A Small Framework for Proof Checking

PAAR 2008, Sydney, 11 August 2008

Hans de Nivelle & Piotr Witkowski

University of Wrocław, Poland.

Introduction

I describe a small proof checker.

It is based on natural deduction + first-order automated theorem proving.

Every theorem prover that speaks TPTP can be used.

The calculus contains higher-order logic, but the higher order is kept away from the theorem prover.

We acknowledge that we learnt quite a lot from Mizar, but we want to use standard logic, and to be transparant.

Our long term aims are

- To obtain a tool that can be used in teaching. (introductory logic courses and more advanced courses on verification)
- To do some medium sized verifications in it.
- To obtain test problems for Geo, to find out what features it should have.

Examples

$\wedge$-intro and $\wedge$-elim can be proven as follows:

ASSUME PREDICATE p: 0, q: 0
IN

andintro : PROVE $p \to q \to p \wedge q$

andelim1 : PROVE $p \wedge q \to p$

andelim2 : PROVE $p \wedge q \to q$

END

$\wedge$-intro can be used as follows:

PROVE $F_1 \wedge F_2$
FROM andintro { PRED $p := F_1$, PRED $q := F_2$ }, f1, f2

(assuming that f1 refers to F_1, and f2 refers to F_2. Of course, no self respecting theorem prover needs andintro)

Examples (2)

Or as follows:

ASSUME PREDICATE p: 0, q: 0

IN

    ASSUME FORMULA a1: p, a2: q IN

        andintro: PROVE $p \wedge q$ FROM a1, a2

    END

    ASSUME FORMULA a: $p \wedge q$ IN

        andelim1 : PROVE p FROM a

        andelim2 : PROVE q FROM a

    END

END

Examples (3)

We prove $\exists$-elim:

ASSUME PRED P:1, Q:0

IN

existselim:

PROVE ($\exists$ FUNC x:0 $P(x)$) $\to$ ($\forall$ FUNC x:0 $P(x) \to Q$) $\to$ Q

END

It can be used as follows:

ASSUME p : ($\exists$FUNC x:0 $\neg\neg P(x)$) IN

ASSUME FUNC x:0 IN

a : PROVE $\neg\neg p(x) \to \exists$ FUNC x:0 $p(x)$

END

g : PROVE $\exists$ FUNC x:0 $p(x)$ FROM p, existselim

{ PRED P := $\lambda x\ \neg\neg p(x)$, PRED Q := ($\exists$ FUNC x:0 $p(x)$) }

END

Induction

Induction is represented by the following formula:

AXIOM ind:
$\forall$ PRED p:1 $p(0) \wedge [\ \forall$ FUNC n:0 $N(n) \rightarrow P(n) \rightarrow P(\mathrm{succ}(n))\ ]$
$\quad \rightarrow \forall$ FUNC m:0 $N(m) \rightarrow P(m)$.

It can be used as follows:

PROVE ... FROM ind { FUNC $p := \lambda x\ (0 + x = x)$ }.

Direct Definitions

Direct function definitions have the following form:

DEFINE plus2 AS E : λx succ(succ(x))

Indirect function definitions have the following form:

DEFINE plus2 FROM E : $\lambda xy \;\; y = \text{succ}(\text{succ}(x))$
UNIQUENESS EXISTENCE

(now you have more or less seen the complete system)

WUSO Formulas

Definition: A declaration has one of the following two forms:

- A function declaration FUNC $f{:}n$ declaring f as an n-ary function symbol.
- A predicate declaration PRED $p{:}n$ declaring p as an n-ary predicate symbol.

Definition: The set of weak untyped second order formulas is recursively defined as follows:

- A usual (first order) atom is a WUSO formula.
- WUSO formulas are closed under the propositional operators

$$\bot,\ \top,\ \neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow .$$

- If F is a WUSO formula, $D_1, \ldots, D_n$, is a sequence of declarations, then $\forall D_1, \ldots, D_n\ F$ and $\exists D_1, \ldots, D_n\ F$ are

WUSO formulas.

Contexts

A context remembers assumptions, declarations, definitions and theorems.

Using contexts for modelling natural deduction is standard. Our use is non-standard in two ways:

1. Conclusions are stored in contexts.
2. Items can be dropped from the middle.

Contexts (2)

Definition: A context Γ is a sequence of form $\Gamma_1, \ldots, \Gamma_p,\ p > 0$. Each Γ_i has form C_i or $L_i{:}C_i$. Each C_i must have one of the following forms:

1. A declaration FUNC $f{:}n$ or PRED $p{:}n$.
2. A definition of form FUNC $f := \lambda x_1 \cdots x_n\ t$ or of form PRED $p := \lambda x_1 \cdots x_n\ F$.
3. An indirect function definition of form FUNC $f := \lambda x_1 \cdots x_n\ y\ F$. (Definitions of type 2 are called direct)
4. An assumption F.
5. A theorem F.

Contexts (3)

Definition: For a context element C_i, we define the characteristic formula of C_i as follows:

- A declaration has no characteristic formula.
- The characteristic formula of a direct function definition FUNC $f := \lambda x_1 \cdots x_n\ t$ equals

$$\forall x_1 \cdots x_n\ f(x_1, \ldots, x_n) = t[x_1, \ldots, x_n].$$

- The c.f. of a predicate definition PRED $p := \lambda p_1 \cdots p_n\ F$ equals

$$\forall x_1 \cdots x_n\ p(x_1, \ldots, x_n) \leftrightarrow F[x_1, \ldots, x_n].$$

Contexts (4)

- The c.f. of an indirect function definition
  FUNC $f := \lambda x_1 \cdots x_n \; y \; F$ equals

$$\forall x_1 \cdots x_n \; y \; f(x_1, \ldots, x_n) = y \leftrightarrow F[x_1, \ldots, x_n, y].$$

- The c.f. of an assumption F equals F.
- The c.f of a theorem F equals F.

