Interactive Theorem Proving in HOL4

Course 03: Higher Order Logic

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

Australian
National
University

15 August 2024

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

Derived Inference Rules (Drules)

Drules

- Drules are SML functions implemented by primitive inference rules (and other drules);
- By using derivation rules (and primitive inference; rules), one is already using the theorem prover to prove new theorems;
- SML programs calling drules (on terms and theorems) to build new theorems are kind of "formal proofs" (in *forward style*).

Adding an assumption

```
ADD_ASSUM : term -> thm -> thm
```

$$\frac{\Gamma \vdash t}{\Gamma, t' \vdash t}\ \texttt{ADD\_ASSUM}\ t'\ (\Gamma \vdash t)$$

```
> val _ = show_assums := true;
> EXCLUDED_MIDDLE;
val it =  [] |- !t. t \/ ~t: thm
> ADD_ASSUM ``P:bool`` EXCLUDED_MIDDLE;
val it =  [P] |- !t. t \/ ~t: thm
```

1. $t' \vdash t'$ [ASSUME]
2. $\Gamma \vdash t$ [Hypothesis]
3. $\Gamma \vdash t' \Rightarrow t$ [DISCH 2]
4. $\Gamma, t' \vdash t$ [MP 1,3]

Undischarging

`UNDISCH : thm -> thm`

$$\frac{\Gamma \vdash t_1 \Rightarrow t_2}{\Gamma, t_1 \vdash t_2}\ \texttt{UNDISCH}\ (\Gamma \vdash t_1 \Rightarrow t_2)$$

```
> load "realTheory"; open realTheory;
> POW_0';
val it =  [] |- 0 < n ==> 0 pow n = 0: thm
> UNDISCH POW_0';
val it =  [0 < n] |- 0 pow n = 0: thm
```

1. $t_1 \vdash t_1$ [ASSUME]
2. $\Gamma \vdash t_1 \Rightarrow t_2$ [Hypothesis]
3. $\Gamma, t_1 \vdash t_2$ [MP 2,1]

Symmetry of equality

`SYM : thm -> thm`

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_2 = t_1}\ \texttt{SYM}\ (\Gamma \vdash t_1 = t_2)$$

```
> realTheory.ONE_MINUS_HALF;
val it =  [] |- 1 - 1 / 2 = 1 / 2: thm
> SYM realTheory.ONE_MINUS_HALF;
val it =  [] |- 1 / 2 = 1 - 1 / 2: thm
```

1. $\Gamma \vdash t_1 = t_2$ [Hypothesis]
2. $\vdash t_1 = t_1$ [REFL]
3. $\Gamma \vdash t_2 = t_1$ [SUBST 1,2]

Application of a term to a theorem

`AP_TERM : term -> thm -> thm`

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t\ t_1 = t\ t_2}\ \texttt{AP\_TERM}\ t\ (\Gamma \vdash t_1 = t_2)$$

```
> load "extreal_baseTheory";
> extreal_baseTheory.half_cancel;
val it =  [] |- 2 * (1 / 2) = 1: thm
> AP_TERM ``sqrt :extreal -> extreal`` it;
val it =  [] |- sqrt (2 * (1 / 2)) = sqrt 1: thm
> AP_TERM ``sqrt :real -> real`` realTheory.ONE_MINUS_HALF;
val it =  [] |- sqrt (1 - 1 / 2) = sqrt (1 / 2): thm
```

1. $\Gamma \vdash t_1 = t_2$ [Hypothesis]
2. $\vdash t\ t_1 = t\ t_1$ [REFL]
3. $\Gamma \vdash t\ t_1 = t\ t_2$ [SUBST 1,2]

Application of a theorem to a term

`AP_THM : thm -> term -> thm`

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_1\, t = t_2\, t}\ \texttt{AP\_THM}\ (\Gamma \vdash t_1 = t_2)\ t$$

```
> load "combinTheory"; open combinTheory;
> val _ = (show_types := true; show_assums := false);
> K_DEF;
val it = |- (K :'a -> 'b -> 'a) = (\(x :'a) (y :'b). x): thm
> AP_THM K_DEF ``x:'a``;
val it = |- (K (x :'a) :'b -> 'a) = (\(x :'a) (y :'b). x) x: thm
```

1. $\Gamma \vdash t_1 = t_2$ [Hypothesis]
2. $\vdash t_1\, t = t_1\, t$ [REFL]
3. $\Gamma \vdash t_1\, t = t_2\, t$ [SUBST 1,2]

Modus Pones for equality

```
EQ_MP : thm -> thm -> thm
```

$$\frac{\Gamma_1 \vdash t_1 = t_2 \qquad \Gamma_2 \vdash t_1}{\Gamma_1 \cup \Gamma_2 \vdash t_2}\ \texttt{EQ\_MP}\ (\Gamma_1 \vdash t_1 = t_2)\ (\Gamma_2 \vdash t_1)$$

```
> T_DEF;
val it = |- T <=> (\x. x) = (\x. x): thm
> TRUTH;
val it = |- T: thm
> EQ_MP T_DEF TRUTH;
val it = |- (\x. x) = (\x. x): thm
```

