Computational Logic

Interactive Theorem Proving

Week 6

Cezary Kaliszyk

April 26, 2013

Summary

So far

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

- Properties of $\lambda_{\rightarrow}$
- 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
- HOL Light library

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

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

- 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:

$$C \to C_1 \to C_2 \to \ldots \to C_n$$

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 $4 \ldots F$. $q_0 = 4$ 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 has a proof in the predicate calculus of automaton A

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!

Examples

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

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)

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

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

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

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

Rules of λ_2 (1/2)

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

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

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

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 $\lambda 2$

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

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

Summary

Today

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

Next time

- Set Theory Introduction