A First Class Boolean Sort in First-Order Theorem Proving and TPTP

Evgenii Kotelnikov[1] Laura Kovács[1] Andrei Voronkov[2]

[1] Chalmers University of Technology, Gothenburg, Sweden

[2] The University of Manchester, Manchester, UK

CICM 2015

16 July

Introduction: Many-Sorted First-Order Logic

Syntax

$$\begin{aligned} \mathit{term} &\rightarrow \mathit{var} \\ &\mid\ f(\mathit{term}, \ldots) \end{aligned}$$

$$\begin{aligned} \mathit{formula} &\rightarrow p(\mathit{term}, \ldots) \\ &\mid\ \mathit{term} = \mathit{term} \\ &\mid\ \mathit{formula} \otimes \mathit{formula} \\ &\mid\ \neg\ \mathit{formula} \\ &\mid\ \forall\ \mathit{var}\ \mathit{formula} \\ &\mid\ \exists\ \mathit{var}\ \mathit{formula} \end{aligned}$$

Sorts

$$\mathit{sort} \rightarrow \mathrm{int}, \mathrm{real}, \alpha, \beta, \ldots$$

$$\mathit{var} : \mathit{sort}$$

$$f : \mathit{sort} \times \ldots \times \mathit{sort} \rightarrow \mathit{sort}$$

$$p : \mathit{sort} \times \ldots \times \mathit{sort}$$

Introduction: Many-Sorted First-Order Logic

Syntax

$$
\begin{aligned}
term &\rightarrow var \\
&\mid\ f(term, \ldots) \\
\\
formula &\rightarrow p(term, \ldots) \\
&\mid\ term = term \\
&\mid\ formula \otimes formula \\
&\mid\ \neg\ formula \\
&\mid\ \forall\ var\ formula \\
&\mid\ \exists\ var\ formula
\end{aligned}
$$

Sorts

$$sort \rightarrow \mathrm{int}, \mathrm{real}, \alpha, \beta, \ldots$$

$$
\begin{aligned}
var &: sort \\
f &: sort \times \ldots \times sort \rightarrow sort \\
p &: sort \times \ldots \times sort
\end{aligned}
$$

Introduction: FOL with First-Class Boolean Sort

Syntax

$$
\begin{aligned}
term \rightarrow\ & var \\
\mid\ & f(term, \ldots) \\
\mid\ & term = term \\
\mid\ & \forall\ var\ term \\
\mid\ & \exists\ var\ term
\end{aligned}
$$

Terms of the sort bool are formulas

Connective are interpreted boolean functions

Sorts

$$sort \rightarrow \mathrm{bool}, \mathrm{int}, \mathrm{real}, \alpha, \beta, \ldots$$

$$var : sort$$

$$f : sort \times \ldots \times sort \rightarrow sort$$

Introduction: FOL with First-Class Boolean Sort

Syntax

$$
\begin{aligned}
term \to\ & var \\
\mid\ & f(term, \ldots) \\
\mid\ & term = term \\
\mid\ & \forall\ var\ term \\
\mid\ & \exists\ var\ term
\end{aligned}
$$

Sorts

$$sort \to \mathrm{bool}, \mathrm{int}, \mathrm{real}, \alpha, \beta, \ldots$$

$$
\begin{aligned}
var &: sort \\
f &: sort \times \ldots \times sort \to sort
\end{aligned}
$$

Terms of the sort bool are formulas

Connective are interpreted boolean functions

Contributions

FOOL = FOL + Bool

- FOL with first class boolean sort, `if-then-else` and `let-in`
- A translation from FOOL to FOL
- A technique for efficient superposition in FOOL
- A proposal for changes in TPTP

Outline

Motivation

Translation from FOOL to FOL

Superposition for FOOL

Changes to TPTP

Future work

Outline

Motivation

Translation from FOOL to FOL

Superposition for FOOL

Changes to TPTP

Future work

Motivation: Proof automation

Interactive theorem provers routinely use quantifiers over booleans.

