTR_where r}))⌝
        (strip_asm_tac o rewrite_rule[length_map_thm])
        THEN1 asm_rewrite_tac[]);
```

SML

```
a(cases_tac⌜
        #((DT_rows (Snd (tc tl1)) ↑ {r|c dominates DTR_row r})
                        ↑ {r|c dominates DTR_where r}) = 1⌝
        THEN asm_rewrite_tac[]);
a(lemma_tac⌜
        #((DT_rows (Snd (tc tl0)) ↑ {r|c dominates DTR_row r})
                        ↑ {r|c dominates DTR_where r}) = 1⌝
        THEN1 asm_rewrite_tac[]);
a(LIST_DROP_NTH_ASM_T[1,2]
        (MAP_EVERY(strip_asm_tac o rewrite_rule[length_1_thm])));
a(asm_rewrite_tac[]);
a(lemma_tac⌜HideDerTableRow c x = HideDerTableRow c x'⌝);
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
```

SML

```
(* *** Goal "2.2" *** *)
a(POP_ASM_T ante_tac THEN
        rewrite_tac[MkDerTableRow_lemma, let_def, get_spec⌜HideDerTableRow⌝]);
a(strip_tac);
a(LEMMA_T⌜
        #(Map (λ (c', i)• if c dominates c'
                then (c', i)
                else (c', ValuedItemItem (MkValuedItem sterling dummyVal)))
              (DTR_cols x)) =
        #(Map (λ (c', i)• if c dominates c'
                then (c', i)
                else (c', ValuedItemItem (MkValuedItem sterling dummyVal)))
              (DTR_cols x'))⌝
        (strip_asm_tac o rewrite_rule[length_map_thm])
        THEN1 asm_rewrite_tac[]);
```

SML

```
a(asm_rewrite_tac[]);
a(cases_tac⌜# (DTR_cols x') = 1⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜# (DTR_cols x) = 1⌝ THEN1 asm_rewrite_tac[]);
a(LIST_DROP_NTH_ASM_T[1,2]
        (MAP_EVERY(strip_asm_tac o rewrite_rule[length_1_thm])));
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
a(cases_tac⌜c dominates Fst x''⌝ THEN asm_rewrite_tac[]);
a(cases_tac⌜c dominates Fst x'''⌝ THEN asm_rewrite_tac[]
        THEN REPEAT strip_tac THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 3 ante_tac THEN asm_rewrite_tac[]);
val SingleValue_OK_d_lemma = save_pop_thm"SingleValue_OK_d_lemma";
```

6.17 *JoinedRowExistence*

SML

```
set_goal([], ⌜∀c i• JoinedRowExistence i ∈ OK_VC_d c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_d⌝, ⌜JoinedRowExistence⌝])
        THEN REPEAT strip_tac);
a(GET_NTH_ASM_T 2 ante_tac THEN
        rewrite_tac(MkDerTableRow_lemma::map get_spec[⌜HideDerTableRow⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(PC_T1 "prop_eq" asm_prove_tac[]);
val JoinedRowExistence_OK_d_lemma = save_pop_thm"JoinedRowExistence_OK_d_lemma";
```

7 CLASSIFICATION OKNESS PROOFS

7.1 Constant Expression

SML

```
set_goal([], ⌜∀c ci• DenoteConstant ci ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜DenoteConstant⌝]) THEN REPEAT strip_tac);
val DenoteConstant_OK_c_lemma = save_pop_thm"DenoteConstant_OK_c_lemma";
```

7.2 *Contents*

```
SML
set_goal([], ⌜∀c i• Contents i ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜Contents⌝])
        THEN REPEAT strip_tac);
a(all_fc_tac[HideDerTableRow_Length_lemma]);
a(cases_tac ⌜1 ≤ i ∧ i ≤ # (DTR_cols r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜i ≤ # (DTR_cols r_0)⌝ THEN1 asm_rewrite_tac[]);
a(ALL_FC_T (MAP_EVERY(ante_tac o ∀_elim⌜c⌝)) [Nth_HideDerTableRow_lemma]);
a(asm_rewrite_tac[]);
a(PC_T1 "prop_eq_pair" prove_tac[]);
val Contents_OK_c_lemma = save_pop_thm"Contents_OK_c_lemma";
```

7.3 *Classification*

```
SML
set_goal([], ⌜∀c i• Classification i ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜Classification⌝])
        THEN REPEAT strip_tac);
a(all_fc_tac[HideDerTableRow_Length_lemma]);
a(cases_tac ⌜1 ≤ i ∧ i ≤ # (DTR_cols r_1)⌝ THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 4 ante_tac THEN
        rewrite_tac(MkDerTableRow_lemma::map get_spec[⌜Let⌝, ⌜HideDerTableRow⌝])
        THEN taut_tac);
val Classification_OK_c_lemma = save_pop_thm"Classification_OK_c_lemma";
```

7.4 *Countall*

SML

```
set_goal([], ⌜∀c• CountAll ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜CountAll⌝,⌜Let⌝]) THEN REPEAT strip_tac);
a(LIST_DROP_NTH_ASM_T [1,3] discard_tac);
a(POP_ASM_T ante_tac);
a(intro_∀_tac(⌜rl_1⌝,⌜rl_1⌝));
a(list_induction_tac⌜rl_0⌝);
(* *** Goal "1" *** *)
a(∀_tac);
a(strip_asm_tac (∀_elim⌜rl_1⌝ list_cases_thm)
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "2" *** *)
a(REPEAT ∀_tac);
a(strip_asm_tac (∀_elim⌜rl_1⌝ list_cases_thm)
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
a(rewrite_tac(lubl_lemma::MkDerTableRow_lemma::map get_spec[⌜Let⌝, ⌜HideDerTableRow⌝])
        THEN REPEAT strip_tac);
a(ALL_ASM_FC_T asm_rewrite_tac[]);
val CountAll_OK_c_lemma = save_pop_thm"CountAll_OK_c_lemma";
```

7.5 Monadic

SML

```
set_goal([], ⌜∀c f vc• vc ∈ OK_VC_c c ⇒ MonOp f vc ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜MonOp⌝, ⌜Let⌝]) THEN REPEAT strip_tac);
val MonOp_OK_c_lemma = save_pop_thm"MonOp_OK_c_lemma";
```

7.6 Binary

SML

```
set_goal([],
        ⌜∀c vcl• Elems vcl ⊆ OK_VC_d c ∧ Elems vcl ⊆ OK_VC_c c
        ⇒ BinOpAnd c vcl ∈ OK_VC_c c⌝);
a(rewrite_tac(
        map get_spec[⌜BinOpAnd⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_0 rl_0 r_0) vcl) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_1 rl_1 r_1) vcl)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeAnd_lemma]]);
a(asm_ante_tac ⌜Elems vcl ⊆ OK_VC_c c⌝ THEN asm_ante_tac ⌜Elems vcl ⊆ OK_VC_d c⌝
        THEN list_induction_tac⌜vcl⌝);
```

SML

```
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "3" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "4" *** *)
a(asm_rewrite_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]) THEN REPEAT_N 3 strip_tac);
a(LEMMA_T⌜x ∈ OK_VC_d c ∧ x ∈ OK_VC_c c⌝
        (strip_asm_tac o rewrite_rule(map get_spec[⌜OK_VC_d⌝, ⌜OK_VC_c⌝])) THEN1
        PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[]);
a(cases_tac⌜c dominates Fst (x tl_1 rl_1 r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (x tl_0 rl_0 r_0)⌝ THEN1 asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val BinOpAnd_OK_c_lemma = save_pop_thm"BinOpAnd_OK_c_lemma";
```

SML

