Interactive Theorem Proving in HOL4

Course 08: Set Theory

Dr Chun TIAN
chun.tian@anu.edu.au

19 September 2024

Australian National University

Acknowledgement of Country

> We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respect to the elders past and present.

More information about Acknowledgement of Country can be found here and here

Rewriting tactics with simplification sets

RW_TAC std_ss [...] includes STRIP_TAC, case-splitting and built-in knowledge of declared datatypes, is the first tactic in many proofs.

```
RW_TAC       : simpset -> thm list -> tactic (* rewriting + striping *)
SIMP_TAC     : simpset -> thm list -> tactic (* higher-order rewriting *)
ASM_SIMP_TAC : simpset -> thm list -> tactic (* using assumptions *)
```

built-in simplification sets

- bool_ss: rewrite rules for propositions and first order terms;
- std_ss: bool_ssplus rewrite rules for terms involving options, pairs, and sums;
- arith_ss: std_ss plus rewrite rules for arithmetics (of num).
- list_ss: std_ss plus rewrite rules for lists.
- real_ss: arith_ss plus rewrite rules for real arithmetics.

Advanced rewriting tactics (bossLib)

These tactics uses stateful simplification set (`srw_ss()`), i.e. theorems marked with `[simp]`.

```
val simp : thm list -> tactic (* no touching assumptions *)
val csimp : thm list -> tactic
val dsimp : thm list -> tactic
val lrw : thm list -> tactic
val lfs : thm list -> tactic
val lrfs : thm list -> tactic
val rw : thm list -> tactic (* strip the goal *)
val fs : thm list -> tactic (* rewrite assumptions *)
val rfs : thm list -> tactic
val gs : thm list -> tactic (* = fs + rfs *)
val gvs : thm list -> tactic
val gns : thm list -> tactic
val gnvs : thm list -> tactic
val rgs : thm list -> tactic
```

First-order automatic solvers

The following two solvers can solve a goal with assumptions and supplied theorems:

```
PROVE_TAC : thm list -> tactic
METIS_TAC : thm list -> tactic
```

In theory, if you can provide the list of *all* needed theorems for solving a goal, these automatic solvers will search the proof for you (but may takes a lot of time).

More set operators (1)

$\vdash \emptyset = (\lambda x.\ \mathrm{F})$	[EMPTY_DEF]
$\vdash \mathcal{U}(:\alpha) = (\lambda x.\ \mathrm{T})$	[UNIV_DEF]
$\vdash s \subseteq t \iff \forall x.\ x \in s \Rightarrow x \in t$	[SUBSET_DEF]
$\vdash s \cap t = \{x \mid x \in s \land x \in t\}$	[INTER_DEF]
$\vdash s \cup t = \{x \mid x \in s \lor x \in t\}$	[UNION_DEF]
$\vdash \bigcap P = \{x \mid \forall s.\ s \in P \Rightarrow x \in s\}$	[BIGINTER]
$\vdash \bigcup P = \{x \mid \exists s.\ s \in P \land x \in s\}$	[BIGUNION]
$\vdash s\ \mathrm{DIFF}\ t = \{x \mid x \in s \land x \notin t\}$	[DIFF_DEF]
$\vdash \mathrm{COMPL}\ P = \mathcal{U}(:\alpha)\ \mathrm{DIFF}\ P$	[COMPL_DEF]
$\vdash \mathrm{POW}\ set = \{s \mid s \subseteq set\}$	[POW_DEF]
$\vdash x\ \mathrm{INSERT}\ s = \{y \mid y = x \lor y \in s\}$	[INSERT_DEF]
$\vdash \mathrm{SING}\ s \iff \exists x.\ s = \{x\}$	[SING_DEF]
$\vdash s\ \mathrm{DELETE}\ x = s\ \mathrm{DIFF}\ \{x\}$	[DELETE_DEF]
$\vdash s \neq \emptyset \Rightarrow \mathrm{CHOICE}\ s \in s$	[CHOICE_DEF]
$\vdash \mathrm{REST}\ s = s\ \mathrm{DELETE}\ \mathrm{CHOICE}\ s$	[REST_DEF]

More set operators (2)

$\vdash$ IMAGE $(f : \alpha \to \beta)$ $(s : \alpha \to$ bool$)$ = $\{f\ x \mid x \in s\}$ [IMAGE_DEF]

$\vdash$ PREIMAGE $(f : \alpha \to \beta)$ $(s : \beta \to$ bool$)$ = $\{x \mid f\ x \in s\}$ [PREIMAGE_def]