An example from Isabelle/HOL

$$(\forall p : bool)(\forall l : list_A)(\forall x : A)(\forall y : A)$$
$$\mathtt{contains}(l, \mathrm{ite}(p, x, y)) =$$
$$(p \Rightarrow \mathtt{contains}(l, x)) \wedge (\neg p \Rightarrow \mathtt{contains}(l, y))$$

SMT-LIB

FOOL is the smallest superset of SMT-LIB Core and TF0.

Motivation: Program analysis

Straightforward mapping of PL's boolean type.

Bubble sort

```
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);
```

Motivation: if-then-else and let-in

Syntax in FOOL

$$
\begin{aligned}
term \rightarrow\ & ... \\
& \mid\ \texttt{if}\ term\ \texttt{then}\ term\ \texttt{else}\ term \\
& \mid\ \texttt{let}\ f(var : sort, ...) = term\ \texttt{in}\ term
\end{aligned}
$$

In current TPTP

1. `$ite_t` and `$ite_f`
2. `$let_tt`, `$let_tf`, `$let_ft` and `$let_ff`

Outline

Translation from FOOL to FOL

- Every FOL formula is syntactically a FOOL formula, but not the other way around.
- A restricted subset of FOOL is FOL with a distinguished boolean sort and constants $true$ and $false$.
- The translation replaces FOOL subterms that are not allowed in FOL with the ones that are, preserving models.

Translation from FOOL to FOL

- Every FOL formula is syntactically a FOOL formula, but not the other way around.
- A restricted subset of FOOL is FOL with a distinguished boolean sort and constants $true$ and $false$.
- The translation replaces FOOL subterms that are not allowed in FOL with the ones that are, preserving models.

Translation from FOOL to FOL

- Every FOL formula is syntactically a FOOL formula, but not the other way around.
- A restricted subset of FOOL is FOL with a distinguished boolean sort and constants $true$ and $false$.
- The translation replaces FOOL subterms that are not allowed in FOL with the ones that are, preserving models.

Translation from FOOL to FOL

Terms that are allowed in FOOL but not in FOL

1. Boolean variables in formula context.

   $(\forall x : \mathrm{bool})\,(x \vee P(x))$

2. Formulas in term context.

   $(\forall x : \sigma_1)\,P((\forall y : \sigma_2)\,Q(x, y))$

3. `if-then-else` expressions.

   $(\forall x : \sigma)\,P(\texttt{if }Q(x)\texttt{ then }c_1\texttt{ else }c_2)$

4. `let-in` expressions.

   $(\forall x : \sigma)\,(x = \texttt{let }f(y : \sigma) = p(x, q(y))\texttt{ in }f(f(x)))$

Translation from FOOL to FOL

Input

- FOOL formula φ
- Set of definitinos $D = \emptyset$

Apply replacements

Each of four replacements:

- Introduces a fresh symbol
- Makes a substitution in φ
- Might add a formula to D

Output

FOOL formula $\bigwedge_{\psi \in D} \psi \wedge \varphi'$

Translation from FOOL to FOL: Replacements

Boolean variable x in formula context

Replace x with $x = true$.

Translation from FOOL to FOL: Replacements

$(\forall x : \text{bool})\,(x \vee P(x))$

Translation from FOOL to FOL: Replacements

$(\forall x : \text{bool})\,(x = true \vee P(x))$

Translation from FOOL to FOL: Replacements

Formula φ in term context

1. Let $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ be all free variables of φ.
2. Add definition of a fresh symbol g
   $(\forall x_1 : \sigma_1) \ldots (\forall x_n : \sigma_n)\ (\varphi \Leftrightarrow g(x_1, \ldots, x_n) = true)$.
3. Replace φ by $g(x_1, \ldots, x_n)$.