```
set_goal([],
        ⌜∀c vcl• Elems vcl ⊆ OK_VC_d c ∧ Elems vcl ⊆ OK_VC_c c
        ⇒ BinOpOr c vcl ∈ OK_VC_c c⌝);
a(rewrite_tac(
        map get_spec[⌜BinOpOr⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_0 rl_0 r_0) vcl) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (λ e• e tl_1 rl_1 r_1) vcl)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeOr_lemma]]);
a(asm_ante_tac ⌜Elems vcl ⊆ OK_VC_c c⌝ THEN asm_ante_tac ⌜Elems vcl ⊆ OK_VC_d c⌝
        THEN list_induction_tac⌜vcl⌝);
```

SML

```
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "3" *** *)
a(PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
(* *** Goal "4" *** *)
a(asm_rewrite_tac(map get_spec[⌜Elems⌝, ⌜Map⌝]) THEN REPEAT_N 3 strip_tac);
a(LEMMA_T⌜x ∈ OK_VC_d c ∧ x ∈ OK_VC_c c⌝
        (strip_asm_tac o rewrite_rule(map get_spec[⌜OK_VC_d⌝, ⌜OK_VC_c⌝])) THEN1
        PC_T1"sets_ext1" asm_prove_tac[get_spec⌜Elems⌝]);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[]);
a(cases_tac⌜c dominates Fst (x tl_1 rl_1 r_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (x tl_0 rl_0 r_0)⌝ THEN1 asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val BinOpOr_OK_c_lemma = save_pop_thm"BinOpOr_OK_c_lemma";
```

SML

```
set_goal([], ⌜∀c f vc1 vc2• vc1 ∈ OK_VC_c c ∧ vc2 ∈ OK_VC_c c
        ⇒ BinOp f vc1 vc2 ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜BinOp⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(ALL_ASM_FC_T rewrite_tac[]);
val BinOp_OK_c_lemma = save_pop_thm"BinOp_OK_c_lemma";
```

7.7 Triadic

SML

```
set_goal([], ⌜∀c f vc1 vc2 vc3•
        vc1 ∈ OK_VC_c c ∧ vc2 ∈ OK_VC_c c ∧ vc3 ∈ OK_VC_c c ⇒
        TriOp f vc1 vc2 vc3 ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜TriOp⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(ALL_ASM_FC_T rewrite_tac[]);
val TriOp_OK_c_lemma = save_pop_thm"TriOp_OK_c_lemma";
```

7.8 Conversions

Omitted. see [4].

7.9 Sterling

Omitted. see [4].

7.10 Dinary

Omitted. see [4].

7.11 Declarations

Omitted. see [4].

7.12 Case Expressions

SML

```
set_goal([], ⌜∀c te cel ee•
        te ∈ OK_VC_d c ∧
        te ∈ OK_VC_c c ∧
        Elems (Map Fst cel) ⊆ OK_VC_d c ∧
        Elems (Map Fst cel) ⊆ OK_VC_c c ∧
        Elems (Map Snd cel) ⊆ OK_VC_c c ∧
        ee ∈ OK_VC_c c ⇒
        CaseVal c te cel ee ∈ OK_VC_c c
⌝);
a(REPEAT ∀_tac);
a(list_induction_tac⌜cel⌝ THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(LIST_DROP_NTH_ASM_T [1,5] (MAP_EVERY ante_tac)
        THEN rewrite_tac(CaseVal_lemma:: map get_spec[⌜OK_VC_c⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(rewrite_tac[∀_elim⌜Fst⌝ fun_if_thm]);
a(ALL_ASM_FC_T rewrite_tac[]);
```

SML

```
(* *** Goal "2" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "3" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "4" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "5" *** *)
a(rewrite_tac(∀_elim⌜Snd⌝ fun_if_thm::∀_elim⌜Fst⌝ fun_if_thm::CaseVal_lemma::
        map get_spec[⌜OK_VC_c⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜
        Fst (te tl_0 rl_0 r_0) = Fst (te tl_1 rl_1 r_1) ∧
        Fst (Fst x tl_0 rl_0 r_0) = Fst (Fst x tl_1 rl_1 r_1) ∧
        Fst (Snd x tl_0 rl_0 r_0) = Fst (Snd x tl_1 rl_1 r_1) ∧
        Fst (CaseVal c te cel ee tl_0 rl_0 r_0) =
                Fst (CaseVal c te cel ee tl_1 rl_1 r_1)⌝);
```

SML

```
(* *** Goal "5.1" *** *)
a(lemma_tac ⌜Fst x ∈ OK_VC_c c ∧ Snd x ∈ OK_VC_c c⌝
        THEN1 PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
a(LIST_DROP_NTH_ASM_T [1,2,10,12]
        (MAP_EVERY(strip_asm_tac o rewrite_rule[get_spec⌜OK_VC_c⌝])));
a(ALL_ASM_FC_T rewrite_tac[]);
(* *** Goal "5.2" *** *)
a(asm_rewrite_tac[]);
a(cases_tac⌜c dominates Fst (te tl_1 rl_1 r_1)⌝
        THEN cases_tac⌜c dominates Fst (Fst x tl_1 rl_1 r_1)⌝
        THEN asm_rewrite_tac[]);
(* *** Goal "5.2.1" *** *)
a(LEMMA_T⌜Snd (te tl_0 rl_0 r_0) = Snd (te tl_1 rl_1 r_1)
        ∧ Snd (Fst x tl_0 rl_0 r_0) = Snd (Fst x tl_1 rl_1 r_1)⌝
        rewrite_thm_tac);
a(lemma_tac ⌜Fst x ∈ OK_VC_d c⌝
        THEN1 PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
a(LIST_DROP_NTH_ASM_T [1,16]
        (MAP_EVERY(strip_asm_tac o rewrite_rule[get_spec⌜OK_VC_d⌝])));
a(lemma_tac⌜c dominates Fst (te tl_0 rl_0 r_0)
        ∧       c dominates Fst (Fst x tl_0 rl_0 r_0)⌝
        THEN1 asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
```

SML

```
(* *** Goal "5.2.2" *** *)
a(CASES_T ⌜Snd (te tl_0 rl_0 r_0) = Snd (Fst x tl_0 rl_0 r_0)⌝
        rewrite_thm_tac
        THEN CASES_T⌜Snd (te tl_1 rl_1 r_1) = Snd (Fst x tl_1 rl_1 r_1)⌝
        rewrite_thm_tac);
(* *** Goal "5.2.3" *** *)
a(CASES_T ⌜Snd (te tl_0 rl_0 r_0) = Snd (Fst x tl_0 rl_0 r_0)⌝
        rewrite_thm_tac
        THEN CASES_T⌜Snd (te tl_1 rl_1 r_1) = Snd (Fst x tl_1 rl_1 r_1)⌝
        rewrite_thm_tac);
(* *** Goal "5.2.4" *** *)
a(CASES_T ⌜Snd (te tl_0 rl_0 r_0) = Snd (Fst x tl_0 rl_0 r_0)⌝
        rewrite_thm_tac
        THEN CASES_T⌜Snd (te tl_1 rl_1 r_1) = Snd (Fst x tl_1 rl_1 r_1)⌝
        rewrite_thm_tac);
val CaseVal_OK_c_lemma = save_pop_thm"CaseVal_OK_c_lemma";
```

SML

