Lemma 1 Ltd.
2nd Floor
31A Chain St.
Reading
Berks
RG1 2HX

Theorems on Finiteness (Draft)

Abstract

This paper supplies proofs of some theorems about finite sets. It is part of work in progress on ProofPower-HOL.

Version:	2.8
Date:	29th September 1992; This revision 6 August 2004
Reference:	PPDOC/WRK044
Pages:	25

Prepared by:	Rob Arthan
Tel:	+44 118 958 4409
E-Mail:	rda@lemma-one.com

CONTENTS

1 INTRODUCTION

This document contains some proofs which are mainly concerned with finiteness as defined in the theory *fin_set* supplied as part of the ProofPower-HOL system. Some miscellaneous material about certain other concepts defined in the ancestors of *fin_set* is also provided.

The material displays several aspects of the use of ProofPower-HOL., including:

1. Proof of an induction theorem and its use to produce an induction tactic.
2. Proof of a theorem for use as a rewrite rule to "compute" values of a function in certain useful special cases (namely the minimum function *Min* applied to set displays whose elements are numeric constants).

Note that the proofs are typically packaged in the following style:

Example

```
val blah_blah_thm = save_thm("fin_set_induction_thm", (
push_goal([], (* Statement of theorem *));
(* Rest of subgoal package command script goes here *)
pop_thm()
));
```

To experiment with such a proof interactively simply omit the first line. I.e. cut-and-paste the *push_goal* command into the ProofPower-HOL window followed by the subgoal package commands (which are almost invariably just tactic applications of the form $a(\ldots)$) modifying these as you wish. To finish the proof off either cut-and-paste the *pop_thm* line followed by a semi-colon (if you have completed the proof) or abandon the proof attempt with *drop_main_goal*.

2 PREAMBLE

SML

```
open_theory"fin_set";
new_theory"fin_thms";
set_pc"hol";
```

3 THE THEOREMS

3.1 Induction over Finite Sets

SML

```
val fin_set_induction_thm = save_thm("fin_set_induction_thm", (
push_goal([], ⌜
                ∀P•    P {} ∧ (∀a x•a ∈ Finite ∧ P a ∧ ¬x ∈ a ⇒ P({x} ∪ a))
                ⇒      ∀a•a ∈ Finite ⇒ P a
⌝);
a(REPEAT strip_tac);
a(asm_ante_tac⌜a ∈ Finite⌝);
a(rewrite_tac[get_spec⌜Finite⌝]);
a(REPEAT strip_tac);
a(spec_nth_asm_tac 1 ⌜{b | b ∈ Finite ∧ P b}⌝);
(* *** Goal "1" *** *)
a(swap_asm_concl_tac ⌜¬ {} ∈ Finite⌝);
a(rewrite_tac[get_spec⌜Finite⌝]);
a(PC_T "hol1" (REPEAT strip_tac));
(* *** Goal "2" *** *)
a(swap_asm_concl_tac ⌜a' ∈ Finite⌝);
a(asm_ante_tac ⌜¬ {x} ∪ a' ∈ Finite⌝);
a(rewrite_tac[get_spec⌜Finite⌝]);
a(PC_T "hol1" (contr_tac));
a(spec_nth_asm_tac 1 ⌜s⌝);
(* *** Goal "2.1" *** *)
a(list_spec_nth_asm_tac 5 [⌜a''⌝, ⌜x'⌝]);
(* *** Goal "2.2" *** *)
a(list_spec_nth_asm_tac 4 [⌜a'⌝, ⌜x⌝]);
(* *** Goal "3" *** *)
a(swap_asm_concl_tac ⌜¬ P ({x} ∪ a')⌝);
a(cases_tac⌜x ∈ a'⌝);
(* *** Goal "3.1" *** *)
a(LEMMA_T⌜{x} ∪ a' = a'⌝ asm_rewrite_thm_tac);
a(PC_T "hol1" (REPEAT strip_tac));
a(asm_rewrite_tac[]);
(* *** Goal "3.2" *** *)
a(list_spec_nth_asm_tac 6 [⌜a'⌝, ⌜x⌝]);
pop_thm()
));
```

SML
```
val fin_set_induction_tac : TACTIC =
                gen_induction_tac1 fin_set_induction_thm;
```

3.2 Theorems about *Min*

SML
```
val min_thm = save_thm("min_thm", (
push_goal([], ⌜∀m a•m ∈ a ⇒ Min a ∈ a ∧ Min a ≤ m⌝);
a(REPEAT ∀_tac THEN ⇒_tac);
a(strip_asm_tac (rewrite_rule[](∀_elim⌜(λj•j ∈ a)⌝≤_well_order_thm))
        THEN asm_prove_tac[]);
(* *** Goal "1" *** *)
a(strip_asm_tac (list_∀_elim[⌜m'⌝, ⌜a⌝](get_spec⌜Min⌝)));
(* *** Goal "1.1" *** *)
a(asm_prove_tac[]);
(* *** Goal "1.2" *** *)
a(asm_rewrite_tac[] THEN REPEAT strip_tac);
(* *** Goal "2" *** *)
a(strip_asm_tac (list_∀_elim[⌜m'⌝, ⌜a⌝](get_spec⌜Min⌝)));
(* *** Goal "2.1" *** *)
a(asm_prove_tac[]);
(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[] THEN asm_prove_tac[]);
pop_thm()
));
```

3.3 Theorems about *Min*

SML
```
val min_clause1 = (
push_goal([], ⌜∀m•Min {m} = m⌝);
a(REPEAT strip_tac);
a(strip_asm_tac (list_∀_elim[⌜m⌝, ⌜{m}⌝](get_spec⌜Min⌝)));
a(swap_asm_concl_tac⌜¬m ≤ i⌝ THEN asm_rewrite_tac[≤_clauses]);
pop_thm()
);
```

