

Interactive Theorem Proving

Week 7

Cezary Kaliszyk

April 28, 2015



Summary

So far

Proof Assistants, HOL Light, λ_{\rightarrow} , Gentzen-style, Tactics

- Properties of λ_{\rightarrow}
- BHK interpretation and λ -cube again
- Dependent types
- λ_P
- HOL Light more advanced tactics (...)

Today

- How hard is λ_P
- Second order logic
- Order of variables
- λ_2

How hard is λ_P

Reasoning in Predicate Calculus

Does a given predicate logic formula follow from the given axioms (axiom schemes)?

Theorem (Linial, Post, 1949)

Reasoning in predicate logic is undecidable

Automaton with two counters

$$A = \langle Q, q_0, q_f, \delta \rangle$$

- 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: $C = \langle q, n_1, n_2 \rangle$

Automaton with two counters

Definition (accept)

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

$$C \rightarrow C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n$$

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

- $$\tau_n(\alpha) = \alpha^{n-1} \rightarrow \alpha$$
- $$\sigma(\alpha) = (\alpha \rightarrow \alpha) \rightarrow \alpha$$
- 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 $4 \dots F$. $q_0 = 4$ and $q_f = F$.

Encoding the configuration

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

$$\tau_2(\alpha_1) \rightarrow \dots \rightarrow \tau_2(\alpha_n) \rightarrow \tau_3(\beta)$$

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

$$\text{code}(q) \rightarrow \text{code}(n_1) \rightarrow \text{code}(n_2) \rightarrow \sigma(\xi)$$

Encoding the state transition function

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

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

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

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

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

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

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

- Finally

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

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. λ_{\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)

- $a b c \dots$
- $A \rightarrow B$
- \perp
- \top
- $\neg A$
- $A \wedge B$
- $A \vee B$
- $\forall a. A$
- $\exists a. A$

Example

$$\forall \alpha. \alpha \rightarrow \alpha$$

Second Order propositional logic (rules)

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

New rules: Universal quantification

Universal introduction

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

a cannot be free variable in any open assumption!

Universal elimination

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

New rules: Existential quantification

Existential intro

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

Existential elimination

$$\frac{\begin{array}{c} \vdots \\ \exists a. A \end{array} \quad \begin{array}{c} \vdots \\ \forall a. (A \rightarrow B) \end{array}}{B} \text{E}\exists$$

a cannot be free in *B*!

Examples

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

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

$$\forall a. (a(0) \rightarrow (\forall m. a(m) \rightarrow a(S(m)))) \rightarrow \forall n. a(n)$$

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

Syntax of λ_2

$*$, \square

x, y, z, \dots

MN

$\lambda x : M. N$

$\Pi x : M. N$

Rules of λ_2 (1/2)

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

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

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

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

(where s is $*$ or \square)

Rules of λ_2 (2/2)

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

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

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

with $B =_{\beta} B'$

The product rules

all systems

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

only in λP

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

$\text{nat} \rightarrow *$

only in $\lambda 2$

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

$\Pi a : *. a \rightarrow a$

Summary

Today

- How hard is λ_P
- Second order logic
- Order of variables
- λ_2

Next time

- λ_C
- Set Theory Introduction