
Mechanizing Theories in Twelf

A Tutorial Course (Part III)

Susmit Sarkar

<http://www.cl.cam.ac.uk/~ss726/twelf>

Recap

- Meta theoretic theorems and proofs
- Realized as total logic programs
- `%mode` declarations
- `%worlds` declarations
- `%total` declarations

Agenda today

- A variety of case studies
- Useful proof techniques

Case Study 1: F^ω

```
[ [ fomega.elf ] ]
```

- Illustration: Subordination
- `Print.subord` gives Twelf view

Recall: Subordination [Virga 99]

- Binary relation \preceq between type family constants
- Reflexive and transitive
- Π type subordination: If $\Pi x:A_1.A_2$ appears in signature, $A_1 \preceq A_2$
- Index subordination: If $a:\Pi x_1:A_1.\dots\Pi x_n:A_n.\text{type}$ is declared, then $\text{head}(A_i) \preceq a$

Subordination: Intuition

- If $a \preceq b$ then a term of type $a \dots$ can occur as a subterm of $b \dots$
- Recall: Canonical type families are $\Pi x:A_1.A_2$ or $aM_1 \dots M_n$
- Important for adequacy proofs

Case Study 2: Church-Rosser

```
[[ cr.elf ]]
```

- Example: identity lemma for parallel reduction

Identity lemma: try 1

```
identity : {M:term} M => M -> type.  
%mode identity +M -R.
```

```
%worlds () (identity M R).
```

- Abstraction requires us to descend within binder
- What do variables reduce to?

Identity lemma: try 2

```
%block idblock : block {x:term}
                        {eqx:x => x}.
%worlds (idblock) (identity M R).
```

- How do we relate variable reductions?

Identity lemma: correct world

```
%block idblock : block {x:term}
                        {eqx: x => x}
                        {u:identity x eqx}.
%worlds (idblock) (identity M R).
```

- Key technique: theorem case for variable is assumed

Continuing on Church-Rosser

`[[cr.elf]]`

- Calling lemma in a different world
- World subsumption between diamond lemma and strip lemma

Case Study 3: Cut admissibility

```
[[ cutelim.elf ]]
```

- Sequent calculus for intuitionistic logic

Cut admissibility: cases

- Axiom cuts
- Left commutative cuts
- Right commutative cuts
- Principal cuts

Termination argument

- Cut formula goes down
- Or cut formula stays same, but derivations become smaller
- Mixed lexicographic and joint order

Cut admissibility: contd.

```
[[ cutelim.elf ]]
```

- Move to full first-order logic
- Quantifier rules introduce individuals
- Not right rule quantifies over propositions

Non-uniform worlds

- Worlds are regular
- But blocks of different shapes

Case study 4: Linear Logic

`[[linear.elf]]`

- Encoding linear logic
- Higher-order assumptions are intuitionistic
- Want to ensure linearly used

Linearity as a predicate

- Have proof terms
- Predicate of linearity on proof terms
- Typing rules assume and maintain linearity

Substitutions for linearity

- Substitution for linear variable
- With term that is independent of variable
- Results in a linear term

Case study 5: References

- Consider the simply typed calculus with references
- Stores and store typings
- Typing judgment needs store types
- Statement of preservation extended with stores

Typing judgment (buggy)

```
of : store -> exp -> tp -> type.
```

```
t_loc : of S (loc L) (ref T)
      <- find_in_store L S T.
```

```
t_ref : of S (alloc E) (ref T)
      <- of S E T.
```

```
t_abs : of S (lam T1 E) (T1 => T2)
      <- ( {x:exp} of S x T1 -> of S (E x) T2 ).
```

Preservation

preservation :

```
%{ if }% of S E T -> wt_heap S H
-> step H E H' E' ->
%{ then }% store_extends S S'
-> wt_heap S' H'
-> of S' E' T -> type.
```

```
%mode preservation +Q1 +W1 +S1 -P1 -W2 -Q2.
```

Required lemma

```
of_store_extends :  
  %{ if      }%   of S1 E T ->  
                    store_extends S1 S2 ->  
  %{ then }%   of S2 E T -> type.  
%mode of_store_extends +Q1 +P -Q2.
```

Abstraction

```
t_abs : of S (lam T1 E) (T1 => T2)
      <- ( {x:exp} of S x T1 -> of S (E x) T2 ).
```

- Typing assumption of variable
- Assumption in negative position
- Anti-monotonicity of stores??

Comparison with paper proof

- Typing assumption for variable does not mention store
- At use of variable, store is checked
- Mark variable assumptions

Corrected judgments

`var : exp -> tp -> type.`

`t_var :`
 `of S E T`
 `<- var E T.`

`t_abs :`
 `of S (lam T1 E) (T1 => T2)`
 `<- ({x:exp} var x T1 -> of S (E x) T2).`

Proof

`[[refs.elf]]`

- Do not extend expressions
- Substitution lemma not for free!

Thank You!



<http://www.cs.cmu.edu/~twelf>