```
set_goal([], ⌜∀c cel ee•
        Elems (Map Fst cel) ⊆ OK_VC_d c ∧
        Elems (Map Fst cel) ⊆ OK_VC_c c ∧
        Elems (Map Snd cel) ⊆ OK_VC_c c ∧
        ee ∈ OK_VC_c c ⇒
        Case c cel ee ∈ OK_VC_c c
⌝);
a(rewrite_tac(map get_spec[⌜Let⌝,⌜Case⌝]));
a(REPEAT ∀_tac);
a(list_induction_tac⌜cel⌝ THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN
        rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜Let⌝,⌜Map⌝,⌜CaseC⌝,⌜CaseValue⌝])
        THEN REPEAT strip_tac);
(* *** Goal "2" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "3" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
(* *** Goal "4" *** *)
a(PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
```

SML

```
(* *** Goal "5" *** *)
a(rewrite_tac(map get_spec[ ⌜Let⌝,⌜Map⌝,⌜CaseC⌝,⌜CaseValue⌝,⌜OK_VC_c⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜Fst x ∈ OK_VC_c c⌝ THEN1
        PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
a(POP_ASM_T (fn th => all_asm_fc_tac[rewrite_rule[get_spec⌜OK_VC_c⌝]th]));
a(asm_rewrite_tac[]);
a(cases_tac⌜¬ c dominates Fst (Fst x tl_1 rl_1 r_1)⌝
        THEN asm_rewrite_tac[]);
a(lemma_tac⌜Snd x ∈ OK_VC_c c⌝ THEN1
        PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
a(POP_ASM_T (fn th => all_asm_fc_tac[rewrite_rule[get_spec⌜OK_VC_c⌝]th]));
a(asm_rewrite_tac[]);
a(LEMMA_T⌜Snd (Fst x tl_0 rl_0 r_0) = Snd (Fst x tl_1 rl_1 r_1)⌝
        rewrite_thm_tac);
(* *** Goal "5.1" *** *)
a(lemma_tac⌜Fst x ∈ OK_VC_d c⌝ THEN1
        PC_T1 "sets_ext1" asm_prove_tac(map get_spec[⌜Map⌝, ⌜Elems⌝]));
a(POP_ASM_T (strip_asm_tac o rewrite_rule[get_spec⌜OK_VC_d⌝]));
a(lemma_tac⌜c dominates Fst (Fst x tl_0 rl_0 r_0)⌝
        THEN1 asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
(* *** Goal "5.2" *** *)
a(cases_tac⌜ItemBool (Snd (Fst x tl_1 rl_1 r_1))⌝
        THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 12 (fn th => all_asm_fc_tac[rewrite_rule[get_spec⌜OK_VC_c⌝]th]));
val Case_OK_c_lemma = save_pop_thm"Case_OK_c_lemma";
```

7.13 Set Functions

SML

```
set_goal([], ⌜∀c f vc• vc ∈ OK_VC_c c ⇒ SetFuncAll f vc ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜SetFuncAll⌝, ⌜OK_VC_c⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(lemma_tac ⌜∀tl_0 tl_1 rl1 rl2 rl_1•
        Map (HideDerTable c) tl_0 = Map (HideDerTable c) tl_1 ∧
        Map (HideDerTableRow c) rl1 = Map (HideDerTableRow c) rl2 ∧
        Map (HideDerTableRow c) rl_0 = Map (HideDerTableRow c) rl_1
            ⇒ Fst (Split (Map (vc tl_0 rl1) rl_0))
                = Fst(Split (Map (vc tl_1 rl2) rl_1))⌝
        THEN_LIST[id_tac, ALL_ASM_FC_T rewrite_tac[]]);
a(LIST_DROP_NTH_ASM_T [1, 2, 3] discard_tac);
a(list_induction_tac⌜rl_0⌝ THEN REPEAT strip_tac);
```

SML

```
(* *** Goal "1" *** *)
a(strip_asm_tac (∀_elim⌜rl_1⌝ list_cases_thm));
(* *** Goal "1.1" *** *)
a(asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "1.2" *** *)
a(var_elim_nth_asm_tac 1);
a(all_asm_ante_tac THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
```

SML

```
(* *** Goal "2" *** *)
a(strip_asm_tac (∀_elim⌜rl_1⌝ list_cases_thm));
(* *** Goal "2.1" *** *)
a(swap_asm_concl_tac
        ⌜Map (HideDerTableRow c) (Cons x rl_0) = Map (HideDerTableRow c) rl_1⌝
        THEN asm_rewrite_tac(map get_spec[⌜Map⌝]));
(* *** Goal "2.2" *** *)
a(var_elim_nth_asm_tac 1);
a(all_asm_ante_tac THEN rewrite_tac[fst_snd_split_thm, get_spec⌜Map⌝]
        THEN REPEAT strip_tac);
(* *** Goal "2.2.1" *** *)
a(all_asm_fc_tac[]);
(* *** Goal "2.2.2" *** *)
a(all_asm_fc_tac[]);
val SetFuncAll_OK_c_lemma = save_pop_thm"SetFuncAll_OK_c_lemma";
```

SML

```
set_goal([], ⌜∀c f vc• vc ∈ OK_VC_c c ⇒ SetFuncDistinct f vc ∈ OK_VC_c c⌝);
a(REPEAT strip_tac);
a(lemma_tac⌜SetFuncDistinct f vc = SetFuncAll (f o Elems) vc⌝
        THEN_LIST [rewrite_tac(map get_spec[⌜SetFuncDistinct⌝, ⌜SetFuncAll⌝, ⌜Let⌝]),
                POP_ASM_T rewrite_thm_tac]);
a(bc_tac[SetFuncAll_OK_c_lemma]THEN asm_rewrite_tac[]);
val SetFuncDistinct_OK_c_lemma = save_pop_thm"SetFuncDistinct_OK_c_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ∧ vc ∈ OK_VC_c c
        ⇒ SetFuncAllAnd c vc ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜SetFuncAllAnd⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl_0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl_1)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeAnd_lemma]]);
a(LEMMA_T⌜
        ∀rl0 rl1•
        Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
        ⇒
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl1)⌝
        (fn th => all_fc_tac[th]));
```

SML

```
a(strip_tac);
a(list_induction_tac⌜rl0⌝ THEN asm_rewrite_tac[map_null_thm, get_spec⌜Map⌝]);
(* rewrite solves base case *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝ list_cases_thm) THEN all_var_elim_asm_tac1);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
a(strip_tac THEN ALL_ASM_FC_T rewrite_tac[]);
a(cases_tac⌜c dominates Fst (vc tl_1 rl_1 x')⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (vc tl_0 rl_0 x)⌝
        THEN1 ALL_ASM_FC_T asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val SetFuncAllAnd_OK_c_lemma = save_pop_thm"SetFuncAllAnd_OK_c_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VC_d c ∧ vc ∈ OK_VC_c c ⇒ SetFuncAllOr c vc ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜SetFuncAllOr⌝, ⌜OK_VC_d⌝, ⌜OK_VC_c⌝])
        THEN REPEAT strip_tac);
a(lemma_tac⌜
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl_0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl_1)⌝
        THEN_LIST [id_tac, all_fc_tac[ComputeOr_lemma]]);
a(LEMMA_T⌜
        ∀rl0 rl1•
        Map (HideDerTableRow c) rl0 = Map (HideDerTableRow c) rl1
        ⇒
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_0 rl_0) rl0) =
        Map (λ (c', i)• (c', (if c dominates c' then i else Arbitrary)))
        (Map (vc tl_1 rl_1) rl1)⌝
        (fn th => all_fc_tac[th]));
```

SML

