The Vampire and the FOOL

Evgenii Kotelnikov[1] Laura Kovács[1] Giles Reger[2]
Andrei Voronkov[123]

[1] Chalmers University of Technology, Gothenburg, Sweden

[2] The University of Manchester, Manchester, UK

[3] Easychair

CPP 2016

18 January

Outline

Outline

First-Order Logic with First Class Boolean Sort

E. Kotelnikov, L. Kovács, and A. Voronkov. *A First Class Boolean Sort in First-Order Theorem Proving and TPTP.* In Proc. of CICM, volume 9150 of LNCS, pages 71–86, 2015.

FOOL (FOL + Bool)

1. First-class boolean sort
   - Interpreted sort $bool$
   - Boolean variables can be used as formulas
   - Formulas can be used as boolean terms
2. `if-then-else` expressions
3. `let-in` expressions

Reasoning with the Boolean Sort

- Add boolean constants $true$ and $false$
- Add axioms $true \not\doteq false$ and $(\forall x : bool)(x \doteq true \vee x \doteq false)$

FOOL paramodulation

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

where

1. s is a term of the sort $bool$ other than $true$ and $false$
2. s is not a variable

Enabled with `--fool_paramodulation on`

Implementation of FOOL in Vampire

- Translate FOOL formulas into standard FOL
- For each part of the input problem that is not syntactically correct in standard FOL
  1. Introduce a fresh function or predicate symbol
  2. Replace with an application of the symbol
  3. Extend the set of assumtions with definition(s) for the symbol
- Extend input language (modification of TFF0)

Boolean variables

Example

$(\forall x : bool)(x \vee \neg x)$

Translation

$(\forall x : bool)(x \doteq true \vee x \not\doteq true)$

TPTP

```
![X:$o]: (X | ~X)
```

Boolean variables

Example

$(\forall x : bool)(x \vee \neg x)$

Translation

$(\forall x : bool)(x \doteq true \vee x \not\doteq true)$

TPTP

```
![X:$o]: (X | ~X)
```

Boolean variables

Example

$(\forall x : bool)(x \vee \neg x)$

Translation

$(\forall x : bool)(x \doteq true \vee x \not\doteq true)$

TPTP

```
![X:$o]: (X | ~X)
```

Functions and Predicates with Boolean Arguments

Example

$(\forall x : bool)(\forall y : bool)(impl(x, y) \Leftrightarrow \neg x \vee y)$

TPTP

```
tff(impl, type, ($o * $o) > $o).
tff(impl_definition, axiom,
    ![X:$o, Y:$o]: (impl(X,Y) <=> (~X | Y))).
```

Formulas as Arguments

Example

$$(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau) impl(P(x, y) \wedge P(x, z), y \doteq z)$$

Translation

$$(\forall x : \sigma)(\forall y : \sigma)(\forall z : \sigma) impl(g_1(x, y, z), g_2(y, z))$$

- $(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau)(P(x, y) \wedge P(x, z) \Leftrightarrow g_1(x, y, z) \doteq true)$
- $(\forall y : \tau)(\forall z : \tau)(y \doteq z \Leftrightarrow g_2(y, z) \doteq true)$

TPTP

```
![X:s, Y:t, Z:t]: impl(p(X,Y) & p(X,Z), Y = Z)
```

Formulas as Arguments

Example

$(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau) impl(P(x, y) \wedge P(x, z), y \doteq z)$

Translation

$(\forall x : \sigma)(\forall y : \sigma)(\forall z : \sigma) impl(g_1(x, y, z), g_2(y, z))$

- $(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau)(P(x, y) \wedge P(x, z) \Leftrightarrow g_1(x, y, z) \doteq true)$
- $(\forall y : \tau)(\forall z : \tau)(y \doteq z \Leftrightarrow g_2(y, z) \doteq true)$

TPTP

```
![X:s, Y:t, Z:t]: impl(p(X,Y) & p(X,Z), Y = Z)
```

Formulas as Arguments

Example

$(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau) impl(P(x, y) \wedge P(x, z), y \doteq z)$

Translation

$(\forall x : \sigma)(\forall y : \sigma)(\forall z : \sigma) impl(g_1(x, y, z), g_2(y, z))$

- $(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau)(P(x, y) \wedge P(x, z) \Leftrightarrow g_1(x, y, z) \doteq true)$
- $(\forall y : \tau)(\forall z : \tau)(y \doteq z \Leftrightarrow g_2(y, z) \doteq true)$

TPTP

```
![X:s, Y:t, Z:t]: impl(p(X,Y) & p(X,Z), Y = Z)
```

Formulas as Arguments

Example

$(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau) impl(P(x, y) \wedge P(x, z), y \doteq z)$

Translation

$(\forall x : \sigma)(\forall y : \sigma)(\forall z : \sigma) impl(g_1(x, y, z), g_2(y, z))$

- $(\forall x : \sigma)(\forall y : \tau)(\forall z : \tau)(P(x, y) \wedge P(x, z) \Leftrightarrow g_1(x, y, z) \doteq true)$
- $(\forall y : \tau)(\forall z : \tau)(y \doteq z \Leftrightarrow g_2(y, z) \doteq true)$

TPTP

```
![X:s, Y:t, