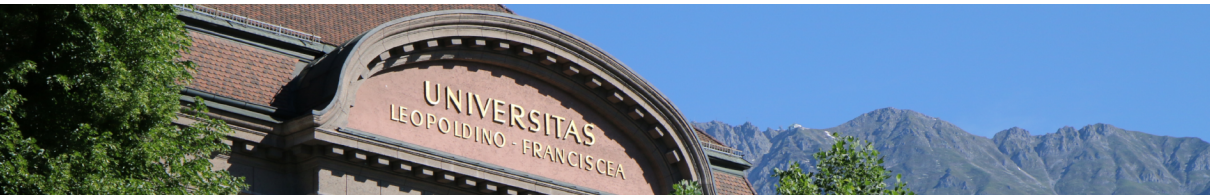
universität innsbruck



Interactive Theorem Proving

Lecture & Exercises Week 8

Cezary Kaliszyk

Summary

Previous Lecture

- Dependent types
- λ_P
- Curry Howard for λ_P

Today

- Tactics
- Lambda 2

Exercises

- Give λ_P derivations whose types are the conclusions of:
  1. $(\forall y.Qy \to Py) \to Qa \to Pa$
  2. $(\forall x.\forall y.Px \to Py) \to Pa \to \forall x.Px$
  3. $(\forall x.Px \to A) \to (A \to \forall x.\forall y.Qyx) \to Pa \to Qbb$

Exercises

- Give λ_P derivations whose types are the conclusions of:
  1. $(\forall y.Qy \to Py) \to Qa \to Pa$
  2. $(\forall x.\forall y.Px \to Py) \to Pa \to \forall x.Px$
  3. $(\forall x.Px \to A) \to (A \to \forall x.\forall y.Qyx) \to Pa \to Qbb$
- In the last PS we have looked at backwards proof
  - Find the definition of the type `tactic` defined in HOL Light.
  - Prove in HOL-Light using tactics:

```
a)  A \/ B ==> B \/ A

b)  (A ==> B ==> C) ==> (A ==> B) ==> A ==> C

(optional)
    (!x. P x ==> Q x) ==> ((?y. Q y) ==> P a) ==> Q b ==> Q (a : A)
```

Using non-automated tactics. Hint: look for the tactics in `VERY_QUICK_REFERENCE.txt`

Correspondence with first order logic

FOL corresponds to a fairly weak fragment of λ_P

- Only one type variable 0 (constant, type of individuals)
- All kinds are of the form $0 \Rightarrow \ldots \Rightarrow 0 \Rightarrow \star$
- Function symbols are distinguished object variables $0 \to \ldots \to 0$
- Constants are distinguished variables of type 0
- Other declarations may only be of the form $x : 0$

Correspondence with first order logic

FOL corresponds to a fairly weak fragment of λ_P

- Only one type variable 0 (constant, type of individuals)
- All kinds are of the form $0 \Rightarrow \ldots \Rightarrow 0 \Rightarrow \star$
- Function symbols are distinguished object variables $0 \to \ldots \to 0$
- Constants are distinguished variables of type 0
- Other declarations may only be of the form $x : 0$

Proof correspondence

- Proof by generalization
  - abstraction $\lambda x : 0.M^{\varphi}$
- Proof by MP
  - application $M^{\forall x:0\varphi}N^{0}$

More properties of λ_P

- Strong Normalization
- Subject Reduction
- In Church style: Type checking and type inference decidable in PTIME.
  - In Curry style: Already type checking is undecidable [*Dowek'93*]
- Type inhabitation in λ_P is undecidable
- Curry-Howard isomorphism of a weak fragment of λ_P and IFOL.
  - Still we have only implications
- Possible to add other connectives [*Urzyczyn,Geuvers*]

Automaton with two counters

$$A =< Q, q_0, q_f, \delta >$$

- Q - finite set of states
- q_0 - initial state
- q_f - final state
- δ - transition function
  - Domain: $Q - q_f$
  - $\delta(q)$ is one of the three forms, for $i = 1$ or 2:
    - $c_i := c_i + 1;$ goto p
    - $c_i := c_i - 1;$ goto p
    - if $c_i = 0$ then goto p else goto q
- Configuration of the automaton: $C =< q, n_1, n_2 >$

Automaton with two counters

Definition (accept)

An automaton accepts a configuration C if there exists a sequence:

$$C \to C_1 \to C_2 \to ... \to C_n$$

Where C_n is a configuration with a final state.