```
a(strip_tac);
a(list_induction_tac⌜rl0⌝ THEN asm_rewrite_tac[map_null_thm, get_spec⌜Map⌝]);
(* rewrite solves base case *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜rl1⌝ list_cases_thm) THEN all_var_elim_asm_tac1);
(* *** Goal "1" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN rewrite_tac[get_spec⌜Map⌝]);
a(strip_tac THEN ALL_ASM_FC_T rewrite_tac[]);
a(cases_tac⌜c dominates Fst (vc tl1 rl1 x')⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (vc tl0 rl0 x)⌝
        THEN1 ALL_ASM_FC_T asm_rewrite_tac[]);
a(contr_tac THEN all_asm_fc_tac[]);
val SetFuncAllOr_OKc_lemma = save_pop_thm"SetFuncAllOr_OKc_lemma";
```

7.14 Count Functions

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VCc c ⇒ CountNonNull vc ∈ OK_VCc c⌝);
a(rewrite_tac(map get_spec[⌜CountNonNull⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(bc_tac[SetFuncAll_OKc_lemma]THEN asm_rewrite_tac[]);
val CountNonNull_OKc_lemma = save_pop_thm"CountNonNull_OKc_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VCc c ⇒ CountDistinct vc ∈ OK_VCc c⌝);
a(rewrite_tac(map get_spec[⌜CountDistinct⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(bc_tac[SetFuncDistinct_OKc_lemma]THEN asm_rewrite_tac[]);
val CountDistinct_OKc_lemma = save_pop_thm"CountDistinct_OKc_lemma";
```

SML

```
set_goal([], ⌜∀c vc• vc ∈ OK_VCc c ⇒ CommonValue vc ∈ OK_VCc c⌝);
a(rewrite_tac(map get_spec[⌜CommonValue⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(ALL_FC_T rewrite_tac [SetFuncAll_OKc_lemma]);
val CommonValue_OKc_lemma = save_pop_thm"CommonValue_OKc_lemma";
```

7.15 *ExistsTuples*

SML
```
set_goal([], ⌜∀c tc• tc ∈ OK_TC_d c ∧ tc ∈ OK_TC_c c ⇒ ExistsTuples c tc ∈ OK_VC_c c⌝);
a(rewrite_tac(
        map get_spec[⌜OK_TC_d⌝, ⌜OK_TC_c⌝, ⌜OK_VC_c⌝, ⌜ExistsTuples⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(cases_tac⌜¬ HideDerTable c (Snd (tc tl_0)) = HideDerTable c (Snd (tc tl_1))⌝);
(* *** Goal "1" *** *)
a(all_asm_fc_tac[]);
a(DROP_ASM_T ⌜¬ c dominates Fst (tc tl_0)⌝ ante_tac THEN asm_rewrite_tac[]);
a(strip_tac THEN asm_rewrite_tac[]);
```

SML
```
(* *** Goal "2" *** *)
a(all_asm_fc_tac[]);
a(cases_tac⌜c dominates Fst (tc tl_1)⌝ THEN asm_rewrite_tac[]);
a(DROP_ASM_T ⌜HideDerTable c (Snd (tc tl_0)) = HideDerTable c (Snd (tc tl_1))⌝
        ante_tac);
a(rewrite_tac(MkDerTable_lemma::
        pc_rule1 "sets_ext1" prove_rule[]
        ⌜{r|c dominates DTR_row r ∧ c dominates DTR_where r} =
        {r|c dominates DTR_row r} ∩ {r|c dominates DTR_where r}⌝::
        ⌈_∩_lemma::
        map get_spec[⌜HideDerTable⌝, ⌜HideDerTableData⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(all_fc_tac[map_hide_map_hide_⌈_lemma]);
a(lemma_tac⌜
        Map DTR_row(Map(HideDerTableRow c)
                ((DT_rows (Snd (tc tl_0)) ⌈ {r|c dominates DTR_row r})
                        ⌈ {r|c dominates DTR_where r})) =
        Map DTR_row(Map(HideDerTableRow c)
                ((DT_rows (Snd (tc tl_1)) ⌈ {r|c dominates DTR_row r})
                        ⌈ {r|c dominates DTR_where r}))⌝
        THEN1 asm_rewrite_tac[]);
a(POP_ASM_T ante_tac THEN
        rewrite_tac[map_o_lemma, DTR_row_o_HideDerTableRow_lemma]);
a(STRIP_T rewrite_thm_tac);
val ExistsTuples_OK_c_lemma = save_pop_thm"ExistsTuples_OK_c_lemma";
```

7.16 *SingleValue*

SML

```
set_goal([], ⌜∀c tc• tc ∈ OK_TC_d c ∧ tc ∈ OK_TC_c c ⇒ SingleValue c tc ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_TC_d⌝, ⌜OK_TC_c⌝, ⌜OK_VC_c⌝, ⌜SingleValue⌝, ⌜Let⌝])
        THEN REPEAT strip_tac);
a(all_asm_fc_tac[] THEN asm_rewrite_tac[∀_elim⌜Fst⌝fun_if_thm]);
a(cases_tac⌜c dominates Fst (tc tl_1)⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜c dominates Fst (tc tl_0)⌝ THEN1 asm_rewrite_tac[]);
a(cases_tac⌜¬HideDerTable c (Snd (tc tl_0)) = HideDerTable c (Snd (tc tl_1))⌝
        THEN1 all_asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac(MkDerTable_lemma::
        pc_rule1 "sets_ext1" prove_rule[]
        ⌜{r|c dominates DTR_row r ∧ c dominates DTR_where r} =
        {r|c dominates DTR_row r} ∩ {r|c dominates DTR_where r}⌝::
        map get_spec[⌜HideDerTable⌝, ⌜HideDerTableData⌝, ⌜Let⌝]));
a(rewrite_tac[↑_∩_lemma] THEN strip_tac);
a(all_fc_tac[map_hide_map_hide_↑_lemma]);
```

SML

```
a(LEMMA_T⌜
        #(Map (HideDerTableRow c)
                ((DT_rows (Snd (tc tl_0)) ↑ {r|c dominates DTR_row r})
                        ↑ {r|c dominates DTR_where r})) =
        #(Map (HideDerTableRow c)
                ((DT_rows (Snd (tc tl_1)) ↑ {r|c dominates DTR_row r})
                        ↑ {r|c dominates DTR_where r}))⌝
        (strip_asm_tac o rewrite_rule[length_map_thm])
        THEN1 asm_rewrite_tac[]);
a(cases_tac⌜
        #((DT_rows (Snd (tc tl_1)) ↑ {r|c dominates DTR_row r})
                        ↑ {r|c dominates DTR_where r}) = 1⌝
        THEN asm_rewrite_tac[]);
a(lemma_tac⌜
        #((DT_rows (Snd (tc tl_0)) ↑ {r|c dominates DTR_row r})
                        ↑ {r|c dominates DTR_where r}) = 1⌝
        THEN1 asm_rewrite_tac[]);
a(LIST_DROP_NTH_ASM_T[1,2]
        (MAP_EVERY(strip_asm_tac o rewrite_rule[length_1_thm])));
a(asm_rewrite_tac[]);
a(lemma_tac⌜HideDerTableRow c x = HideDerTableRow c x'⌝);
```