SML

```
val min_clause2 = (
push_goal([], ⌜∀m n•Min {m; n} = if m ≤ n then m else n⌝);
a(REPEAT strip_tac);
a(strip_asm_tac (list_∀_elim[⌜if m ≤ n then m else n⌝, ⌜{m; n}⌝](get_spec⌜Min⌝)));
(* *** Goal "1" *** *)
a(all_asm_ante_tac THEN cases_tac⌜m ≤ n⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(swap_asm_concl_tac⌜¬(if m ≤ n then m else n) ≤ i⌝ THEN asm_rewrite_tac[]);
a(cases_tac⌜m ≤ n⌝ THEN asm_rewrite_tac[≤_clauses]);
a(strip_asm_tac(list_∀_elim[⌜m⌝, ⌜n⌝]≤_cases_thm));
(* *** Goal "3" *** *)
a(swap_asm_concl_tac⌜¬(if m ≤ n then m else n) ≤ i⌝ THEN asm_rewrite_tac[]);
a(cases_tac⌜m ≤ n⌝ THEN asm_rewrite_tac[≤_clauses]);
pop_thm()
);
```

SML

```
val min_clause3 = (
push_goal([], ⌜∀m a•Min ({m} ∪ a) = if a = {} then m else Min{m; Min a}⌝);
a(REPEAT strip_tac);
a(cases_tac⌜a = {}⌝ THEN asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(rewrite_tac[min_clause1]);
(* *** Goal "2" *** *)
a(all_asm_ante_tac THEN (PC_T"sets_ext" strip_tac));
a(all_asm_ante_tac THEN strip_tac);
a(strip_asm_tac (list_∀_elim[⌜x⌝, ⌜a⌝]min_thm));
a(lemma_tac⌜∀ i• i ∈ {m} ∪ a ⇒ Min {m; Min a} ≤ i⌝);
(* *** Goal "2.1" *** *)
a(REPEAT strip_tac THEN rewrite_tac[min_clause2]);
(* *** Goal "2.1.1" *** *)
a(cases_tac⌜m ≤ Min a⌝ THEN asm_rewrite_tac[≤_clauses]);
a(strip_asm_tac (list_∀_elim[⌜m⌝, ⌜Min a⌝]≤_cases_thm));
(* *** Goal "2.1.2" *** *)
a(strip_asm_tac (list_∀_elim[⌜i⌝, ⌜a⌝]min_thm));
a(cases_tac ⌜m ≤ Min a⌝ THEN asm_rewrite_tac[]);
a(fc_tac [≤_trans_thm] THEN asm_fc_tac[]);
(* *** Goal "2.2" *** *)
a(LEMMA_T ⌜Min {m; Min a} ∈ {m} ∪ a⌝ asm_tac);
(* *** Goal "2.2.1" *** *)
a(cases_tac⌜m ≤ Min a⌝ THEN asm_rewrite_tac[min_clause2] THEN REPEAT strip_tac);
(* *** Goal "2.2.2" *** *)
```

```
a(ante_tac (list_∀_elim[⌜Min {m; Min a}⌝, ⌜({m} ∪ a)⌝](get_spec⌜Min⌝))
        THEN asm_rewrite_tac[]);
pop_thm()
);
```

SML

```
val min_clauses = save_thm("min_clauses",
        list_∧_intro[min_clause1, min_clause2, min_clause3]);
```

3.4 Relationship between *Finite* and *Elems*

SML

```
val fin_set_thm1 = save_thm("fin_set_thm1", (
push_goal([], ⌜∀a• a ∈ Finite ⇒ ∃list•a = Elems list ∧ list ∈ Distinct⌝);
a(strip_tac);
a fin_set_induction_tac;
(* *** Goal "1" *** *)
a(∃_tac ⌜[]⌝ THEN rewrite_tac(map get_spec[⌜Elems⌝, ⌜Distinct⌝]));
(* *** Goal "2" *** *)
a(∃_tac ⌜Cons x list⌝ THEN asm_rewrite_tac(map get_spec[⌜Elems⌝, ⌜Distinct⌝]));
a(asm_ante_tac ⌜¬ x ∈ a⌝ THEN asm_rewrite_tac[]);
pop_thm()
));
```

SML

```
val elems_thm1 = save_thm("elems_thm1", (
push_goal([], ⌜∀list• Elems list = {} ⇔ list = []⌝);
a(strip_tac THEN strip_asm_tac(∀_elim⌜list:'a LIST⌝ list_cases_thm));
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Elems⌝]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[get_spec⌜Elems⌝]);
a(PC_T "sets_ext" (REPEAT strip_tac));
a(REPEAT strip_tac);
a(∃_tac⌜x⌝ THEN REPEAT strip_tac);
pop_thm()
));
```

SML

```
val elems_thm2 = save_thm("elems_thm2", (
push_goal([], ⌜(∀list• Elems list = {} ⇔ list = [])
        ∧ (∀list• {} = Elems list ⇔ list = [])⌝);
a(rewrite_tac[elems_thm1]);
a(rewrite_tac[conv_rule (ONCE_MAP_C eq_sym_conv) elems_thm1]);
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
pop_thm()
));
```

SML

```
val elems_thm3 = save_thm("elems_thm3", (
push_goal([], ⌜∀list1 list2• Elems(list1 @ list2) = Elems list1 ∪ Elems list2⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜list1⌝);
(* *** Goal "1" *** *)
a(rewrite_tac(map get_spec[⌜Append⌝, ⌜Elems⌝]));
(* *** Goal "2" *** *)
a(asm_rewrite_tac(map get_spec[⌜Append⌝, ⌜Elems⌝]));
a(PC_T"hol1" (REPEAT strip_tac));
pop_thm()
));
```

SML

```
val length_thm = save_thm("length_thm", (
push_goal([], ⌜∀list1 list2• Length (list1 @ list2) = Length list1 + Length list2⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜list1⌝ THEN rewrite_tac(map get_spec[⌜Length⌝, ⌜Append⌝]));
a(asm_rewrite_tac[plus_assoc_thm]);
pop_thm()
));
```

SML

```
val ↾_thm1 = save_thm("↾_thm1", (
push_goal([], ⌜∀list a•Elems(list ↾ a) = Elems list ∩ a⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜list⌝ THEN rewrite_tac(map get_spec[⌜Elems⌝, ⌜$↾⌝]));
a(strip_tac THEN cases_tac ⌜x ∈ a⌝ THEN asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Elems⌝]);
a(PC_T "hol1" (REPEAT strip_tac));
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
```

```
a(PC_T "hol1" (REPEAT strip_tac));
a(asm_ante_tac⌜x' ∈ a⌝ THEN asm_rewrite_tac[]);
pop_thm()
));
```

