Interactive Theorem Proving in HOL4

Course 09: number system

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

26 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

Abbreviations: local definitions in the proof (1)

Introduces an abbreviation into a goal

- The tactic `Q.ABBREV_TAC q` (or `qabbrev_tac q`) parses the quotation `q` in the context of the goal to which it is applied.
- The result must be a term of the form `v = e` with `v` a variable.
- The effect of the tactic is to replace the term `e` wherever it occurs in the goal by `v` (or a primed variant of `v` if `v` already occurs in the goal), and to add the assumption `Abbrev(v = e)` to the goal's assumptions.

Substitution in the goal

```
> Q.ABBREV_TAC ‘n = 10‘ ([], ‘‘10 < 9 * 10‘‘);
val it = ([([‘‘Abbrev (n = 10)‘‘], ‘‘n < 9 * n‘‘)], fn):
   goal list * validation
```

Abbreviations: local definitions in the proof (2)

Substitution in the assumptions

```
> Q.ABBREV_TAC `m = n + 2` ([``f (n + 2) < 6``], ``n < 7``);
val it = ([([``Abbrev (m = n + 2)``, ``f m < 6``], ``n < 7``)], fn)
```

Substitution in both goal and assumptions

```
> Q.ABBREV_TAC `u = x ** 32` ([``x ** 32 = f z``],``g (x ** 32 + 6) - 10 < 65``);
val it =
   ([([``Abbrev (u = x ** 32)``, ``u = f z``], ``g (u + 6) - 10 < 65``)], fn)
```

Abbreviate functions

```
> (qabbrev_tac `f = \x. x + 1` >> asm_simp_tac bool_ss [])
      ([], ``3 + 1 = 4 + 1``);
val it = ([([``Abbrev (f = (\x. x + 1))``], ``f 3 = f 4``)], fn)
```

Use (remove) abbreviations

Abbreviations make your proofs and goals clean and controllable.

Expand abbreviations while rewriting

`Abbr` provides the theorem as input to rewriting tactics, e.g. `SIMP_TAC std_ss [Abbr ‘a‘]` or `fs [Abbr ‘a‘]`

```
Abbr : term quotation -> thm
```

Cancel abbreviations (without doing anything else)

```
Q.UNABBREV_TAC  : term quotation -> tactic
qunabbrev_tac   : term quotation -> tactic
qunabbrevl_tac  : term quotation list -> tactic
```

Define abbreviations by pattern matching

```
Q.MATCH_ABBREV_TAC : term quotation -> tactic
qmatch_abbrev_tac  : term quotation -> tactic
```

If the current goal is

```
  ?- (n + 10) * y <= 42315 /\ (!x y. x < y ==> f x < f y)
```

then applying the tactic `Q.MATCH_ABBREV_TAC ‘X <= Y /\ P‘` results in the goal

```
   Abbrev(X = (n + 10) * y),
   Abbrev(Y = 42315),
   Abbrev(P = !x y. x < y ==> f x < f y)
  ---------------------------------------
  ?- X <= Y /\ P
```

Number systems in HOL

Relationship of different number systems

$$\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R} \subseteq \mathbb{C}, \qquad \mathbb{R} \subseteq \overline{\mathbb{R}}\,(= \mathbb{R} \cup \{+\infty, -\infty\})$$

- In traditional mathematics (textbook), the symbols $\subseteq$ literally mean *subset* relationship;
- In higher order logic, the symbols $\subseteq$ are shorthands of *embeddings*.

Embedding natural numbers into integers (`int_of_num`)

`[integerTheory.INT]`

$\vdash$ `((&SUC (`n` :num)) :int) = ((&`n`) :int)` $+$ `(1 :int)`

$\vdash$ `&SUC` n `=` `&`n $+$ 1

Theories of numbers in HOL

numTheory/arithmeticsTheory, integerTheory, ratTheory, realTheory, intrealTheory, `real_of_ratTheory`, complexTheory, extrealTheory, ...

Overloaded arithmetic operators

Common arithmetic operators like $+$, $-$, $\cdot$, $/$ are overloaded across different number systems:

operator	:num	:int	:rat	:real	:extreal
+	ADD	int_add	rat_add	real_add	extreal_add
-	SUB	int_sub	rat_sub	real_sub	extreal_sub
~	-	int_neg	rat_ainv	real_neg	extreal_ainv
*	MULT	int_mul	rat_mul	real_mul	extreal_mul
/	DIV	int_div	rat_div	real_div	extreal_div
inv	-	-	rat_minv	real_inv	extreal_inv
**	EXP	int_exp	rat_expn	real_pow (pow)	extreal_pow (pow)
<	<	int_lt	rat_lt	real_lt	extreal_lt
<=	<=	int_le	rat_le	real_le	extreal_le
&	-	int_of_num	rat_of_num	real_of_num	extreal_of_num

For real numbers (only) in HOL, $\vdash 0^{-1} = 0$ and thus $\vdash \forall x.\, x/0 = 0$ (division-by-zero).

Operators on set of real numbers

Σ and Π (for finite sets only)

[real_sigmaTheory.REAL_SUM_IMAGE_THM]

$\vdash \sum f\ \emptyset = 0\ \land$
$\quad \forall e\ s.\ \mathtt{FINITE}\ s \Rightarrow \sum f\ (e\ \mathtt{INSERT}\ s) = f\ e + \sum f\ (s\ \mathtt{DELETE}\ e)$

[real_sigmaTheory.REAL_PROD_IMAGE_THM]

$\vdash \prod f\ \emptyset = 1\ \land$
$\quad \forall e\ s.\ \mathtt{FINITE}\ s \Rightarrow \prod f\ (e\ \mathtt{INSERT}\ s) = f\ e \times \prod f\ (s\ \mathtt{DELETE}\ e)$

Note: In HOL, $\Sigma_{i=0}^{n} f_i$ is usually represented by $\Sigma\, f\, \{m \mid m < n\}$ (count n).

sup and inf

$\vdash \mathtt{sup}\ P = \varepsilon s.\ \forall y.\ (\exists x.\ P\ x \land y < x) \iff y < s$ [realTheory.sup]

$\vdash \mathtt{inf}\ p = -\mathtt{sup}\ (\lambda r.\ p\ (-r))$ [realTheory.inf_def]

Note: For real numbers, $\sup \mathbb{R}$ and $\inf \emptyset$ do not exist.

Decision procedures

Natural numbers

```
numLib.ARITH_CONV  : conv
numLib.ARITH_PROVE : term -> thm
numLib.ARITH_TAC   : tactic
numLib.DECIDE      : term -> thm
numLib.DECIDE_TAC  : tactic
```

Integers (Presburg decision procedure)

```
intLib.ARITH_CONV  : conv
intLib.ARITH_PROVE : term -> thm
intLib.ARITH_TAC   : tactic
```

Real numbers (semiring decision procedure)

```
realLib.REAL_ARITH     : term -> thm
realLib.REAL_ARITH_TAC : tactic
```