ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Dipl.-Inf. Achim D. Brucker
Dr. Burkhart Wolff

Computer-supported Modeling and Reasoning

http://www.infsec.ethz.ch/education/permanent/csmr/

(rev. 16814)

Submission date: –

PL in LF

In this exercise, we will use a very powerful meta-logic, introduced under the name *LF* ("logical framework"). Its purpose is to represent not only the syntax of propositional logics (PL), but the deductive system in form of its natural deduction system. As a consequence, we will deepen our understanding of notions like *proof objects* and the propositions-as-types principle.

By encoding PL in LF, we also give an intuition into Isabelle and its character as logical framework itself—at the end, Isabelle's built-in logic *Pure* is used to encode LF with the same techniques as we are studying PL in LF.

1 Background

1.1 Revisiting LF

We briefly revisit the LF system as presented in the lecture. LF is defined as a λ-calculus with dependent types; these were represented by a several mutual recursive judgments formalizing *signatures* Σ and *contexts* Γ.

LF

The basic theory `http://www.infsec.ethz.ch/education/permanent/csmr/material/LF.thy` contains a *shallow embedding* of the raw terms—also called: pseudo terms—of the λ-calculus (i.e. substitution and generation of free

LF.thy

shallow embedding

variables is done by *Pure*). However, the type-system is represented by axioms that define the notion of signature and context. As in previous exercises, we make no statement about the logical consistency of our presentation.

1.2 Signatures and Contexts

signature Generally, a *signature* specifies the "constant symbols" (as opposed to variables). A signature Σ is a sequence of pairs of the form $c : \tau$, where c is a constant symbol and τ is a type.

context A *context* specifies the types of the variables used in an expression. A context Γ is a sequence of pairs of the form $x : A$, where $x \in Var$ and A is a raw term.

The axioms for signatures and contexts define inductively the subset of *valid* signatures and contexts.

1.3 The judgments of LF

Valid signatures and contexts are defined via three (mutually recursive) kinds of judgments:

1. judgments stating that a signature is valid, $\vdash_{sig} \Sigma$;
2. judgments stating that a context is valid, $\vdash_{con} \Gamma$;
3. judgments stating that a term has a certain type; this is a relation between a signature Σ, a context Γ and an expression of the form $t : A$, written $\Gamma \vdash_\Sigma t : A$.

Note, however, that our implementation of LF in Isabelle differs from the presentation in lecture in that there is no Σ and $\vdash_\Sigma$. Statements for them were simulated by constant declarations and suitable axioms.

The judgments in LF are of the form $x_1 : X_1 \ \ldots \ x_n : X_n \ \vdash x : X$. An example for a judgment is `x:o y:o |- x:o`.

The following table shows how the various syntactical entities of LF are written in `LF.thy`:

LF	`LF.thy`
$\Pi x^A.\, b$	Prod(A, λx.B) or Pi x:A. B
$A \to B$	A->B
$\lambda x^A.\, b$	Abs(A, λx.B) or Lam x:A. B
$F(a)$ (application)	F^A

The notations Prod(A, λx.B) and Abs(A, λx.B) may be parsed and printed alternatively by Isabelle. There are also some differences between the LF presentation in the lecture and the way the *rules* are encoded in Isabelle:

- There is no assumption rule, since signatures are mimicked by contexts and by theory extensions.
- The hypothesis rule requires that the type assignment to be proven is the first in the context (which is implicitly assumed to be a set). In Isabelle/LF, the context is more a list-like structure which makes the introduction of a weakening-rule necessary.

2 Exercises

2.1 Exercise 24

Prove three of the following judgments in LF. To learn more, you might want to try and guess the instantiation of the metavariable in advance:

1. $i : \mathit{Type} \vdash \Pi x^{i}.\ \mathit{Type} : ?T$
2. $A : \mathit{Type}, B : \mathit{Type} \vdash A \to B : ?T$
3. $A : \mathit{Type}, B : \mathit{Type} \vdash \lambda x^{A}.\ A \to B : ?T$
4. $f : \Pi x^{A}.\ B, a : A \vdash f(a) : ?T$ (note how Isabelle displays $\Pi x^{A}.\ B$!)
5. $A : \mathit{Type}, P : A \to \mathit{Type}, a : A \vdash P(a) : ?T$
6. $A : \mathit{Type}, P : A \to \mathit{Type} \vdash \lambda a^{A}.\ \lambda b^{P(a)}.\ b : ?T$

2.2 Exercise 25

Encode syntax and deductive system of propositional logic (PL) and call the resulting theory PL_in_LF. The cases for and and or are sufficient.

Example for the syntax:

```
consts
  "o"          :: "term"
  "imp"        :: "term"

axioms
  o_def:       "G|- o:Type"
  imp_def:     "G|- imp: o->o->o"
```

Example for the deductive system:

```
consts
  "pr"        :: "term"

  "impl"      :: "term"

axioms
  pr_def :    "G|- pr: o->Type"

  impl_def :  "G|- impl:Pi A:o. Pi B:o. (pr^A->pr^B)->pr ^(imp^A^B)"
```

Do not forget the impE-rule!

2.3 Exercise 26

Prove in PL_in_LF that $\vdash \Pi x^o.\, \Pi y^o.\, pr\ (imp\ x\ y) \to (pr\ x) \to (pr\ y) : Type$.

2.4 Exercise 27

Prove one of the following propositions in PL_in_LF:

1. $a \to a$
2. $a \to b \to a$

Hints:

1. One states that a proof for the goal is a term of type $pr(imp\ a\ a)$, and gives a proof object for it, i.e. one states

   $$a : o \vdash ?t : pr(imp\ a\ a)$$

   for an appropriate, given t and proves this statement.

2. Alternatively, one *synthesizes* the $?t$ through the meta-level proof. Since the unifications for the `application`-rule are highly ambiguous (Isabelle may even be unable to find existing unifiers!), you will have to make tricky explicit instantiations. An (unsafe and incomplete) alternative is to use `back` until Isabelle has found the right unifier.

3. The proof object for the second exercise is:

   impl^a^(imp^b^a)^ (Lam x: pr^a. impl^b^a^(Lam xa:pr^b. x))

   The proof is difficult.

2.5 Exercise 28

Prove one of the following propositions in PL_in_LF:

1. $a \wedge b \rightarrow a$
2. $a \rightarrow b \vee a$