SML

```
(* *** Goal "1" *** *)
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
(* *** Goal "2" *** *)
a(POP_ASM_T ante_tac THEN
        rewrite_tac[MkDerTableRow_lemma, let_def, get_spec⌜HideDerTableRow⌝]);
a(strip_tac);
a(LEMMA_T⌜
        #(Map (λ (c', i)• if c dominates c'
                then (c', i)
                else (c', ValuedItemItem (MkValuedItem sterling dummyVal)))
              (DTR_cols x)) =
        #(Map (λ (c', i)• if c dominates c'
                then (c', i)
                else (c', ValuedItemItem (MkValuedItem sterling dummyVal)))
              (DTR_cols x'))⌝
        (strip_asm_tac o rewrite_rule[length_map_thm])
        THEN1 asm_rewrite_tac[]);
```

SML

```
a(asm_rewrite_tac[]);
a(cases_tac⌜# (DTR_cols x') = 1⌝ THEN asm_rewrite_tac[]);
a(lemma_tac⌜# (DTR_cols x) = 1⌝ THEN1 asm_rewrite_tac[]);
a(LIST_DROP_NTH_ASM_T[1,2]
        (MAP_EVERY(strip_asm_tac o rewrite_rule[length_1_thm])));
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[get_spec⌜Map⌝]);
a(cases_tac⌜c dominates Fst x''⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2.1" *** *)
a(cases_tac⌜c dominates Fst x'''⌝ THEN asm_rewrite_tac[]
        THEN REPEAT strip_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(cases_tac⌜c dominates Fst x'''⌝ THEN asm_rewrite_tac[]);
a(PC_T1 "prop_eq_pair" prove_tac[]);
val SingleValue_OK_c_lemma = save_pop_thm"SingleValue_OK_c_lemma";
```

7.17 *JoinedRowExistence*

SML

```
set_goal([], ⌜∀c i• JoinedRowExistence i ∈ OK_VC_c c⌝);
a(rewrite_tac(map get_spec[⌜OK_VC_c⌝, ⌜JoinedRowExistence⌝])
        THEN REPEAT strip_tac);
val JoinedRowExistence_OK_c_lemma = save_pop_thm"JoinedRowExistence_OK_c_lemma";
```

8 CLOSING DOWN

9 THE THEORY fef033

9.1 Parents

fef031 fef032

9.2 Children

fef035

9.3 Theorems

TableComputations_consistent
ValueComputations_consistent

$$
\begin{array}{l}
\vdash \mathit{Consistent} \\
\quad (\lambda\ (\mathit{TableComputations}',\ \mathit{ValueComputations}') \\
\quad\quad \bullet\ \forall\ \mathit{cc} \\
\quad\quad\quad \bullet\ (\mathit{TableComputations}'\ \mathit{cc}, \\
\quad\quad\quad\quad\quad \mathit{ValueComputations}'\ \mathit{cc}) \\
\quad\quad\quad\quad = \bigcap_2 \\
\quad\quad\quad\quad\quad \{(\mathit{tes},\ \mathit{es}) \\
\quad\quad\quad\quad\quad\quad |(\forall\ \mathit{ci} \bullet\ \mathit{DenoteConstant}\ \mathit{ci} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{i} \bullet\ \mathit{Contents}\ \mathit{i} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{i} \bullet\ \mathit{Classification}\ \mathit{i} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{CountAll} \in \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{f}\ \mathit{e} \bullet\ \mathit{e} \in \mathit{es} \Rightarrow \mathit{MonOp}\ \mathit{f}\ \mathit{e} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{f}\ \mathit{e1}\ \mathit{e2} \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ \mathit{e1} \in \mathit{es} \wedge \mathit{e2} \in \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \mathit{BinOp}\ \mathit{f}\ \mathit{e1}\ \mathit{e2} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{f}\ \mathit{e1}\ \mathit{e2}\ \mathit{e3} \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ \mathit{e1} \in \mathit{es} \wedge \mathit{e2} \in \mathit{es} \wedge \mathit{e3} \in \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \mathit{TriOp}\ \mathit{f}\ \mathit{e1}\ \mathit{e2}\ \mathit{e3} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{el} \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ \mathit{Elems}\ \mathit{el} \subseteq \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \mathit{BinOpAnd}\ \mathit{cc}\ \mathit{el} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{el} \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ \mathit{Elems}\ \mathit{el} \subseteq \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \mathit{BinOpOr}\ \mathit{cc}\ \mathit{el} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{te}\ \mathit{cel}\ \mathit{ee} \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ \mathit{te} \in \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Fst}\ \mathit{cel}) \subseteq \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Snd}\ \mathit{cel}) \subseteq \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{ee} \in \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow \mathit{CaseVal}\ \mathit{cc}\ \mathit{te}\ \mathit{cel}\ \mathit{ee} \in \mathit{es}) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ \mathit{cel}\ \mathit{ee} \\
\quad\quad\quad\quad\quad\quad\quad \bullet\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Fst}\ \mathit{cel}) \subseteq \mathit{es} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Snd}\ \mathit{cel}) \subseteq \mathit{es}
\end{array}
$$

$$
\begin{array}{l}
\quad \wedge\ ee \in es \\
\Rightarrow\ Case\ cc\ cel\ ee \in es) \\
\wedge\ (\forall\ e \\
\bullet\ e \in es \Rightarrow SetFuncAllAnd\ cc\ e \in es) \\
\wedge\ (\forall\ e \\
\bullet\ e \in es \Rightarrow SetFuncAllOr\ cc\ e \in es) \\
\wedge\ (\forall\ e \\
\bullet\ e \in es \Rightarrow CountNonNull\ e \in es) \\
\wedge\ (\forall\ e \\
\bullet\ e \in es \Rightarrow CountDistinct\ e \in es) \\
\wedge\ (\forall\ e \\
\bullet\ e \in es \Rightarrow CommonValue\ e \in es) \\
\wedge\ (\forall\ f\ e \\
\bullet\ e \in es \Rightarrow SetFuncAll\ f\ e \in es) \\
\wedge\ (\forall\ f\ e \\
\bullet\ e \in es \\
\quad \Rightarrow SetFuncDistinct\ f\ e \in es) \\
\wedge\ (\forall\ te \\
\bullet\ te \in tes \\
\quad \Rightarrow ExistsTuples\ cc\ te \in es) \\
\wedge\ (\forall\ te \\
\bullet\ te \in tes \\
\quad \Rightarrow SingleValue\ cc\ te \in es) \\
\wedge\ JoinedRowExistence\ cc \in es \\
\wedge\ (\forall\ i\bullet\ TableContents\ i \in tes) \\
\wedge\ (\forall\ esl\ tel\ e1\ ml\ nl\ e2 \\
\bullet\ Elems\ (Map\ Fst\ esl) \subseteq es \\
\quad \wedge\ Elems\ tel \subseteq tes \\
\quad \wedge\ e1 \in es \\
\quad \wedge\ e2 \in es \\
\quad \Rightarrow AllTuples \\
\quad\quad cc \\
\quad\quad esl \\
\quad\quad tel \\
\quad\quad e1 \\
\quad\quad ml \\
\quad\quad nl \\
\quad\quad e2 \\
\quad \in tes)\})
\end{array}
$$

BoolItem_OneOne_lemma

$$\vdash\ \forall\ i1\ i2\bullet\ BoolItem\ i1 = BoolItem\ i2 \Leftrightarrow i1 \Leftrightarrow i2$$

