First Class Boolean Type in First-Order Theorem Proving and TPTP

Evgeny Kotelnikov Laura Kovács Andrei Voronkov

ST Winter Meeting

February 3, 2015

A translation from source code to FOL

A code snippet

```
if (r(a)) {
  a := a + 1
} else {
  a := a + q(a)
}
```

A translation from source code to FOL

A code snippet

```
if (r(a)) {
  a := a + 1
} else {
  a := a + q(a)
}
```

The next state function for a

```
a' = if r(a) then
         let a = a + 1 in a
       else
         let a = a + q(a) in a
```

A translation from source code to FOL

A code snippet

```
if (r(a)) {
  a := a + 1
} else {
  a := a + q(a)
}
```

The next state function for a

```
a' = if r(a) then
        let a = a + 1 in a
     else
        let a = a + q(a) in a
```

The TPTP translation

```
tff(a_def, type, a : $int).
tff(a_next_state, hypothesis,
  a' = $ite_t(r(a), $let_tt(a, $sum(a, 1), a),
                    $let_tt(a, $sum(a, q(a)), a))).
```

Boolean flags

```
void bubbleSort(int n, int array[]) {
    bool isSorted;
    do {
        isSorted = true;
        for (int i = 0; i < n - 1; i++) {
            if (array[i] > array[i + 1]) {
                swap(array[i], array[i + 1]);
                isSorted = false;
                break;
            }
        }
    } while (!isSorted);
}
```

FOL with first class boolean sort

FOL + bool = FOOL

1. The $bool$ sort
2. Terms of the sort $bool$ can be returned from a function and be passed as arguments
3. Terms of the sort $bool$ and formulas are interchangeble
4. `if-then-else` and `let-in` included

Outline

Type system of FOL

Type system of FOOL

Translation from FOOL to FOL

Implementation in a theorem prover

Future work

Outline

- Type system of FOL
- Type system of FOOL
- Translation from FOOL to FOL
- Implementation in a theorem prover
- Future work

Syntax of FOL

$$
\begin{aligned}
\mathit{term} &\rightarrow \mathit{constant} \\
&\mid \ \mathit{variable} \\
&\mid \ \mathit{function}(\mathit{term}, \ldots) \\[1em]
\mathit{connective} &\rightarrow \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots \\[1em]
\mathit{formula} &\rightarrow \mathit{predicate}(\mathit{term}, \ldots) \\
&\mid \ \mathit{term} = \mathit{term} \\
&\mid \ \mathit{formula}\ \mathit{connective}\ \mathit{formula} \\
&\mid \ \neg\ \mathit{formula} \\
&\mid \ \forall\ \mathit{variable}\ \mathit{formula} \\
&\mid \ \exists\ \mathit{variable}\ \mathit{formula}
\end{aligned}
$$

Sorts in FOL

Terms: $atomic\ (\alpha, \beta, \ldots)$

Formulas: bool

Variables: $atomic$

Function symbols: $atomic \times \ldots \times atomic \to atomic$

Predicate symbols: $atomic \times \ldots \times atomic \to \mathrm{bool}$

Typing rules in FOL

Function application

$$\frac{\Gamma \vdash t_1 : \sigma_1, \ldots, \Gamma \vdash t_n : \sigma_n \qquad \Gamma \vdash f : \sigma_1 \times \ldots \times \sigma_n \to \tau}{\Gamma \vdash f(t_1, \ldots, t_n) : \tau}$$

Predicate application

$$\frac{\Gamma \vdash t_1 : \sigma_1, \ldots, \Gamma \vdash t_n : \sigma_n \qquad \Gamma \vdash P : \sigma_1 \times \ldots \times \sigma_n \to \mathrm{bool}}{\Gamma \vdash P(t_1, \ldots, t_n) : \mathrm{bool}}$$

Logical connectives

$$\frac{\Gamma \vdash P : \mathrm{bool} \qquad \Gamma \vdash Q : \mathrm{bool}}{\Gamma \vdash P \otimes Q : \mathrm{bool}} \otimes \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$$

Extension with if-then-else and let-in

$$
\begin{aligned}
\mathit{term} \rightarrow\ & \ldots \\
\mid\ & \text{\$ite\_t}(\mathit{formula}, \mathit{term}, \mathit{term}) \\
\mid\ & \text{\$let\_tt}(\mathit{function} = \mathit{term}, \mathit{term}) \\
\mid\ & \text{\$let\_ft}(\mathit{function} = \mathit{formula}, \mathit{term})
\end{aligned}
$$

$$
\begin{aligned}
\mathit{formula} \rightarrow\ & \ldots \\
\mid\ & \text{\$ite\_f}(\mathit{formula}, \mathit{formula}, \mathit{formula}) \\
\mid\ & \text{\$let\_tf}(\mathit{function} = \mathit{term}, \mathit{formula}) \\
\mid\ & \text{\$let\_ff}(\mathit{function} = \mathit{formula}, \mathit{formula})
\end{aligned}
$$