SML

```
val ↑_thm2 = save_thm("↑_thm2", (
push_goal([], ⌜∀list; a:'a SET•Length((list ↑ a) @ (list ↑ ∼a)) = Length list⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜list⌝ THEN rewrite_tac(map get_spec[⌜Length⌝, ⌜$↑⌝, ⌜Append⌝]));
a(strip_tac THEN cases_tac ⌜x ∈ a⌝ THEN asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(asm_rewrite_tac(map get_spec[⌜Length⌝, ⌜$↑⌝, ⌜Append⌝]));
(* *** Goal "2" *** *)
a(all_asm_ante_tac THEN rewrite_tac[length_thm] THEN REPEAT strip_tac);
a(asm_rewrite_tac[get_spec⌜Length⌝, ∀_elim⌜1⌝plus_order_thm]);
pop_thm()
));
```

SML

```
val ↑_thm3 = save_thm("↑_thm3", (
push_goal([], ⌜∀list a•Elems list ⊆ a ⇒ list ↑ a = list ⌝);
a(strip_tac);
a(list_induction_tac⌜list⌝);
(* *** Goal "1" *** *)
a(rewrite_tac[get_spec⌜$↑⌝]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac(map get_spec[⌜$↑⌝, ⌜Elems⌝]));
a(REPEAT strip_tac);
a(lemma_tac ⌜x ∈ a⌝);
(* *** Goal "2.1" *** *)
a(POP_ASM_T ante_tac);
a(PC_T"hol1"(REPEAT strip_tac));
a(spec_nth_asm_tac 1 ⌜x⌝);
(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[]);
a(lemma_tac ⌜Elems list ⊆ a⌝);
(* *** Goal "2.2.1" *** *)
a(asm_ante_tac⌜{x} ∪ Elems list ⊆ a⌝ THEN PC_T"hol1"(REPEAT strip_tac));
a(spec_nth_asm_tac 2 ⌜x'⌝);
(* *** Goal "2.2.2" *** *)
a(spec_nth_asm_tac 4 ⌜a⌝);
pop_thm()
```

```
));
```

SML

```
val ↾_thm4 = save_thm("↾_thm4", (
push_goal([], ⌜∀list; x:'a•x ∈ Elems list ⇒ Length(list ↾ ∼{x}) < Length list ⌝);
a(strip_tac);
a(list_induction_tac⌜list⌝ THEN asm_rewrite_tac[get_spec⌜Elems⌝]);
(* *** Goal "" *** *)
a(asm_rewrite_tac(map get_spec[⌜$↾⌝, ⌜Length⌝]) THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(asm_rewrite_tac[get_spec⌜Length⌝] THEN cases_tac⌜x ∈ Elems list⌝);
(* *** Goal "1.1" *** *)
a(spec_nth_asm_tac 3 ⌜x⌝);
a(strip_asm_tac(list_∀_elim
        [⌜Length (list ↾ ∼ {x})⌝, ⌜Length list⌝, ⌜Length list + 1⌝]
        less_trans_thm));
a(lemma_tac⌜Elems list ⊆ ∼{x}⌝);
(* *** Goal "1.2.1" *** *)
a(PC_T"hol1"(REPEAT strip_tac THEN rewrite_tac[]));
a(swap_asm_concl_tac ⌜x'' ∈ Elems list⌝ THEN asm_rewrite_tac[]);
(* *** Goal "1.2.2" *** *)
a(strip_asm_tac(list_∀_elim[⌜list⌝, ⌜∼{x}⌝]↾_thm3));
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 2 (strip_asm_tac o ∀_elim ⌜x'⌝));
a(cases_tac⌜x = x'⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2.1" *** *)
a(strip_asm_tac(list_∀_elim
        [⌜Length (list ↾ ∼ {x'})⌝, ⌜Length list⌝, ⌜Length list + 1⌝]
        less_trans_thm));
(* *** Goal "2.2" *** *)
a(rewrite_tac[get_spec⌜Length⌝] THEN REPEAT strip_tac);
pop_thm()
));
```

SML

```
val distinct_thm1 = save_thm("distinct_thm1", (
push_goal([], ⌜∀list1 list2•
                list1 ∈ Distinct ∧ Elems list1 = Elems list2
        ⇒       Length list1 ≤ Length list2⌝);
a(strip_tac);
a(list_induction_tac⌜list1⌝);
(* *** Goal "1" *** *)
```

```
a(rewrite_tac(elems_thm2 :: map get_spec[⌜Elems⌝, ⌜Distinct⌝, ⌜Length⌝]));
(* *** Goal "2" *** *)
a(REPEAT strip_tac);
a(lemma_tac ⌜Length ([x] @ (list2 ↾ ∼{x})) ≤ Length list2⌝);
(* *** Goal "2.1" *** *)
a(rewrite_tac[get_spec⌜Length⌝, length_thm]);
a(cases_tac⌜x ∈ Elems list2⌝);
(* *** Goal "2.1.1" *** *)
a(strip_asm_tac(list_∀_elim[⌜list2⌝, ⌜x⌝]↾_thm4));
a(asm_ante_tac⌜Length (list2 ↾ ∼ {x}) < Length list2⌝ THEN
        rewrite_tac[∀_elim⌜1⌝plus_order_thm, less_def]);
(* *** Goal "2.1.2" *** *)
a(lemma_tac⌜Elems list2 ⊆ ∼{x}⌝);
(* *** Goal "2.1.2.1" *** *)
a(PC_T"hol1"(REPEAT strip_tac THEN rewrite_tac[]));
a(swap_asm_concl_tac⌜x′ ∈ Elems list2⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.2" *** *)
a(lemma_tac⌜x ∈ Elems list2⌝);
a(asm_ante_tac ⌜Elems (Cons x list1) = Elems list2⌝);
a(rewrite_tac[get_spec⌜Elems⌝]);
a(PC_T"hol1"(REPEAT strip_tac));
a(spec_nth_asm_tac 1⌜x⌝);
(* *** Goal "2.2" *** *)
a(DROP_NTH_ASM_T 3 (strip_asm_tac o rewrite_rule[get_spec⌜Distinct⌝]));
a(lemma_tac⌜Elems list1 = Elems (list2 ↾ ∼ {x})⌝);
(* *** Goal "2.2.2" *** *)
a(rewrite_tac[↾_thm1]);
a(DROP_NTH_ASM_T 4 (rewrite_thm_tac o eq_sym_rule));
a(rewrite_tac[get_spec⌜Elems⌝]);
a(PC_T"hol1"(REPEAT strip_tac));
(* *** Goal "2.2.1.1" *** *)
a(PC_T"hol1"(rewrite_tac[]));
a(swap_asm_concl_tac⌜x′ ∈ Elems list1⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2.2.1.2" *** *)
a(TOP_ASM_T (strip_asm_tac o rewrite_rule[]));
(* *** Goal "2.2.2" *** *)
a(DROP_NTH_ASM_T 6 (strip_asm_tac o ∀_elim⌜list2 ↾ ∼ {x}⌝));
a(rewrite_tac(map get_spec[⌜Length⌝]));
a(swap_nth_asm_concl_tac 5 THEN rewrite_tac(map get_spec[⌜$Append⌝, ⌜Length⌝]));
a(lemma_tac⌜x ∈ Elems list2⌝);
(* *** Goal "2.2.2.1" *** *)
a(GET_NTH_ASM_T 6 (fn th => rewrite_tac[eq_sym_rule th, get_spec⌜Elems⌝]));
```

```
a(PC_T"hol1" (REPEAT strip_tac));
(* *** Goal "2.2.2.2" *** *)
a(strip_asm_tac(list_∀_elim[⌜list2⌝, ⌜x⌝]↾_thm4));
a(swap_nth_asm_concl_tac 1);
a(rewrite_tac[get_spec⌜$<⌝]);
a(strip_asm_tac(list_∀_elim
        [⌜Length list1 + 1⌝, ⌜Length (list2 ↾ ~ {x}) + 1⌝, ⌜Length list2⌝]
        ≤_trans_thm));
pop_thm()
));
```