HideDerTableRow_Length_lemma

$$
\begin{array}{l}
\vdash\ \forall\ c\ r1\ r2 \\
\quad \bullet\ HideDerTableRow\ c\ r1 = HideDerTableRow\ c\ r2 \\
\quad\quad \Rightarrow \#\ (DTR\_cols\ r1) = \#\ (DTR\_cols\ r2)
\end{array}
$$

Nth_HideDerTableRow_lemma

$$
\begin{array}{l}
\vdash\ \forall\ c\ r\ i \\
\quad \bullet\ 1 \leq i \wedge i \leq \#\ (DTR\_cols\ r)
\end{array}
$$

$\Rightarrow Nth\ (DTR\_cols\ (HideDerTableRow\ c\ r))\ i$
$= (Fst\ (Nth\ (DTR\_cols\ r)\ i),$
$(if\ c\ dominates\ Fst\ (Nth\ (DTR\_cols\ r)\ i)$
$then\ Snd\ (Nth\ (DTR\_cols\ r)\ i)$
$else$
$ValuedItemItem$
$(MkValuedItem\ sterling\ dummyVal)))$

DTR_row_o_HideDerTableRow_lemma
$\vdash \forall\ c\bullet\ DTR\_row\ o\ HideDerTableRow\ c = DTR\_row$

fun_if_thm $\vdash \forall\ f\ a\ b\ c$
$\bullet\ f\ (if\ a\ then\ b\ else\ c) = (if\ a\ then\ f\ b\ else\ f\ c)$

⌈_null_map_hide_lemma
$\vdash \forall\ c\ rl$
$\bullet\ rl$
$\restriction \{r$
$|c\ dominates\ DTR\_row\ r$
$\wedge\ c\ dominates\ DTR\_where\ r\}$
$= []$
$\Leftrightarrow Map$
$(HideDerTableRow\ c)$
$(rl \restriction \{r|c\ dominates\ DTR\_row\ r\})$
$\restriction \{r|c\ dominates\ DTR\_where\ r\}$
$= []$

map_hide_map_hide_⌈_lemma
$\vdash \forall\ c\ rl1\ rl2$
$\bullet\ Map\ (HideDerTableRow\ c)\ rl1$
$= Map\ (HideDerTableRow\ c)\ rl2$
$\Rightarrow Map$
$(HideDerTableRow\ c)$
$(rl1 \restriction \{r|c\ dominates\ DTR\_where\ r\})$
$= Map$
$(HideDerTableRow\ c)$
$(rl2 \restriction \{r|c\ dominates\ DTR\_where\ r\})$

⌈_∩_lemma $\vdash \forall\ l\ a\ b\bullet\ l \restriction (a \cap b) = l \restriction a \restriction b$

split_thm $\vdash \forall\ xy\ list$
$\bullet\ Split\ [] = ([], [])$
$\wedge\ Split\ (Cons\ xy\ list)$
$= (Cons\ (Fst\ xy)\ (Fst\ (Split\ list)),$
$Cons\ (Snd\ xy)\ (Snd\ (Split\ list)))$

length_0_thm $\vdash \forall\ list\bullet\ \#\ list = 0 \Leftrightarrow list = []$

length_1_thm $\vdash \forall\ list\bullet\ \#\ list = 1 \Leftrightarrow (\exists\ x\bullet\ list = [x])$

fst_snd_split_thm
$\vdash \forall\ list$
$\bullet\ Fst\ (Split\ list) = Map\ Fst\ list$
$\wedge\ Snd\ (Split\ list) = Map\ Snd\ list$

lubl_lemma $\vdash \forall\ c\ cl$
$\bullet\ lubl\ [] = lattice\_bottom$
$\wedge\ lubl\ (Cons\ c\ cl) = c\ lub\ lubl\ cl$

lub_lattice_bottom_thm

$$\vdash \forall\ c\bullet\ c\ lub\ lattice\_bottom = c$$

dominates_lubl_map_hide_lemma

$$\begin{array}{l}\vdash \forall\ cc\ cil\\ \quad \bullet\ cc\ dominates\ lubl\ (Map\ Fst\ cil)\\ \qquad \Rightarrow Map\\ \qquad\quad (\lambda\ (c,\ i)\\ \qquad\qquad \bullet\ (c,\\ \qquad\qquad\quad (if\ cc\ dominates\ c\\ \qquad\qquad\qquad then\ i\\ \qquad\qquad\qquad else\ Arbitrary)))\\ \qquad\quad cil\\ \qquad = cil\end{array}$$

CaseVal_lemma

$$\begin{array}{l}\vdash \forall\ cc\ tst\ cev\ cevs\ ev\\ \quad \bullet\ CaseVal\ cc\ tst\ []\ ev\\ \qquad = (\lambda\ tl\ rl\ r\\ \qquad \bullet\ (let\ (tc,\ ti) = tst\ tl\ rl\ r\\ \qquad\quad in\ let\ (ec,\ ei) = ev\ tl\ rl\ r\\ \qquad\qquad in\ if\ cc\ dominates\ tc\\ \qquad\qquad then\ (ec,\ ei)\\ \qquad\qquad else\ (tc,\ ei)))\\ \qquad \wedge\ CaseVal\ cc\ tst\ (Cons\ cev\ cevs)\ ev\\ \qquad = (\lambda\ tl\ rl\ r\\ \qquad \bullet\ (let\ (cr,\ ir)\\ \qquad\qquad\quad = CaseVal\ cc\ tst\ cevs\ ev\ tl\ rl\ r\\ \qquad\quad in\ let\ (tc,\ ti) = tst\ tl\ rl\ r\\ \qquad\qquad in\ let\ (ce,\ cv) = cev\\ \qquad\qquad\quad in\ let\ (cec,\ cei) = ce\ tl\ rl\ r\\ \qquad\qquad\qquad in\ let\ (cvc,\ cvi) = cv\ tl\ rl\ r\\ \qquad\qquad\qquad in\ if\ ti = cei\\ \qquad\qquad\qquad then\\ \qquad\qquad\qquad\quad if\\ \qquad\qquad\qquad\qquad cc\ dominates\ tc\\ \qquad\qquad\qquad\qquad\quad \wedge\ cc\ dominates\ cec\\ \qquad\qquad\qquad\quad then\ (cvc,\ cvi)\\ \qquad\qquad\qquad\quad else\ if\ \neg\ cc\ dominates\ tc\\ \qquad\qquad\qquad\quad then\ (tc,\ cvi)\\ \qquad\qquad\qquad\quad else\ (cec,\ cvi)\\ \qquad\qquad\qquad else\ if\\ \qquad\qquad\qquad\quad cc\ dominates\ tc\\ \qquad\qquad\qquad\qquad \wedge\ cc\ dominates\ cec\\ \qquad\qquad\qquad then\ (cr,\ ir)\\ \qquad\qquad\qquad else\ if\ \neg\ cc\ dominates\ tc\\ \qquad\qquad\qquad then\ (tc,\ ir)\\ \qquad\qquad\qquad else\ (cec,\ ir)))\end{array}$$