1. $\Gamma_1 \vdash t_1 = t_2$ [Hypothesis]
2. $\Gamma_2 \vdash t_1$ [Hypothesis]
3. $\Gamma_1 \cup \Gamma_2 \vdash t_2$ [SUBST 1,2]

Transitivity of equations

`TRANS : thm -> thm -> thm`

$$\frac{\Gamma_1 \vdash t_1 = t_2 \qquad \Gamma_2 \vdash t_2 = t_3}{\Gamma_1 \cup \Gamma_2 \vdash t_1 = t_3}\ \texttt{TRANS}\ (\Gamma_1 \vdash t_1 = t_2)\ (\Gamma_2 \vdash t_2 = t_3)$$

```
> load "extreal_baseTheory"; open extreal_baseTheory;
> half_cancel;
val it = |- 2 * (1 / 2) = 1: thm
> third_cancel;
val it = |- 3 * (1 / 3) = 1: thm
> TRANS half_cancel (SYM third_cancel);
val it = |- 2 * (1 / 2) = 3 * (1 / 3): thm
```


Implication from equality

`EQ_IMP_RULE : thm -> thm * thm`

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_1 \Rightarrow t_2 \qquad \Gamma \vdash t_2 \Rightarrow t_1}\ \texttt{EQ\_IMP\_RULE}\ (\Gamma_1 \vdash t_1 = t_2)$$

```
> F_DEF;
val it = |- F <=> !t. t: thm
> EQ_IMP_RULE F_DEF;
val it = (|- F ==> !t. t, |- (!t. t) ==> F): thm * thm
```

1. $\Gamma \vdash t_1 \Leftrightarrow t_2$ [Hypothesis]
2. $t_1 \vdash t_1$ [ASSUME]
3. $\Gamma, t_1 \vdash t_2$ [EQ_MP 1,2]
4. $\Gamma \vdash t_1 \Rightarrow t_2$ [DISCH 3]
5. $\Gamma \vdash t_2 = t_1$ [SYM 1]

6. $t_2 \vdash t_2$ [ASSUME]
7. $\Gamma, t_2 \vdash t_1$ [EQ_MP 5,6]
8. $\Gamma \vdash t_2 \Rightarrow t_1$ [DISCH 7]
9. $\Gamma \vdash t_1 \Rightarrow t_2$ and $\Gamma \vdash t_2 \Rightarrow t_1$ [4,8]

T-introduction

```
TRUTH : thm

    ⊢ T
```

1. $\vdash \mathtt{T} = ((\lambda x.\, x) = (\lambda x.\, x))$ [Definition of T]
2. $\vdash ((\lambda x.\, x) = (\lambda x.\, x)) = \mathtt{T}$ [SYM 1]
3. $\vdash (\lambda x.\, x) = (\lambda x.\, x)$ [REFL]
4. $\vdash \mathtt{T}$ [EQ_MP 2,3]

Equality-with-T elimination

`EQT_ELIM : thm -> thm`

$$\frac{\Gamma \vdash t = \mathtt{T}}{\Gamma \vdash t}\ \texttt{EQT\_ELIM}\ (\Gamma \vdash t = \mathtt{T})$$

```
> RES_FORALL_TRUE;
val it = |- (!x::P. T) <=> T: thm
> EQT_ELIM RES_FORALL_TRUE;
val it = |- !x::P. T: thm
```

1. $\Gamma \vdash t = \mathtt{T}$ [Hypothesis]
2. $\Gamma \vdash \mathtt{T} = t$ [SYM 1]
3. $\vdash \mathtt{T}$ [TRUTH]
4. $\Gamma \vdash t$ [EQ_MP 2,3]

Equality-with-T introduction

```
EQT_INTRO : thm -> thm
```

$$\frac{\Gamma \vdash t}{\Gamma \vdash t = \mathtt{T}}\ \mathtt{EQT\_INTRO}\ (\Gamma \vdash t)$$

```
> EQT_INTRO INFINITY_AX; (* or any other theorem *)
val it = |- (?f. ONE_ONE f /\ ~ONTO f) <=> T: thm
```

(The implementation is a bit complex by primitive inferences, omitted here.)