$\vdash$ INJ $f\ s\ t \iff$
$(\forall x.\ x \in s \Rightarrow f\ x \in t) \land$
$\forall x\ y.\ x \in s \land y \in s \Rightarrow f\ x = f\ y \Rightarrow x = y$ [INJ_DEF]

$\vdash$ SURJ $f\ s\ t \iff$
$(\forall x.\ x \in s \Rightarrow f\ x \in t) \land \forall x.\ x \in t \Rightarrow \exists y.\ y \in s \land f\ y = x$ [SURJ_DEF]

$\vdash$ BIJ $f\ s\ t \iff$ INJ $f\ s\ t \land$ SURJ $f\ s\ t$ [BIJ_DEF]

[FINITE_DEF]

$\vdash$ FINITE $s \iff \forall P.\ P\ \emptyset \land (\forall s.\ P\ s \Rightarrow \forall e.\ P\ (e$ INSERT $s)) \Rightarrow P\ s$

$\vdash$ CARD $\emptyset = 0 \land$
$\forall s.$ FINITE $s \Rightarrow$
$\forall x.$ CARD $(x$ INSERT $s)$ =
if $x \in s$ **then** CARD s **else** SUC (CARD s) [CARD_DEF]

$\vdash s$ HAS_SIZE $n \iff$ FINITE $s \land$ CARD $s = n$ [HAS_SIZE]

Induction on finite sets

[FINITE_INDUCT]
$\vdash P\ \emptyset \wedge (\forall s.\ \texttt{FINITE}\ s \wedge P\ s \Rightarrow \forall e.\ e \notin s \Rightarrow P\ (e\ \texttt{INSERT}\ s)) \Rightarrow$
$\quad \forall s.\ \texttt{FINITE}\ s \Rightarrow P\ s$

[FINITE_COMPLETE_INDUCTION]
$\vdash (\forall x.\ (\forall y.\ y \subset x \Rightarrow P\ y) \Rightarrow \texttt{FINITE}\ x \Rightarrow P\ x) \Rightarrow$
$\quad \forall x.\ \texttt{FINITE}\ x \Rightarrow P\ x$

$\vdash s \subset t \iff s \subseteq t \wedge s \neq t$ [PSUBSET_DEF]

using irule

Instead of HO_MATCH_MP_TAC, the tactic irule is necessary for applying the above induction theorems (of form $\forall P.\ R \Rightarrow \forall x.Q(x) \Rightarrow P(x)$) for the goal matching $P(x)$.

Set representations

Explicit sets

In HOL, `{a;b;c}` is abbreviation of "`a INSERT b INSERT c INSERT {}`".

Set comprehension

In HOL, `{t | P}` is abbrev. of "$\mathtt{GSPEC}\,(\lambda(x_1, x_2, \ldots, x_n).\,(t, P))$" where x_i are free variables of t.

$\vdash v \in \mathtt{GSPEC}\ f \iff \exists x.\ (v,\mathtt{T}) = f\ x$ [GSPECIFICATION]

For example,

$$
\begin{aligned}
& a \in \{p + q \,|\, p < q \land q < r\} \\
\iff & a \in \mathtt{GSPEC}\,(\lambda(p, q).\,(p + q, p < q \land q < r)) \\
\iff & \exists x.\,(a, \mathbf{T}) = (\lambda(p, q).\,(p + q, p < q \land q < r))\,x \\
\iff & \exists(p, q).\,(a, \mathbf{T}) = (p + q, p < q \land q < r) \\
\iff & \exists(p, q).\,(a = p + q) \land (p < q \land q < r)
\end{aligned}
$$

Decision procedure for set-theoretic theorems

The following tactics (and rules) expands the goal with definitions of common set operators (and input theorems), and then call `METIS_TAC` to solve it.

```
SET_TAC     : thm list -> tactic
ASM_SET_TAC : thm list -> tactic
SET_RULE    : term -> thm
```

Sample

```
Theorem DISJOINT_RESTRICT_L :
    !s t c. DISJOINT s t ==> DISJOINT (s INTER c) (t INTER c)
Proof
    SET_TAC []
QED
```

Bijections and Countable Sets

More theorems about bijections

$\vdash$ BIJ $f\ s\ t \iff$ INJ $f\ s\ t \land$ SURJ $f\ s\ t$ [BIJ_DEF]