OK_TC$_d$_lemma

$$\vdash \forall\ c\bullet\ OK\_TC_d\ c \subseteq OkTableComputation\ c$$

DenoteConstant_OK$_d$_lemma

$$\vdash \forall\ c\ ci \bullet\ DenoteConstant\ ci \in OK\_VC_d\ c$$

Contents_OK$_d$_lemma

$$\vdash \forall\ c\ i \bullet\ Contents\ i \in OK\_VC_d\ c$$

Classification_OK$_d$_lemma

$$\vdash \forall\ c\ i \bullet\ Classification\ i \in OK\_VC_d\ c$$

CountAll_OK$_d$_lemma

$$\vdash \forall\ c \bullet\ CountAll \in OK\_VC_d\ c$$

MonOp_OK$_d$_lemma

$$\vdash \forall\ c\ f\ vc \bullet\ vc \in OK\_VC_d\ c \Rightarrow MonOp\ f\ vc \in OK\_VC_d\ c$$

ComputeAnd_lemma

$$\begin{array}{l}
\vdash \forall\ cc\ cil1\ cil2 \\
\quad \bullet\ Map \\
\qquad (\lambda\ (c,\ i) \\
\qquad\quad \bullet\ (c, \\
\qquad\qquad (if\ cc\ dominates\ c \\
\qquad\qquad\quad then\ i \\
\qquad\qquad\quad else\ Arbitrary))) \\
\qquad cil1 \\
\qquad = Map \\
\qquad (\lambda\ (c,\ i) \\
\qquad\quad \bullet\ (c, \\
\qquad\qquad (if\ cc\ dominates\ c \\
\qquad\qquad\quad then\ i \\
\qquad\qquad\quad else\ Arbitrary))) \\
\qquad cil2 \\
\quad \Rightarrow Fst\ (ComputeAnd\ cc\ cil1) \\
\qquad = Fst\ (ComputeAnd\ cc\ cil2) \\
\quad \wedge\ (cc\ dominates\ Fst\ (ComputeAnd\ cc\ cil1) \\
\qquad \Rightarrow Snd\ (ComputeAnd\ cc\ cil1) \\
\qquad\quad = Snd\ (ComputeAnd\ cc\ cil2))
\end{array}$$

BinOpAnd_OK$_d$_lemma

$$\begin{array}{l}
\vdash \forall\ c\ vcl \\
\quad \bullet\ Elems\ vcl \subseteq OK_VC_d\ c \wedge Elems\ vcl \subseteq OK_VC_c\ c \\
\qquad \Rightarrow BinOpAnd\ c\ vcl \in OK_VC_d\ c
\end{array}$$

ComputeOr_lemma

$$\begin{array}{l}
\vdash \forall\ cc\ cil1\ cil2 \\
\quad \bullet\ Map \\
\qquad (\lambda\ (c,\ i) \\
\qquad\quad \bullet\ (c, \\
\qquad\qquad (if\ cc\ dominates\ c \\
\qquad\qquad\quad then\ i \\
\qquad\qquad\quad else\ Arbitrary))) \\
\qquad cil1 \\
\qquad = Map \\
\qquad (\lambda\ (c,\ i) \\
\qquad\quad \bullet\ (c, \\
\qquad\qquad (if\ cc\ dominates\ c
\end{array}$$

$$
\begin{array}{l}
\qquad\qquad\qquad\quad \mathit{then}\ i \\
\qquad\qquad\qquad\quad \mathit{else}\ \mathit{Arbitrary}))) \\
\qquad\quad \mathit{cil2} \\
\Rightarrow \mathit{Fst}\ (\mathit{ComputeOr}\ \mathit{cc}\ \mathit{cil1}) \\
\qquad = \mathit{Fst}\ (\mathit{ComputeOr}\ \mathit{cc}\ \mathit{cil2}) \\
\quad \wedge\ (\mathit{cc}\ \mathit{dominates}\ \mathit{Fst}\ (\mathit{ComputeOr}\ \mathit{cc}\ \mathit{cil1}) \\
\qquad \Rightarrow \mathit{Snd}\ (\mathit{ComputeOr}\ \mathit{cc}\ \mathit{cil1}) \\
\qquad\quad = \mathit{Snd}\ (\mathit{ComputeOr}\ \mathit{cc}\ \mathit{cil2}))
\end{array}
$$

BinOpOr_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ \mathit{vcl} \\
\quad \bullet\ \mathit{Elems}\ \mathit{vcl} \subseteq \mathit{OK\_VC}_d\ c \wedge \mathit{Elems}\ \mathit{vcl} \subseteq \mathit{OK\_VC}_c\ c \\
\qquad \Rightarrow \mathit{BinOpOr}\ c\ \mathit{vcl} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

BinOp_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ f\ \mathit{vc1}\ \mathit{vc2} \\
\quad \bullet\ \mathit{vc1} \in \mathit{OK\_VC}_d\ c \wedge \mathit{vc2} \in \mathit{OK\_VC}_d\ c \\
\qquad \Rightarrow \mathit{BinOp}\ f\ \mathit{vc1}\ \mathit{vc2} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

TriOp_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ f\ \mathit{vc1}\ \mathit{vc2}\ \mathit{vc3} \\
\quad \bullet\ \mathit{vc1} \in \mathit{OK\_VC}_d\ c \wedge \mathit{vc2} \in \mathit{OK\_VC}_d\ c \wedge \mathit{vc3} \in \mathit{OK\_VC}_d\ c \\
\qquad \Rightarrow \mathit{TriOp}\ f\ \mathit{vc1}\ \mathit{vc2}\ \mathit{vc3} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

CaseVal_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ \mathit{te}\ \mathit{cel}\ \mathit{ee} \\
\quad \bullet\ \mathit{te} \in \mathit{OK\_VC}_d\ c \\
\qquad\quad \wedge\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Fst}\ \mathit{cel}) \subseteq \mathit{OK\_VC}_d\ c \\
\qquad\quad \wedge\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Snd}\ \mathit{cel}) \subseteq \mathit{OK\_VC}_d\ c \\
\qquad\quad \wedge\ \mathit{ee} \in \mathit{OK\_VC}_d\ c \\
\qquad\quad \Rightarrow \mathit{CaseVal}\ c\ \mathit{te}\ \mathit{cel}\ \mathit{ee} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

Case_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ \mathit{cel}\ \mathit{ee} \\
\quad \bullet\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Fst}\ \mathit{cel}) \subseteq \mathit{OK\_VC}_d\ c \\
\qquad\quad \wedge\ \mathit{Elems}\ (\mathit{Map}\ \mathit{Snd}\ \mathit{cel}) \subseteq \mathit{OK\_VC}_d\ c \\
\qquad\quad \wedge\ \mathit{ee} \in \mathit{OK\_VC}_d\ c \\
\qquad\quad \Rightarrow \mathit{Case}\ c\ \mathit{cel}\ \mathit{ee} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

SetFuncAll_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ f\ \mathit{vc} \\
\quad \bullet\ \mathit{vc} \in \mathit{OK\_VC}_d\ c \Rightarrow \mathit{SetFuncAll}\ f\ \mathit{vc} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

SetFuncDistinct_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ f\ \mathit{vc} \\
\quad \bullet\ \mathit{vc} \in \mathit{OK\_VC}_d\ c \Rightarrow \mathit{SetFuncDistinct}\ f\ \mathit{vc} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

SetFuncAllAnd_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ \mathit{vc} \\
\quad \bullet\ \mathit{vc} \in \mathit{OK\_VC}_d\ c \wedge \mathit{vc} \in \mathit{OK\_VC}_c\ c \\
\qquad \Rightarrow \mathit{SetFuncAllAnd}\ c\ \mathit{vc} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