Specialization ($\forall$-elimination)

```
SPEC : term -> thm -> thm
```

$$\frac{\Gamma \vdash \forall x.\, t}{\Gamma \vdash t[x \mapsto t']}\ \texttt{SPEC}\ t'\ (\Gamma \vdash \forall x.\, t)$$

```
> (arithmeticTheory.ADD1, SPEC ``2:num`` arithmeticTheory.ADD1);
val it = (|- !m. SUC m = m + 1, |- SUC 2 = 2 + 1): thm * thm
```

1. $\vdash \forall = (\lambda P.\, P = (\lambda x.\, \mathtt{T}))$ [INST_TYPE applied to the definition of $\forall$]
2. $\Gamma \vdash \forall\,(\lambda x.\, t)$ [Hypothesis]
3. $\Gamma \vdash (\lambda P.\, P = (\lambda x.\, \mathtt{T}))(\lambda x.\, t)$ [SUBST 1,2]
4. $\vdash (\lambda P.\, P = (\lambda x.\, \mathtt{T}))(\lambda x.\, t) = ((\lambda x.\, t) = (\lambda x.\, \mathtt{T}))$ [BETA_CONV]
5. $\Gamma \vdash (\lambda x.\, t) = (\lambda x.\, \mathtt{T})$ [EQ_MP 4,3]
6. $\Gamma \vdash (\lambda x.\, t)\, t' = (\lambda x.\, \mathtt{T})\, t'$ [AP_THM 5]
7. $(\lambda x.\, t)\, t' = t[x \mapsto t']$ [BETA_CONV]
8. $t[x \mapsto t'] = (\lambda x.\, t)\, t'$ [SYM 7]
9. $\Gamma \vdash t[x \mapsto t'] = (\lambda x.\, \mathtt{T})\, t'$ [TRANS 8,6]
10. $\vdash (\lambda x.\, \mathtt{T})\, t' = \mathtt{T}$ [BETA_CONV]
11. $\Gamma \vdash t[x \mapsto t'] = \mathtt{T}$ [TRANS 9,10]
12. $\Gamma \vdash t[x \mapsto t']$ [EQT_ELIM 11]

Generalization ($\forall$-introduction)

`GEN : term -> thm -> thm`

$$\frac{\Gamma \vdash t}{\Gamma \vdash \forall x.\, t}\ \texttt{GEN}\ x$$

where x is not free in Γ.

```
> (relationTheory.RC_REFL, GEN ``x:'a`` relationTheory.RC_REFL);
val it = (|- RC R x x, |- !x. RC R x x): thm * thm
```

1. $\Gamma \vdash t$ [Hypothesis]
2. $\Gamma \vdash t = \mathtt{T}$ [EQT_INTRO 1]
3. $\Gamma \vdash (\lambda x.\, t) = (\lambda x.\, \mathtt{T})$ [ABS 2]
4. $\vdash \forall (\lambda x.\, t) = \forall (\lambda x.\, t)$ [REFL]
5. $\vdash \forall = (\lambda P.\, P = (\lambda x.\, \mathtt{T}))$ [INST_TYPE applied to the definition of $\forall$]
6. $\vdash \forall (\lambda x.\, t) = (\lambda P.\, P = (\lambda x.\, \mathtt{T}))\,(\lambda x.\, t)$ [SUBST 5,4]
7. $\vdash (\lambda P.\, P = (\lambda x.\, \mathtt{T}))\,(\lambda x.\, t) = ((\lambda x.\, t) = (\lambda x.\, \mathtt{T}))$ [BETA_CONV]
8. $\vdash \forall (\lambda x.\, t) = ((\lambda x.\, t) = (\lambda x.\, \mathtt{T}))$ [TRANS 6,7]
9. $\vdash ((\lambda x.\, t) = (\lambda x.\, \mathtt{T})) = \forall (\lambda x.\, t)$ [SYM 8]
10. $\Gamma \vdash \forall (\lambda x.\, t)$ [EQ_MP 9,3]

More inference rules

- η-conversion (`ETA_CONV : term -> thm`): $\vdash (\lambda x.\, t\, x) = t$
- Extensionality (`EXT : thm -> thm`):

$$\frac{\Gamma \vdash \forall x.\, t_1\, x = t_2\, x}{\Gamma \vdash t_1 = t_2}$$

- $\exists$-introduction (`EXISTS : term * term -> thm -> thm`):

$$\frac{\Gamma \vdash t_1[t_2]}{\Gamma \vdash \exists x.\, t_1[x]}$$

- $\exists$-elimination (`CHOOSE : term * thm -> thm -> thm`):

$$\frac{\Gamma_1 \vdash \exists x.\, t[x] \qquad \Gamma_2, t[v] \vdash t'}{\Gamma_1 \cup \Gamma_2 \vdash t'}$$

- $\wedge$-introduction (`CONJ : thm -> thm -> thm`)
- $\wedge$-elimination (`CONJUNCT1 : thm -> thm`, `CONJUNCT2 : thm -> thm`)

Homework

- Read Chapter 2 (Derived Inference Rules) of *The HOL System DESCRIPTION* and write a SML script to try all of them once with some existing theorems in core theories.[1]
- Implement your own `SPEC` and `GEN` in Standard ML, e.g. `MY_SPEC` and `MY_GEN`.

[1] In HOL4, a command like `help "SPEC"` may show its documentation.