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma_1)\, P((\forall y : \sigma_2)\, Q(x, y))$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma_1)\, P((\forall y : \sigma_2)\, Q(x, y))$

Add $(\forall x : \sigma_1)\, ((\forall y : \sigma_2)\ Q(x, y) \Leftrightarrow g(x) = \mathit{true})$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma_1)\, P(g(x))$

Add $(\forall x : \sigma_1)\, ((\forall y : \sigma_2)\, Q(x, y) \Leftrightarrow g(x) = true)$

Translation from FOOL to FOL: Replacements

$\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t$

1. Let $x_1 : \sigma_1, \dots, x_n : \sigma_n$ be all free variables of φ, s and t.
2. Add definitions of a fresh symbol g
   $(\forall x_1 : \sigma_1) \dots (\forall x_n : \sigma_n)\ (\ \ \varphi \Rightarrow g(x_1, \dots, x_n) = s)$ and
   $(\forall x_1 : \sigma_1) \dots (\forall x_n : \sigma_n)\ (\neg\varphi \Rightarrow g(x_1, \dots, x_n) = t)$.
3. Replace $\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t$ by $g(x_1, \dots, x_n)$.

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\, P(\texttt{if } Q(x) \texttt{ then } c_1 \texttt{ else } c_2)$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\, P(\texttt{if } Q(x) \texttt{ then } c_1 \texttt{ else } c_2)$

Add $(\forall x : \sigma)\,(\quad Q(x) \Rightarrow g(x) = c_1)$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\, P(\texttt{if } Q(x) \texttt{ then } c_1 \texttt{ else } c_2)$

Add $(\forall x : \sigma)\,(\quad Q(x) \Rightarrow g(x) = c_1)$

Add $(\forall x : \sigma)\,(\neg Q(x) \Rightarrow g(x) = c_2)$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\, P(g(x))$

Add $(\forall x : \sigma)\, (\;\;\, Q(x) \Rightarrow g(x) = c_1)$

Add $(\forall x : \sigma)\, (\neg Q(x) \Rightarrow g(x) = c_2)$

Translation from FOOL to FOL: Replacements

$\texttt{let}\ f(x_1 : \sigma_1, \ldots, x_n : \sigma_n) \ = \ s\ \texttt{in}\ t$

1. Let $y_1 : \tau_1, \ldots, y_m : \tau_m$ be all free variables of s and t.
2. Add definition of a fresh symbol g
   $(\forall x_1 : \sigma_1) \ldots (\forall x_n : \sigma_n)(\forall y_1 : \tau_1) \ldots (\forall y_m : \tau_m)$
   $(g(x_1, \ldots, x_n, y_1, \ldots, y_m) = s)$.
3. Replace $\texttt{let}\ f(x_1 : \sigma_1, \ldots, x_n : \sigma_n) \ = \ s\ \texttt{in}\ t$ by t with each application $f(t_1, \ldots, t_n)$ of a free occurrence of f replaced by $g(t_1, \ldots, t_n, y_1, \ldots, y_m)$.

Translation from FOOL to FOL: Replacements

$$(\forall x : \sigma)\,(x = \texttt{let}\, f(y : \sigma) \;=\; p(x, q(y))\; \texttt{in}\, f(f(x)))$$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\,(x = \texttt{let}\, f(y : \sigma) \;=\; p(x, q(y))\;\texttt{in}\, f(f(x)))$

Add $(\forall x : \sigma)(\forall y : \tau)(g(x, y) = p(x, q(y)))$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\,(x = f(f(x)))$

Add $(\forall x : \sigma)(\forall y : \tau)(g(x, y) = p(x, q(y)))$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\,(x = f(g(x, x)))$

Add $(\forall x : \sigma)(\forall y : \tau)(g(x, y) = p(x, q(y)))$

Translation from FOOL to FOL: Replacements

$(\forall x : \sigma)\,(x = g(g(x, x), x))$

Add $(\forall x : \sigma)(\forall y : \tau)(g(x, y) = p(x, q(y)))$

Translation from FOOL to FOL: Final step

$$\bigwedge_{\psi \in D} \psi \wedge \varphi' \wedge (\forall x : bool)\,(x = true \vee x = false) \wedge (true \neq false)$$

Translation from FOOL to FOL: Final step

$$\bigwedge_{\psi \in D} \psi \wedge \varphi' \wedge (\forall x : bool)\,(x = true \vee x = false) \wedge (true \neq false)$$

Translation from FOOL to FOL: Final step

$$\bigwedge_{\psi \in D} \psi \wedge \varphi' \wedge (\forall x : bool)\,(x = true \vee x = false) \wedge (true \neq false)$$