SetFuncAllOr_OK$_d$_lemma

$$
\begin{array}{l}
\vdash \forall\ c\ \mathit{vc} \\
\quad \bullet\ \mathit{vc} \in \mathit{OK\_VC}_d\ c \wedge \mathit{vc} \in \mathit{OK\_VC}_c\ c \\
\qquad \Rightarrow \mathit{SetFuncAllOr}\ c\ \mathit{vc} \in \mathit{OK\_VC}_d\ c
\end{array}
$$

CountNonNull_OK$_d$_lemma

$$\vdash \forall\ c\ vc \bullet\ vc \in OK\_VC_d\ c \Rightarrow CountNonNull\ vc \in OK\_VC_d\ c$$

CountDistinct_OK$_d$_lemma

$$\vdash \forall\ c\ vc \bullet\ vc \in OK\_VC_d\ c \Rightarrow CountDistinct\ vc \in OK\_VC_d\ c$$

CommonValue_OK$_d$_lemma

$$\vdash \forall\ c\ vc \bullet\ vc \in OK\_VC_d\ c \Rightarrow CommonValue\ vc \in OK\_VC_d\ c$$

ExistsTuples_OK$_d$_lemma

$$\vdash \forall\ c\ tc \bullet\ tc \in OK\_TC_d\ c \wedge tc \in OK\_TC_c\ c \Rightarrow ExistsTuples\ c\ tc \in OK\_VC_d\ c$$

SingleValue_OK$_d$_lemma

$$\vdash \forall\ c\ tc \bullet\ tc \in OK\_TC_d\ c \wedge tc \in OK\_TC_c\ c \Rightarrow SingleValue\ c\ tc \in OK\_VC_d\ c$$

JoinedRowExistence_OK$_d$_lemma

$$\vdash \forall\ c\ i \bullet\ JoinedRowExistence\ i \in OK\_VC_d\ c$$

DenoteConstant_OK$_c$_lemma

$$\vdash \forall\ c\ ci \bullet\ DenoteConstant\ ci \in OK\_VC_c\ c$$

Contents_OK$_c$_lemma

$$\vdash \forall\ c\ i \bullet\ Contents\ i \in OK\_VC_c\ c$$

Classification_OK$_c$_lemma

$$\vdash \forall\ c\ i \bullet\ Classification\ i \in OK\_VC_c\ c$$

CountAll_OK$_c$_lemma

$$\vdash \forall\ c \bullet\ CountAll \in OK\_VC_c\ c$$

MonOp_OK$_c$_lemma

$$\vdash \forall\ c\ f\ vc \bullet\ vc \in OK\_VC_c\ c \Rightarrow MonOp\ f\ vc \in OK\_VC_c\ c$$

BinOpAnd_OK$_c$_lemma

$$\vdash \forall\ c\ vcl \bullet\ Elems\ vcl \subseteq OK\_VC_d\ c \wedge Elems\ vcl \subseteq OK\_VC_c\ c \Rightarrow BinOpAnd\ c\ vcl \in OK\_VC_c\ c$$

BinOpOr_OK$_c$_lemma

$$\vdash \forall\ c\ vcl \bullet\ Elems\ vcl \subseteq OK\_VC_d\ c \wedge Elems\ vcl \subseteq OK\_VC_c\ c \Rightarrow BinOpOr\ c\ vcl \in OK\_VC_c\ c$$

BinOp_OK$_c$_lemma

$$\vdash \forall\ c\ f\ vc1\ vc2 \bullet\ vc1 \in OK\_VC_c\ c \wedge vc2 \in OK\_VC_c\ c \Rightarrow BinOp\ f\ vc1\ vc2 \in OK\_VC_c\ c$$

TriOp_OK$_c$_lemma

$$\vdash \forall\ c\ f\ vc1\ vc2\ vc3 \bullet\ vc1 \in OK\_VC_c\ c \wedge vc2 \in OK\_VC_c\ c \wedge vc3 \in OK\_VC_c\ c \Rightarrow TriOp\ f\ vc1\ vc2\ vc3 \in OK\_VC_c\ c$$

CaseVal_OK$_c$_lemma

$$\begin{aligned}\vdash \forall\ & c\ te\ cel\ ee \bullet\ te \in OK\_VC_d\ c \\ & \wedge\ te \in OK\_VC_c\ c \\ & \wedge\ Elems\ (Map\ Fst\ cel) \subseteq OK\_VC_d\ c \\ & \wedge\ Elems\ (Map\ Fst\ cel) \subseteq OK\_VC_c\ c \\ & \wedge\ Elems\ (Map\ Snd\ cel) \subseteq OK\_VC_c\ c\end{aligned}$$

$\wedge\ ee \in OK\_VC_c\ c$
$\Rightarrow\ CaseVal\ c\ te\ cel\ ee \in OK\_VC_c\ c$

Case_OK$_c$_lemma
$\vdash \forall\ c\ cel\ ee$
• $Elems\ (Map\ Fst\ cel) \subseteq OK\_VC_d\ c$
$\wedge\ Elems\ (Map\ Fst\ cel) \subseteq OK\_VC_c\ c$
$\wedge\ Elems\ (Map\ Snd\ cel) \subseteq OK\_VC_c\ c$
$\wedge\ ee \in OK\_VC_c\ c$
$\Rightarrow\ Case\ c\ cel\ ee \in OK\_VC_c\ c$

SetFuncAll_OK$_c$_lemma
$\vdash \forall\ c\ f\ vc$
• $vc \in OK\_VC_c\ c \Rightarrow SetFuncAll\ f\ vc \in OK\_VC_c\ c$

SetFuncDistinct_OK$_c$_lemma
$\vdash \forall\ c\ f\ vc$
• $vc \in OK\_VC_c\ c \Rightarrow SetFuncDistinct\ f\ vc \in OK\_VC_c\ c$

SetFuncAllAnd_OK$_c$_lemma
$\vdash \forall\ c\ vc$
• $vc \in OK\_VC_d\ c \wedge vc \in OK\_VC_c\ c$
$\Rightarrow SetFuncAllAnd\ c\ vc \in OK\_VC_c\ c$

SetFuncAllOr_OK$_c$_lemma
$\vdash \forall\ c\ vc$
• $vc \in OK\_VC_d\ c \wedge vc \in OK\_VC_c\ c$
$\Rightarrow SetFuncAllOr\ c\ vc \in OK\_VC_c\ c$

CountNonNull_OK$_c$_lemma
$\vdash \forall\ c\ vc$• $vc \in OK\_VC_c\ c \Rightarrow CountNonNull\ vc \in OK\_VC_c\ c$

CountDistinct_OK$_c$_lemma
$\vdash \forall\ c\ vc$• $vc \in OK\_VC_c\ c \Rightarrow CountDistinct\ vc \in OK\_VC_c\ c$

CommonValue_OK$_c$_lemma
$\vdash \forall\ c\ vc$• $vc \in OK\_VC_c\ c \Rightarrow CommonValue\ vc \in OK\_VC_c\ c$

ExistsTuples_OK$_c$_lemma
$\vdash \forall\ c\ tc$
• $tc \in OK\_TC_d\ c \wedge tc \in OK\_TC_c\ c$
$\Rightarrow ExistsTuples\ c\ tc \in OK\_VC_c\ c$

SingleValue_OK$_c$_lemma
$\vdash \forall\ c\ tc$
• $tc \in OK\_TC_d\ c \wedge tc \in OK\_TC_c\ c$
$\Rightarrow SingleValue\ c\ tc \in OK\_VC_c\ c$

JoinedRowExistence_OK$_c$_lemma
$\vdash \forall\ c\ i$• $JoinedRowExistence\ i \in OK\_VC_c\ c$

10 INDEX