Halting problem

Does a given automaton A accept the given configuration C?

Theorems

1. The halting problem is undecidable.
2. There exists an A, st. the halting problem is undecidable with A fixed.

Helper definitions

- $$\tau_n(\alpha) = \alpha^{n-1} \to \alpha$$
- $$\sigma(\alpha) = (\alpha \to \alpha) \to \alpha$$
- Lemma: For any $m \neq n$, the types cannot be unified. For any α, β and any substitution S: $S(\tau_m(\alpha)) \neq S(\tau_n(\beta))$
- Lemma: The type $\sigma(\alpha)$ cannot be unified with any $\tau_n(\beta)$. For any S, α, β, n. $S(\tau_n(\alpha)) \neq S(\sigma(\beta))$
- States are numbered 4...F. $q_0 = 4$ and $q_f = F$.

Encoding the configuration

- The code of the number n is any formula of the form:

$$\tau_2(\alpha_1) \to ... \to \tau_2(\alpha_n) \to \tau_3(\beta)$$

- The code of a state q is any formula $\tau_q(\alpha)$.
- The code of $C =< q, n_1, n_2 >$ is any formula:

$$code(q) \to code(n_1) \to code(n_2) \to \sigma(\xi)$$

Encoding the state transition function

- For $\delta(q) = c++; \textit{goto } p$

$$(\tau_p(\alpha) \to (\tau_2(\epsilon) \to \beta) \to \gamma \to \sigma(\xi)) \to (\tau_q(\alpha) \to \beta \to \gamma \to \sigma(\xi))$$

- For $\delta(q) = c--; \textit{goto } p$

$$(\tau_p(\alpha) \to \beta \to \gamma \to \sigma(\xi)) \to (\tau_q(\alpha) \to (\tau_2(\epsilon) \to \beta) \to \gamma \to \sigma(\xi))$$

- For $\delta(q) = \textit{if } c_1 = 0 \textit{ then goto } p \textit{ else goto } r$ two formulas:

$$(\tau_p(\alpha) \to \tau_3(\beta) \to \gamma \to \sigma(\xi)) \to (\tau_q(\alpha) \to \tau_3(\beta) \to \gamma \to \sigma(\xi))$$

$$(\tau_r(\alpha) \to (\tau_2(\epsilon) \to \beta) \to \gamma \to \sigma(\xi)) \to (\tau_q(\alpha) \to (\tau_2(\epsilon) \to \beta) \to \gamma \to \sigma(\xi))$$

- Finally

$$\tau_F(\alpha) \to \beta \to \gamma \to \sigma(\xi)$$

Main result

For any configuration C and any formula φ, which codes C, the following are equivalent:

- Automaton A accepts the configuration C.
- The formula φ of automaton A has a proof in the predicate calculus

Proof:

- $(\downarrow)$: Induction w.r.t. the computation length.
- $(\uparrow)$: Induction w.r.t. the length of the proof.

Curry Howard again

1st order propositional logic	$\leftrightarrow$	simple type theory e.g. $\lambda_{\rightarrow}$
1st order predicate logic	$\leftrightarrow$	type theory with dependent types e.g. λ_P
2nd order propositional logic	$\leftrightarrow$	polymorphic type theory e.g. λ_2

Second Order propositional logic (syntax)

- $a\ b\ c\ \ldots$
- $A \to B$
- $\bot$
- $\top$
- $\neg A$
- $A \wedge B$
- $A \vee B$
- $\forall a\,.\,A$
- $\exists a\,.\,A$

Example

$$\forall \alpha . \alpha \to \alpha$$

Second Order propositional logic (rules)

- $I[x]\rightarrow$, $E\rightarrow$
- $I\top$, $E\bot$
- $I[x]\neg$, $E\neg$
- $I\wedge$, $E\wedge_{\{1,2\}}$
- $I\vee_{\{1,2\}}$, $E\vee$
- $I\forall$, $E\forall$
- $I\exists$, $E\exists$

New rules: Universal quantification

Universal introduction

$$\frac{\begin{array}{c}\vdots\\ A\end{array}}{\forall a.\, A}\ I\forall$$

a cannot be free variable in any open assumption!