SML

```
val size_thm1 = save_thm("size_thm1", (
push_goal([], ⌜Size {} = 0⌝);
a(rewrite_tac[get_spec⌜Size⌝, elems_thm1]);
a(conv_tac (ONCE_MAP_C prove_∃_conv));
a(rewrite_tac[get_spec⌜Length⌝]);
a(lemma_tac⌜{i|0 = i} = {0}⌝ );
(* *** Goal "1" *** *)
a(PC_T"hol1" (REPEAT strip_tac)
        THEN TRY_T (asm_ante_tac ⌜¬x = 0⌝) THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[min_clauses]);
pop_thm()
));
```

SML

```
val size_thm2 = save_thm("size_thm2", (
push_goal([], ⌜∀list• list ∈ Distinct ⇒ Size(Elems list) = Length list⌝);
a(rewrite_tac[get_spec⌜Size⌝] THEN REPEAT strip_tac);
a(strip_asm_tac (list_∀_elim
        [⌜Length list⌝, ⌜{i|∃ list'• Length list' = i ∧ Elems list' = Elems list}⌝]
        min_thm));
(* *** Goal "1" *** *)
a(POP_ASM_T (strip_asm_tac o rewrite_rule[] o ∀_elim⌜list⌝));
a(TOP_ASM_T ante_tac THEN GET_NTH_ASM_T 3 (rewrite_thm_tac o eq_sym_rule));
a(strip_tac THEN strip_asm_tac (list_∀_elim
        [⌜Length list⌝, ⌜Length list'⌝]
        ≤_antisym_thm));
(* *** Goal "2.1" *** *)
a(strip_asm_tac (list_∀_elim[⌜list⌝, ⌜list'⌝]distinct_thm1));
a(asm_ante_tac ⌜¬ Elems list = Elems list'⌝);
a(GET_NTH_ASM_T 5 rewrite_thm_tac);
```

```
(* *** Goal "2.2" *** *)
a(GET_NTH_ASM_T 2 rewrite_thm_tac);
pop_thm()
));
```

3.5 Relationship between *Finite* and Set Operations

SML

```
val fin_set_thm2 = save_thm("fin_set_thm2", (
push_goal([], ⌜{} ∈ Finite⌝);
a(rewrite_tac[get_spec⌜Finite⌝]);
a(PC_T "hol1" (REPEAT strip_tac));
pop_thm()
));
```

SML

```
val fin_set_thm3 = save_thm("fin_set_thm3", (
push_goal([], ⌜∀ a x• a ∈ Finite ⇒ ({x} ∪ a) ∈ Finite⌝);
a(rewrite_tac[get_spec⌜Finite⌝] THEN (PC_T "hol1" (REPEAT strip_tac)));
a(spec_nth_asm_tac 3 ⌜s⌝);
(* *** Goal "1" *** *)
a(list_spec_nth_asm_tac 3 [⌜a'⌝, ⌜x'⌝]);
(* *** Goal "2" *** *)
a(list_spec_nth_asm_tac 2 [⌜a⌝, ⌜x⌝]);
pop_thm()
));
```

SML

```
val fin_set_thm4 = save_thm("fin_set_thm4", (
push_goal([], ⌜∀ list• Elems list ∈ Finite⌝);
a(strip_tac);
a(list_induction_tac⌜list⌝);
(* *** Goal "1" *** *)
a(rewrite_tac[get_spec ⌜Elems⌝, fin_set_thm2]);
(* *** Goal "2" *** *)
a(rewrite_tac[get_spec ⌜Elems⌝]);
a(strip_tac);
a(strip_asm_tac (list_∀_elim[⌜Elems list⌝, ⌜x⌝]fin_set_thm3));
pop_thm()
));
```

SML

```
val size_thm3 = save_thm("size_thm3", (
push_goal([], ⌜∀a b• a ∈ Finite ∧ b ∈ Finite
        ⇒              (a ∪ b) ∈ Finite⌝);
a(REPEAT strip_tac);
a(strip_asm_tac (∀_elim ⌜a⌝ fin_set_thm1));
a(strip_asm_tac (∀_elim ⌜b⌝ fin_set_thm1));
a(asm_rewrite_tac[conv_rule(ONCE_MAP_C eq_sym_conv) elems_thm3]);
a(prove_tac[fin_set_thm4]);
pop_thm()
));
```