Outline

Superposition for FOOL: Paramodulation

Paramodulation rule

$$\frac{l = r \lor C \qquad L[s] \lor D}{(L[r] \lor C \lor D)\theta} \quad \theta = \mathrm{mgu}(l, s)$$

Ordered paramodulation in action

$$\frac{f(g(b)) = a \lor R(c) \qquad P(g(f(x))) \lor Q(x)}{P(g(a)) \lor Q(g(b)) \lor R(c)} \quad \theta = \{x \mapsto g(b)\}$$

Superposition for FOOL: Paramodulation

Paramodulation rule

$$\frac{l = r \vee C \qquad L[s] \vee D}{(L[r] \vee C \vee D)\theta} \quad \theta = \mathrm{mgu}(l, s)$$

Ordered paramodulation in action

$$\frac{f(g(b)) = a \vee R(c) \qquad P(g(f(x))) \vee Q(x)}{P(g(a)) \vee Q(g(b)) \vee R(c)} \quad \theta = \{x \mapsto g(b)\}$$

Superposition for FOOL: A problem

$$(\forall x : bool)\,(x = true \lor x = false)$$

Self-paramodulation from $true$ to $true$

$$\frac{x = true \lor x = false \qquad y = true \lor y = false}{x = y \lor x = false \lor y = false}$$

Superposition for FOOL: A problem

$$(\forall x : \mathit{bool})\,(x = \mathit{true} \vee x = \mathit{false})$$

Self-paramodulation from true to true

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

Superposition for FOOL: Our solution

Fixed term ordering

$true \succ false$ and they both are smaller than all other terms.

The only possible inference for $x = true \vee x = false$

$$\frac{x = true \vee x = false \quad C[s]}{C[true] \vee s = false}$$

An extra inference rule instead of $x = true \vee x = false$

$$\frac{C[s]}{C[true] \vee s = false}$$

Superposition for FOOL: Our solution

Fixed term ordering

$true \succ false$ and they both are smaller than all other terms.

The only possible inference for $x = true \vee x = false$

$$\frac{x = true \vee x = false \quad C[s]}{C[true] \vee s = false}$$

An extra inference rule instead of $x = true \vee x = false$

$$\frac{C[s]}{C[true] \vee s = false}$$

Superposition for FOOL: Our solution

Fixed term ordering

$\mathit{true} \succ \mathit{false}$ and they both are smaller than all other terms.

The only possible inference for $x = \mathit{true} \vee x = \mathit{false}$

$$\frac{x = \mathit{true} \vee x = \mathit{false} \qquad C[s]}{C[\mathit{true}] \vee s = \mathit{false}}$$

An extra inference rule instead of $x = \mathit{true} \vee x = \mathit{false}$

$$\frac{C[s]}{C[\mathit{true}] \vee s = \mathit{false}}$$

Outline

Motivation

Translation from FOOL to FOL

Superposition for FOOL

Changes to TPTP

Future work

Changes to TPTP

Symbol declarations in TF0

```
tff(plus, type, plus : (int * int) > int).
tff(less, type, less : (int * int) > $o).
```

To support FOOL

- Allow `$o` to be the sort of an argument
- Allow quantification and equality over `$o`
- Allow formulas inside terms (when sorts coincide)
- Unify `$ite_t` and `$ite_f`
- Unify `$let_tt`, `$let_tf`, `$let_ft` and `$let_ff`

Changes to TPTP

Symbol declarations in TF0

```
tff(plus, type, plus : (int * int) > int).
tff(less, type, less : (int * int) > $o).
```

To support FOOL

- Allow `$o` to be the sort of an argument
- Allow quantification and equality over `$o`
- Allow formulas inside terms (when sorts coincide)
- Unify `$ite_t` and `$ite_f`
- Unify `$let_tt`, `$let_tf`, `$let_ft` and `$let_ff`

Outline

Motivation

Translation from FOOL to FOL

Superposition for FOOL

Changes to TPTP

Future work

Future work

- Implementation of FOOL in Vampire
- Experiments in reasoning in FOOL
- Better translation of `if-then-else` and `let-in`
- Support for TF1
- SMT-LIB parser