Universal elimination

$$\frac{\begin{array}{c}\vdots\\ \forall a.\, A\end{array}}{A[a := B]}\ E\forall$$

New rules: Existential quantification

Existential intro

$$\frac{\begin{array}{c}\vdots\\ A[a := B]\end{array}}{\exists a.\, A}\ I\exists$$

Existential elimination

$$\frac{\begin{array}{c}\vdots\\ \exists a.\, A\end{array} \qquad \begin{array}{c}\vdots\\ \forall a.\, (A \to B)\end{array}}{B}\ E\exists$$

a cannot be free in B!

Exercise

Prove in second-order propositional logic two of the following:

- $(\forall b.\, b) \to a$
- $a \to \forall b.\, ((a \to b) \to b)$
- $(\exists b.\, a) \to a$
- $\exists b.((a \to b) \vee (b \to a))$
- $\forall a.\, \forall b.\, ((a \to b) \vee (b \to a))$
- $\forall a.\, (a \vee \neg a)$
- $a \to \forall a.\, a$
- $(\exists a.\, a) \to a$

(you can choose the ones corresponding to the last two digits of your matrikelnummer mod 8. If the two are the same consider additionally the previous digit).

The “order”

First order

Object

Second order

- Set of objects
- Predicate on objects
- Function from objects to objects.

Third order

- Set of second order objects
- Predicate on predicates of objects
- Function from second order objects to ...

Etc

Example

Induction on nat is a second order predicate logic formula

$$\forall a.\, \big(a(0) \to (\forall m.\, a(m) \to a(S(m))) \to \forall n.\, a(n)\big)$$

m, n	1st order variables
0	1st order constant
a	2nd order variable
S	2nd order constant

Syntax of λ_2

$$*, \Box$$

$$x, y, z, \ldots$$

$$MN$$

$$\lambda x : M. N$$

$$\Pi x : M. N$$

Rules of λ_2 (1/2)

$$\frac{}{\vdash * : \Box}\ \text{axiom}$$

$$\frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}\ \text{application}$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}\ \text{abstraction}$$

$$\frac{\Gamma \vdash A : s \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A.B : *}\ \text{product}$$

(where s is $\star$ or $\Box$)

Rules of λ_2 (2/2)

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\ \text{weakening}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ \text{variable}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}\ \text{conversion}$$

with $B =_\beta B'$

The product rules

all systems

$$\frac{\Gamma \vdash A : *\qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A.B : *}$$

only in λP

$$\frac{\Gamma \vdash A : *\qquad \Gamma, x : A \vdash B : \Box}{\Gamma \vdash \Pi x : A.B : \Box}$$

$$\texttt{nat} \to *$$

only in $\lambda 2$

$$\frac{\Gamma \vdash A : \Box\qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A.B : *}$$

$$\Pi a : *.\, a \to a$$

Exercises (Optional)

For one of the following λ_2 find a term, or give a model that would say why this is not possible:

1. $(\forall b.\, b) \to a$
2. $a \to \forall b.\, ((a \to b) \to b)$
3. $(\exists b.\, a) \to a$
4. $\exists b.((a \to b) \vee (b \to a))$
5. $\forall a.\, \forall b.\, ((a \to b) \vee (b \to a))$
6. $\forall a.\, (a \vee \neg a)$
7. $a \to \forall a.\, a$
8. $(\exists a.\, a) \to a$
9. $\forall a.(a \to \forall b.(b \to (a \wedge b)))$
10. $\forall b.(\forall a.((a \to b) \wedge (b \to a)))$
11. $(\star)$ $\exists a.\exists b.((a \vee b) \wedge (\neg a \vee \neg b))$

Hint: We have not formally defined what a “model” in λ_2 is, so presenting such a notion is a good way to start the exercise for the non-provable theorems.