SML

```
val size_thm4 = save_thm("size_thm4", (
push_goal([], ⌜∀a b• a ∈ Finite ∧ b ⊆ a
        ⇒              b ∈ Finite⌝);
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elim⌜a⌝fin_set_thm1));
a(LEMMA_T ⌜b = Elems (list ↑ b)⌝ once_rewrite_thm_tac);
(* *** Goal "1" *** *)
a(DROP_NTH_ASM_T 2 (fn th => rewrite_tac[eq_sym_rule th, ↑_thm1]));
a(PC_T"hol1"(asm_prove_tac[]));
(* *** Goal "2" *** *)
a(rewrite_tac[fin_set_thm4]);
pop_thm()
));
```

SML

```
val size_thm5 = save_thm("size_thm5", (
push_goal([], ⌜∀a x• a ∈ Finite
        ⇒       Size({x} ∪ a) = if x ∈ a then Size a else Size a + 1⌝);
a(REPEAT strip_tac);
a(cases_tac⌜x ∈ a⌝);
(* *** Goal "1" *** *)
a(LEMMA_T ⌜{x} ∪ a = a⌝ asm_rewrite_thm_tac);
a(PC_T"hol1"(REPEAT strip_tac));
a(asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(strip_asm_tac(list_∀_elim[⌜a⌝]fin_set_thm1));
a(lemma_tac⌜{x} ∪ a = Elems (Cons x list) ∧ Cons x list ∈ Distinct⌝);
(* *** Goal "2.1" *** *)
a(asm_rewrite_tac[get_spec⌜Elems⌝, get_spec⌜Distinct⌝]);
a(DROP_NTH_ASM_T 2 (asm_rewrite_thm_tac o eq_sym_rule));
```

```
(* *** Goal "2.2" *** *)
a(strip_asm_tac(list_∀_elim[⌜list⌝]size_thm2));
a(strip_asm_tac(list_∀_elim[⌜Cons x list⌝]size_thm2));
a(LIST_GET_NTH_ASM_T [7, 4, 1] rewrite_tac);
a(LIST_GET_NTH_ASM_T [6, 2] rewrite_tac);
a(rewrite_tac[get_spec⌜Length⌝]);
pop_thm()
));
```

SML

```
val size_thm6 = save_thm("size_thm6", (
push_goal([], ⌜∀a b• a ∈ Finite ∧ b ∈ Finite ⇒ (a ∩ b) ∈ Finite⌝);
a(REPEAT strip_tac);
a(strip_asm_tac(pc_rule"hol1"(prove_rule[])⌜a ∩ b ⊆ a⌝));
a(strip_asm_tac(list_∀_elim[⌜a⌝, ⌜a ∩ b⌝]size_thm4));
pop_thm()
));
```

SML

```
val size_thm7 = save_thm("size_thm7", (
push_goal([], ⌜∀a b• a ∈ Finite ∧ b ∈ Finite
      ⇒          Size(a ∪ b) + Size (a ∩ b) = Size a + Size b⌝);
a(REPEAT strip_tac);
a(asm_ante_tac ⌜a ∈ Finite⌝ THEN fin_set_induction_tac);
(* *** Goal "1" *** *)
a(rewrite_tac[size_thm1]);
(* *** Goal "2" *** *)
a(strip_asm_tac(list_∀_elim[⌜a⌝, ⌜b⌝]size_thm3));
a(strip_asm_tac(list_∀_elim[⌜a⌝, ⌜b⌝]size_thm6));
a(cases_tac⌜x ∈ b⌝);
(* *** Goal "2.1" *** *)
a(lemma_tac ⌜({x} ∪ a) ∪ b = a ∪ b ∧
      ({x} ∪ a) ∩ b = {x} ∪ (a ∩ b)⌝);
(* *** Goal "2.1.1" *** *)
a(PC_T"hol1"(REPEAT strip_tac THEN asm_rewrite_tac[]));
(* *** Goal "2.1.2" *** *)
a(strip_asm_tac(list_∀_elim[⌜a ∪ b⌝, ⌜x⌝]size_thm5));
a(strip_asm_tac(list_∀_elim[⌜a ∩ b⌝, ⌜x⌝]size_thm5));
a(strip_asm_tac(list_∀_elim[⌜a⌝, ⌜x⌝]size_thm5));
a(LEMMA_T ⌜x ∈ a ∪ b ∧ ¬x ∈ a ∩ b⌝
      (fn th => asm_rewrite_tac[th, ∀_elim⌜1⌝plus_order_thm]));
a(REPEAT strip_tac);
(* *** Goal "2.2" *** *)
```

```
a(lemma_tac ⌜({x} ∪ a) ∪ b = {x} ∪ a ∪ b ∧
        ({x} ∪ a) ∩ b = (a ∩ b)⌝);
(* *** Goal "2.2.1" *** *)
a(PC_T"hol1"(REPEAT strip_tac THEN asm_rewrite_tac[]));
a(asm_ante_tac ⌜x' ∈ b⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2" *** *)
a(strip_asm_tac(list_∀_elim[⌜a ∪ b⌝, ⌜x⌝]size_thm5));
a(strip_asm_tac(list_∀_elim[⌜a ∩ b⌝, ⌜x⌝]size_thm5));
a(strip_asm_tac(list_∀_elim[⌜a⌝, ⌜x⌝]size_thm5));
a(LEMMA_T ⌜¬x ∈ a ∪ b ∧ ¬x ∈ a ∩ b⌝
        (fn th => asm_rewrite_tac[th, ∀_elim⌜1⌝plus_order_thm]));
a(REPEAT strip_tac);
pop_thm()
));
```

SML

```
val size_singleton_thm = save_thm("size_singleton_thm", (
push_goal([], ⌜∀x• #{x} = 1⌝);
a strip_tac;
a(accept_tac(rewrite_rule[fin_set_thm2, size_thm1]
        (list_∀_elim[⌜{}⌝, ⌜x⌝]size_thm5)));
pop_thm()
));
```