$\vdash$ BIJ $f\ s\ t \iff$
$(\forall x.\ x \in s \Rightarrow f\ x \in t) \land \forall y.\ y \in t \Rightarrow \exists! x.\ x \in s \land f\ x = y$ [BIJ_THM]

$\vdash$ BIJ $f\ s\ t \Rightarrow$
$\exists g.$ BIJ $g\ t\ s \land (\forall x.\ x \in s \Rightarrow (g \circ f)\ x = x) \land$
$\forall x.\ x \in t \Rightarrow (f \circ g)\ x = x$ [BIJ_INV]

$\vdash (\exists f.$ BIJ $f\ \mathcal{U}(:\mathtt{num})\ s) \Rightarrow$ countable s [BIJ_NUM_COUNTABLE]

$\vdash$ countable $s \iff \exists f.$ INJ $f\ s\ \mathcal{U}(:\mathtt{num})$ [countable_def]

$\vdash$ countable $s \iff \exists f.\ \forall x.\ x \in s \Rightarrow \exists n.\ f\ n = x$ [COUNTABLE_ALT]

$\vdash$ countable $c \iff c = \emptyset \lor \exists f.\ c =$ IMAGE $f\ \mathcal{U}(:\mathtt{num})$ [COUNTABLE_ENUM]

NOTE: Bijections btween the same set is called permutations (or permutes): `BIJ f s s` is abbreviated as `f permutes s`.

Maximal and minimal natural numbers in the set

$\vdash$ `MAX_SET` $\emptyset$ `= 0` $\wedge$
$\forall e\ s.$ `FINITE` $s \Rightarrow$ `MAX_SET (`e `INSERT` s`) = MAX` e `(MAX_SET` s`)` [MAX_SET_THM]
$\vdash$ `FINITE` $s \Rightarrow \forall x.\ x \in s \Rightarrow x \leq$ `MAX_SET` s [MAX_SET_PROPERTY]

$\vdash$ `(`$\forall e.$ `MIN_SET` $\{e\}$ `=` e`)` $\wedge$
$\forall e\ s.\ s \neq \emptyset \Rightarrow$ `MIN_SET (`e `INSERT` s`) = MIN` e `(MIN_SET` s`)` [MIN_SET_THM']
$\vdash s \neq \emptyset \Rightarrow \forall x.\ x \in s \Rightarrow$ `MIN_SET` $s \leq x$ [MIN_SET_PROPERTY]
$\vdash (\exists x.\ x \in s) \iff s \neq \emptyset$ [MEMBER_NOT_EMPTY]

NOTE: `MAX_SET` of infinite sets is unspecified; `MIN_SET` of empty set is unspecified.

Maximal and minimal measures on sets

Here a *measure* (`:'a -> num`) is a function from sets to natural numbers. Any set of natural numbers contains a mininal number; In additional, any finite set of natural numbers contains a maximal number.

[NUM_SET_WOP]

$\vdash \forall s.\ (\exists n.\ n \in s) \iff \exists n.\ n \in s \land \forall m.\ m \in s \Rightarrow n \leq m$

[SET_MINIMUM]

$\vdash \forall s\ M.\ (\exists x.\ x \in s) \iff \exists x.\ x \in s \land \forall y.\ y \in s \Rightarrow M\ x \leq M\ y$

[is_measure_maximal_def]

$\vdash \texttt{is\_measure\_maximal}\ m\ s\ x \iff x \in s \land \forall y.\ y \in s \Rightarrow m\ y \leq m\ x$

[FINITE_is_measure_maximal]

$\vdash \forall s.\ \texttt{FINITE}\ s \land s \neq \emptyset \Rightarrow \exists x.\ \texttt{is\_measure\_maximal}\ m\ s\ x$

Sets and lists

From lists to sets

[LIST_TO_SET]

$\vdash$ `set [] =` $\emptyset \wedge$ `set (`h`::`t`) =` h `INSERT set` t

[FINITE_LIST_TO_SET]

$\vdash$ `FINITE (set` l`)`

From (finite) sets to lists

[MEM_SET_TO_LIST]

$\vdash$ `FINITE` $s \Rightarrow \forall x.$ `MEM` x `(SET_TO_LIST` s`)` $\iff x \in s$

[ALL_DISTINCT_SET_TO_LIST]

$\vdash$ `FINITE` $s \Rightarrow$ `ALL_DISTINCT (SET_TO_LIST` s`)`

More related theorems are in `listTheory`.