A context element C_i can have a label L_i if it has a characteristic formula.

Conditional Reasoning

The usual conditional rules are

$$\Gamma, A \vdash B \Rightarrow \Gamma \vdash A \to B,$$

and

$$\Gamma, x{:}\, A \vdash B \Rightarrow \Gamma \vdash \forall x{:}\, A\ B.$$

In our setting, theorems are part of the context and assumptions can be dropped from the middle of the context.

In order to be able to do this, we define permutation rules of form:

$$\Gamma_1, A, B, \Gamma_2 \ \Rightarrow\ \Gamma_1, B', A, \Gamma_2'.$$

Not all A, B can be permuted, but we try to define as many permutation rules as possible.

Conditional Reasoning (2)

- If A is a declaration of form FUNC $c{:}0$, and B is a definition of form FUNC $f := \lambda x_1 \cdots x_n \; t$, then

$$B' = \; f' := \lambda c \; x_1 \cdots x_n \; t,$$

  and

$$\Gamma' = \; \Gamma[\; f := \lambda x_1 \cdots x_n \; f'(c, x_1, \ldots, x_n) \;].$$

- The case where A has form FUNC $c{:}0$ and B has form PRED $p := \lambda x_1 \cdots x_n \; F$ is treated analogeously.

Conditional Reasoning (3)

- If A is a declaration of form FUNC $f{:}n$, and B is a theorem, then $B' = \ \forall$ FUNC $f{:}n\ B$, and $\Gamma' = \Gamma$.
- If A is a declaration of form PRED $p{:}n$, and B is a theorem, then $B' = \ \forall$ PRED $p{:}n\ B$, then and $\Gamma' = \Gamma$.
- If F is a formula assumption, and B is a theorem, then $B' = \ F \rightarrow B$, and $\Gamma' = \Gamma$.

Conditional Reasoning (4)

- If A is a definition of form FUNC $f := \lambda x_1 \cdots x_n\ t$, then $B' = \ B[\ f := \lambda x_1 \cdots x_n\ t\ ]$, and $\Gamma' = \Gamma$.
- If A is a definition of form PRED $p := \lambda x_1 \cdots x_n\ F$, then $B' = \ B[\ p := \lambda x_1 \cdots x_n\ F\ ]$, and $\Gamma' = \Gamma$.
- If A is a theorem, then $B' = B$, and $\Gamma' = \Gamma$.

Example

Consider the context:

FUNC $n{:}0$,

PRED $E := N(n) \wedge \exists$ FUNC $m{:}0\ N(m) \wedge m + m = n$,

THEOREM $E \rightarrow \exists$ FUNC $m{:}0\ d(m) = n$.

Dropping the first assumption FUNC $n{:}0$ results in

PRED $E' := \lambda n\ N(n) \wedge \exists$ FUNC $m{:}0\ N(m) \wedge m + m = n$,

THEOREM $\forall$ FUNC $n{:}0\ E(n) \rightarrow \exists$ FUNC $m{:}0\ d(m) = n$.

From

From is pretty much the same as 'by' in Mizar.

Let Γ be a context. We recursively define the set of references and the formulas that they refer to:

Definition:

- A label L is a reference. In case Γ contains an element with label L, L refers to the characteristic formula of this element.
- An expression of form $L \pm i$ is a reference. If L contains an element with label L, then $L \pm i$ refers to the i-th characteristic formula after (before) this element.
- An expression of form $-i$ is a reference. It refers to the i-th last characteristic formula of Γ.

From (2)

If R is a reference, and Θ a substitution of form

$$\text{FUNC } f_1 := \lambda\overline{x}_1\ t_1, \ldots, \text{FUNC } f_n := \lambda\overline{x}_n\ t_n,$$

$$\text{PRED } p_1 := \lambda\overline{y}_1\ F_1, \ldots, \text{PRED } p_m := \lambda\overline{y}_m\ F_m,$$

then $R\Theta$ is also a reference.

If R refers to a formula of form

$$\forall\ \text{FUNC } f_1 \cdots \text{FUNC } f_n\ \text{PRED } p_1 \cdots \text{PRED } p_m\ F,$$

(possibly after rearranging top level quantifiers), then $R\Theta$ refers to $F\Theta$.

From (3)

If R and S are references then $R \times S$ is a reference. If R refers to formula A, and S refers to formula $A \rightarrow B$, then $R \times S$ refers to B.

The general form of from is as follows:

$$(L :)\ F \text{ FROM } R_1, \ldots, R_m.$$

If each R_i refers to a formula F_i, then the first-order theorem prover is called with goal

$$F_1, \ldots, F_n \vdash F\ ?$$

If the prover succeeds then THEOREM F is added to the context.

Interface with the ATPer

With the input, the user can specify:

- The path of the prover,
- Its input parameters
- A string that specifies when the prover has found a proof.
- Format of input/output files, and whether the user wants to run the prover at all.

Conclusions and Future Work

- We developed some set theory, some arithmetic (within the set theory)
- A type system is necessary, at least simple types.
- Geo is quite good at terminating, and that is quite useful.
- Use system in class.
- Rewrite system using trusted code base.
- Redo SEFM 2005 (Hans de Nivelle and Ruzica Piskač, Verification of an off-line checker for priority queues)