3.6 Miscellany

SML

```
val ℕ_set_thm1 = save_thm("ℕ_set_thm1", (
push_goal([], ⌜∀a: ℕ SET• a ∈ Finite ∧ ¬a = {} ⇒ ∃m•m ∈ a ∧ ∀i•i ∈ a ⇒ i ≤ m⌝);
a(strip_tac THEN bc_tac[taut_rule⌜(A ⇒ B ⇒ C) ⇒ (A ∧ B ⇒ C)⌝]);
a(fin_set_induction_tac THEN_TRY asm_rewrite_tac[] THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(∃_tac⌜x:ℕ⌝ THEN REPEAT strip_tac THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(strip_asm_tac(list_∀_elim[⌜x⌝, ⌜m⌝]≤_cases_thm));
(* *** Goal "2.1" *** *)
a(∃_tac⌜m⌝ THEN REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
a(asm_fc_tac[]);
(* *** Goal "2.2" *** *)
a(∃_tac⌜x⌝ THEN REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN rename_tac[(⌜i⌝, "ii")]);
a(bc_tac[≤_trans_thm]);
a(∃_tac⌜m⌝ THEN REPEAT strip_tac);
```

```
pop_thm()
));
```

SML

```
val finite_max_thm = save_thm("finite_max_thm", (
push_goal([], ⌜∀a: ℕ SET• a ∈ Finite ∧ ¬a = {} ⇒ Max a ∈ a ∧ ∀i•i ∈ a ⇒ i ≤ Max a⌝);
a(REPEAT_N 2 strip_tac);
a(fc_tac[ℕ_set_thm1]);
a(LEMMA_T ⌜Max a = m⌝ (fn th => rewrite_tac[th] THEN REPEAT strip_tac));
(* *** Goal "1" *** *)
a(bc_tac[get_spec⌜Max⌝] THEN REPEAT strip_tac THEN asm_fc_tac[]);
(* *** Goal "2" *** *)
a(asm_fc_tac[]);
pop_thm()
));
```

SML

```
val finite_size_thm = save_thm("finite_size_thm", (
push_goal([], ⌜∀a m•
        (∃list•Elems list = a ∧ list ∈ Distinct ∧ Length list = m)
⇔       a ∈ Finite ∧ #a = m
⌝);
a(REPEAT strip_tac);
(* *** Goal "1" *** *)
a(GET_NTH_ASM_T 3 (fn th => rewrite_tac[eq_sym_rule th, fin_set_thm4]));
(* *** Goal "2" *** *)
a(fc_tac[size_thm2]);
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "3" *** *)
a(fc_tac[fin_set_thm1]);
a(∃_tac⌜list⌝ THEN asm_rewrite_tac[]);
a(fc_tac[size_thm2]);
a(LIST_DROP_NTH_ASM_T [1,3,4] (rewrite_tac o map eq_sym_rule));
pop_thm()
));
```

SML

```
val length_map_thm = save_thm("length_map_thm", (
push_goal([], ⌜∀f list•
        Length(Map f list) = Length list
⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜list⌝ THEN asm_rewrite_tac[length_def, map_def]);
```

```
pop_thm()
));
```

SML

```
val elems_map_thm = save_thm("elems_map_thm", (
push_goal([], ⌜∀f list•
        Elems(Map f list) = {y | ∃x• x ∈ Elems list ∧ f x = y}
⌝);
a(REPEAT strip_tac);
a(list_induction_tac⌜list⌝ THEN asm_rewrite_tac[elems_def, map_def]
        THEN PC_T"hol1"(REPEAT strip_tac));
(* *** Goal "1" *** *)
a(∃_tac⌜x⌝ THEN PC_T"hol1"(REPEAT strip_tac) THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(∃_tac⌜x''⌝ THEN PC_T"hol1"(REPEAT strip_tac));
(* *** Goal "3" *** *)
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
a(strip_tac THEN asm_rewrite_tac[]);
(* *** Goal "4" *** *)
a(∃_tac⌜x''⌝ THEN PC_T"hol1"(REPEAT strip_tac));
pop_thm()
));
```

SML

```
val map_distinct_thm = save_thm("map_distinct_thm", (
push_goal([], ⌜∀f list•
        (Map f list) ∈ Distinct
⇔       list ∈ Distinct
        ∧ (∀x y• x ∈ Elems list ∧ y ∈ Elems list ∧ f x = f y ⇒ x = y)
⌝);
a(REPEAT ∀_tac);
a(list_induction_tac⌜list⌝);
(* *** Goal "1" of 5 *** *)
a(asm_rewrite_tac[elems_def, map_def, distinct_def]);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[elems_def, map_def, distinct_def]);
(* *** Goal "3" *** *)
a(rename_tac[]);
a(asm_rewrite_tac[elems_def, map_def, distinct_def]);
a(PC_T"hol1"(REPEAT strip_tac));
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "4" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
```

```
(* *** Goal "5" *** *)
a(asm_rewrite_tac[elems_def, map_def, distinct_def]);
a(rewrite_tac[taut_rule⌜∀p q r•(¬p ⇔ ¬q ∧ r) ⇔ (p ⇔ q ∨ ¬r)⌝, elems_map_thm]);
a(REPEAT_N 3 strip_tac);
(* *** Goal "5.1" *** *)
a(strip_tac THEN cases_tac ⌜x ∈ Elems list⌝ THEN1 REPEAT strip_tac);
a(PC_T"hol1"(REPEAT strip_tac));
a(list_spec_nth_asm_tac 1 [⌜x'⌝, ⌜x⌝]);
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule) THEN strip_tac);
(* *** Goal "5.2" *** *)
a(cases_tac ⌜x ∈ Elems list⌝);
(* *** Goal "5.2.1" *** *)
a(⇒_T (fn th => id_tac));
a(∃_tac⌜x⌝ THEN asm_rewrite_tac[]);
(* *** Goal "5.2.2" *** *)
a(asm_rewrite_tac[]);
a(once_rewrite_tac[taut_rule⌜∀p q•(¬p ⇒ q) ⇔ (¬q ⇒ p)⌝]);
a(PC_T"hol1"(REPEAT strip_tac));
(* *** Goal "5.2.2.1" *** *)
a(asm_rewrite_tac[]);
(* *** Goal "5.2.2.2" *** *)
a(list_spec_nth_asm_tac 4 [⌜y⌝]);
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
a(⇒_T rewrite_thm_tac);
(* *** Goal "5.2.2.3" *** *)
a(list_spec_nth_asm_tac 4 [⌜x'⌝]);
a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac THEN asm_rewrite_tac[]);
a(⇒_T rewrite_thm_tac);
(* *** Goal "5.2.2.4" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
pop_thm()
));
```