Outline

Type system of FOL

Type system of FOOL

Translation from FOOL to FOL

Implementation in a theorem prover

Future work

Syntax of FOOL

$$
\begin{aligned}
\mathit{term} \rightarrow\ & \mathit{constant} \\
\mid\ & \mathit{variable} \\
\mid\ & \mathit{function}(\mathit{term}, \ldots) \\
\mid\ & \mathit{term} = \mathit{term} \\
\mid\ & \forall\ \mathit{variable}\ \mathit{term} \\
\mid\ & \exists\ \mathit{variable}\ \mathit{term} \\
\mid\ & \texttt{if}\ \mathit{term}\ \texttt{then}\ \mathit{term}\ \texttt{else}\ \mathit{term} \\
\mid\ & \texttt{let}\ \mathit{function}\ =\ \mathit{term}\ \texttt{in}\ \mathit{term}
\end{aligned}
$$

Interpreted function symbols: true, false, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$

Sorts in FOOL

Terms: $atomic\ (\mathrm{bool}, \alpha, \beta, \ldots)$

Variables: $atomic$

Function symbols: $atomic \times \ldots \times atomic \to atomic$

Typing rules in FOOL

Function application

$$\frac{\Gamma \vdash t_1 : \sigma_1, \ldots, \Gamma \vdash t_n : \sigma_n \qquad \Gamma \vdash f : \sigma_1 \times \ldots \times \sigma_n \to \tau}{\Gamma \vdash f(t_1, \ldots, t_n) : \tau}$$

`if-then-else`

$$\frac{\Gamma \vdash P : \mathrm{bool} \qquad \Gamma \vdash s : \sigma \qquad \Gamma \vdash t : \sigma}{\Gamma \vdash \texttt{if } P \texttt{ then } s \texttt{ else } t : \sigma}$$

`let-in`

$$\frac{\Gamma \vdash s : \sigma \qquad \Gamma, c : \sigma \vdash t : \tau}{\Gamma \vdash \texttt{let } c = s \texttt{ in } t : \tau}$$

Outline

Translation from FOOL to FOL

Replace subterms that are invalid in FOL

1. Boolean variables as arguments to connectives or directly under quantifiers
2. Formulas as arguments to function symbols
3. `if-then-else` expressions
4. `let-in` expressions

Translation from FOOL to FOL

Step 1

ϕ in FOOL becomes $\bigwedge_{\psi \in S} \psi \Rightarrow \phi'$:

1. Replace x with $x = \mathrm{true}$
2. Replace ϕ with g, add $\phi \Leftrightarrow g = \mathrm{true}$ to S
3. Replace $\texttt{if } \phi \texttt{ then } s \texttt{ else } t$ with g,
   add $\phi \Rightarrow g = s$ and $\neg\phi \Rightarrow g = t$ to S
4. Replace $\texttt{let } f(x_1, \ldots, x_n) = s \texttt{ in } t$ with t, where each $f(t_1, \ldots, t_n)$ is replaced with $g(t_1, \ldots, t_n)$,
   add $\forall x_1, \ldots, x_n (g(x_1, \ldots, x_n) = s)$ to S

Translation from FOOL to FOL

Step 2

Append two axioms that shape the boolean sort:

1. Finite domain axiom: $\forall(x : \mathrm{bool})(x = \mathrm{true} \vee x = \mathrm{false})$
2. Distinct constants axiom: $\mathrm{true} \neq \mathrm{false}$

$\bigwedge_{\psi \in S} \psi \Rightarrow \phi'$ becomes $FD \wedge DC \wedge \bigwedge_{\psi \in S} \psi \Rightarrow \phi'$

Outline

Type system of FOL

Type system of FOOL

Translation from FOOL to FOL

Implementation in a theorem prover

Future work

Implementation in a theorem prover

The paramodulation rule

$$\frac{\phi \vee s = t \qquad \psi[s]}{\phi \vee \psi[t]}$$

Implementation in a theorem prover

The paramodulation rule

$$\frac{\phi \vee s = t \qquad \psi[s]}{\phi \vee \psi[t]}$$

A good paramodulation with the finite domain axiom

$$\frac{\phi[t : \mathrm{bool}] \qquad t = \mathrm{true} \vee t = \mathrm{false}}{\phi[\mathrm{true}] \vee t = \mathrm{false}}$$

Implementation in a theorem prover

The paramodulation rule

$$\frac{\phi \vee s = t \qquad \psi[s]}{\phi \vee \psi[t]}$$

A good paramodulation with the finite domain axiom

$$\frac{\phi[t : \mathrm{bool}] \qquad t = \mathrm{true} \vee t = \mathrm{false}}{\phi[\mathrm{true}] \vee t = \mathrm{false}}$$

A bad paramodulation with the finite domain axiom

$$\frac{x = \mathrm{true} \vee x = \mathrm{false} \qquad y = \mathrm{true} \vee y = \mathrm{false}}{x = \mathrm{false} \vee y = \mathrm{false} \vee x = y}$$

Outline

Type system of FOL

Type system of FOOL

Translation from FOOL to FOL

Implementation in a theorem prover

Future work

Future work

1. Implementation in Vampire
2. Better translation of `if-then-else` and `let-in`
3. A starting point for further refinement of the type system