SML

```
val pair_eq_thm = save_thm("pair_eq_thm", (
push_goal([], ⌜∀x y• Fst x = Fst y ∧ Snd x = Snd y ⇒ x = y⌝);
a(REPEAT strip_tac);
a(LEMMA_T⌜y = (Fst x, Snd x)⌝ rewrite_thm_tac);
a(pure_asm_rewrite_tac[] THEN rewrite_tac[]);
pop_thm()
));
```

SML

```
val at_thm = save_thm("at_thm", (
push_goal([], ⌜∀f: 'a ↔ 'b; x : 'a•
        f ∈ Functional ∧ x ∈ Dom f
        ⇒ ∀y•f@x = y ⇔ (x, y) ∈ f
⌝);
a(rewrite_tac[functional_def, dom_def]);
a(REPEAT strip_tac);
(* *** Goal "1" *** *)
a(fc_tac[get_spec⌜$At⌝]);
(* *** Goal "1.1" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "1.2" *** *)
a(DROP_ASM_T⌜f @ x = y'⌝(asm_rewrite_thm_tac o eq_sym_rule));
(* *** Goal "2" *** *)
a(fc_tac[get_spec⌜$At⌝]);
(* *** Goal "2.1" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "2.2" *** *)
a(asm_rewrite_tac[] THEN asm_fc_tac[] THEN asm_fc_tac[]);
pop_thm()
));
```

SML

```
val finite_function_thm = save_thm("finite_function_thm", (
push_goal([], ⌜∀f: 'a ↔ 'b•
                f ∈ Functional
        ∧       (f ∈ Finite ∨ Dom f ∈ Finite)
        ⇒       f ∈ Finite ∧ Dom f ∈ Finite ∧ #(Dom f) = #f
⌝);
a(REPEAT_N 2 strip_tac);
(* *** Goal "1" *** *)
a(asm_rewrite_tac[]);
a(fc_tac[fin_set_thm1]);
a(bc_tac[finite_size_thm]);
a(∃_tac⌜Map Fst list⌝);
a(fc_tac[size_thm2]);
a(asm_rewrite_tac[length_map_thm, elems_map_thm, map_distinct_thm, dom_def]);
a(PC_T"hol1"(REPEAT strip_tac));
(* *** Goal "1.1" *** *)
a(∃_tac⌜Snd x'⌝ THEN POP_ASM_T (asm_rewrite_thm_tac o eq_sym_rule));
(* *** Goal "1.2" *** *)
a(∃_tac⌜(x, y)⌝ THEN asm_rewrite_tac[]);
```

```
(* *** Goal "1.3" *** *)
a(DROP_ASM_T⌜f = Elems list⌝ (asm_tac o eq_sym_rule));
a(LIST_DROP_NTH_ASM_T [3,4] (MAP_EVERY ante_tac) THEN asm_rewrite_tac[]);
a(DROP_ASM_T⌜f ∈ Functional⌝ (strip_asm_tac o rewrite_rule[get_spec⌜Functional⌝]));
a(REPEAT strip_tac);
a(lemma_tac ⌜(Fst x, Snd y) ∈ f⌝ THEN1 asm_rewrite_tac[]);
a(LIST_SPEC_NTH_ASM_T 4 [⌜Fst x⌝, ⌜Snd x⌝, ⌜Snd y⌝] (strip_asm_tac o rewrite_rule[]));
a(bc_tac[pair_eq_thm] THEN REPEAT strip_tac);
(* *** Goal "2" *** *)
a(asm_rewrite_tac[]);
a(fc_tac[fin_set_thm1]);
a(conv_tac(ONCE_MAP_C eq_sym_conv) THEN bc_tac[finite_size_thm]);
a(∃_tac⌜Map (λx•(x, f@x)) list⌝);
a(fc_tac[size_thm2]);
a(asm_rewrite_tac[length_map_thm, elems_map_thm, map_distinct_thm, dom_def]);
a(PC_T"hol1"(REPEAT strip_tac));
(* *** Goal "2.1" *** *)
a(DROP_NTH_ASM_T 6 (asm_tac o eq_sym_rule));
a(DROP_NTH_ASM_T 4 ante_tac THEN asm_rewrite_tac[] THEN strip_tac);
a(fc_tac[at_thm] THEN asm_fc_tac[]);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac[] THEN strip_tac);
a(asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac[]);
(* *** Goal "2.2" *** *)
a(lemma_tac⌜Fst x ∈ Dom f⌝);
(* *** Goal "2.2.1" *** *)
a(rewrite_tac[dom_def] THEN ∃_tac⌜Snd x⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2" *** *)
a(lemma_tac⌜x = (Fst x, f@(Fst x))⌝);
(* *** Goal "2.2.2.1" *** *)
a(fc_tac[conv_rule (ONCE_MAP_C eq_sym_conv) at_thm] THEN asm_fc_tac[]);
a(POP_ASM_T (ante_tac o ∀_elim⌜Snd x⌝) THEN rewrite_tac[]);
a(strip_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.2.2.2" *** *)
a(DROP_NTH_ASM_T 6 (rewrite_thm_tac o eq_sym_rule));
a(POP_ASM_T once_rewrite_thm_tac THEN ∃_tac⌜Fst x⌝ THEN asm_rewrite_tac[]);
pop_thm()
));
```

4 THE THEORY fin_thms

4.1 Parents

fin_set

4.2 Theorems

fin_set_induction_thm
$\vdash \forall\ P$
$\bullet\ P\ \{\}$
$\wedge\ (\forall\ a\ x$
$\bullet\ a \in Finite \wedge P\ a \wedge \neg\ x \in a \Rightarrow P\ (\{x\} \cup a))$
$\Rightarrow\ (\forall\ a \bullet\ a \in Finite \Rightarrow P\ a)$

min_thm $\vdash \forall\ m\ a \bullet\ m \in a \Rightarrow Min\ a \in a \wedge Min\ a \leq m$

min_clauses $\vdash (\forall\ m \bullet\ Min\ \{m\} = m)$
$\wedge\ (\forall\ m\ n \bullet\ Min\ \{m;\ n\} = (if\ m \leq n\ then\ m\ else\ n))$
$\wedge\ (\forall\ m\ a$
$\bullet\ Min\ (\{m\} \cup a)$
$= (if\ a = \{\}\ then\ m\ else\ Min\ \{m;\ Min\ a\}))$

fin_set_thm1 $\vdash \forall\ a$
$\bullet\ a \in Finite$
$\Rightarrow\ (\exists\ list \bullet\ a = Elems\ list \wedge list \in Distinct)$

elems_thm1 $\vdash \forall\ list \bullet\ Elems\ list = \{\} \Leftrightarrow list = []$

elems_thm2 $\vdash (\forall\ list \bullet\ Elems\ list = \{\} \Leftrightarrow list = [])$
$\wedge\ (\forall\ list \bullet\ \{\} = Elems\ list \Leftrightarrow list = [])$

elems_thm3 $\vdash \forall\ list1\ list2$
$\bullet\ Elems\ (list1 \frown list2) = Elems\ list1 \cup Elems\ list2$

length_thm $\vdash \forall\ list1\ list2 \bullet\ \#\ (list1 \frown list2) = \#\ list1 + \#\ list2$

↾_thm1 $\vdash \forall\ list\ a \bullet\ Elems\ (list \upharpoonright a) = Elems\ list \cap a$

↾_thm2 $\vdash \forall\ list\ a \bullet\ \#\ ((list \upharpoonright a) \frown (list \upharpoonright \sim a)) = \#\ list$

↾_thm3 $\vdash \forall\ list\ a \bullet\ Elems\ list \subseteq a \Rightarrow list \upharpoonright a = list$

↾_thm4 $\vdash \forall\ list\ x \bullet\ x \in Elems\ list \Rightarrow \#\ (list \upharpoonright \sim \{x\}) < \#\ list$

distinct_thm1
$\vdash \forall\ list1\ list2$
$\bullet\ list1 \in Distinct \wedge Elems\ list1 = Elems\ list2$
$\Rightarrow\ \#\ list1 \leq \#\ list2$

size_thm1 $\vdash \#\ \{\} = 0$

size_thm2 $\vdash \forall\ list \bullet\ list \in Distinct \Rightarrow \#\ (Elems\ list) = \#\ list$

fin_set_thm2 $\vdash \{\} \in Finite$

fin_set_thm3 $\vdash \forall\ a\ x \bullet\ a \in Finite \Rightarrow \{x\} \cup a \in Finite$

fin_set_thm4 $\vdash \forall\ list \bullet\ Elems\ list \in Finite$

size_thm3 $\vdash \forall\ a\ b \bullet\ a \in Finite \wedge b \in Finite \Rightarrow a \cup b \in Finite$

size_thm4 $\vdash \forall\ a\ b \bullet\ a \in Finite \wedge b \subseteq a \Rightarrow b \in Finite$

size_thm5 $\vdash \forall\ a\ x$
$\bullet\ a \in Finite$
$\Rightarrow\ \#\ (\{x\} \cup a) = (if\ x \in a\ then\ \#\ a\ else\ \#\ a + 1)$

size_thm6 $\vdash \forall\ a\ b \bullet\ a \in Finite \wedge b \in Finite \Rightarrow a \cap b \in Finite$

size_thm7 $\vdash \forall\ a\ b$
$\bullet\ a \in Finite \wedge b \in Finite$
$\Rightarrow \#\ (a \cup b) + \#\ (a \cap b) = \#\ a + \#\ b$

size_singleton_thm
$\vdash \forall\ x \bullet\ \#\ \{x\} = 1$

$\mathbb{N}$***_set_thm1*** $\vdash \forall\ a$
$\bullet\ a \in Finite \wedge \neg\ a = \{\}$
$\Rightarrow (\exists\ m \bullet\ m \in a \wedge (\forall\ i \bullet\ i \in a \Rightarrow i \leq m))$

finite_max_thm
$\vdash \forall\ a$
$\bullet\ a \in Finite \wedge \neg\ a = \{\}$
$\Rightarrow Max\ a \in a \wedge (\forall\ i \bullet\ i \in a \Rightarrow i \leq Max\ a)$

finite_size_thm
$\vdash \forall\ a\ m$
$\bullet\ (\exists\ list$
$\bullet\ Elems\ list = a \wedge list \in Distinct \wedge \#\ list = m)$
$\Leftrightarrow a \in Finite \wedge \#\ a = m$

length_map_thm
$\vdash \forall\ f\ list \bullet\ \#\ (Map\ f\ list) = \#\ list$

elems_map_thm
$\vdash \forall\ f\ list$
$\bullet\ Elems\ (Map\ f\ list)$
$= \{y | \exists\ x \bullet\ x \in Elems\ list \wedge f\ x = y\}$

map_distinct_thm
$\vdash \forall\ f\ list$
$\bullet\ Map\ f\ list \in Distinct$
$\Leftrightarrow list \in Distinct$
$\wedge (\forall\ x\ y$
$\bullet\ x \in Elems\ list \wedge y \in Elems\ list \wedge f\ x = f\ y$
$\Rightarrow x = y)$

pair_eq_thm $\vdash \forall\ x\ y \bullet\ Fst\ x = Fst\ y \wedge Snd\ x = Snd\ y \Rightarrow x = y$

at_thm $\vdash \forall\ f\ x$
$\bullet\ f \in Functional \wedge x \in Dom\ f$
$\Rightarrow (\forall\ y \bullet\ f\ @\ x = y \Leftrightarrow (x, y) \in f)$

finite_function_thm
$\vdash \forall\ f$
$\bullet\ f \in Functional \wedge (f \in Finite \vee Dom\ f \in Finite)$
$\Rightarrow f \in Finite \wedge Dom\ f \in Finite \wedge \#\ (Dom\ f) = \#\ f$

5 INDEX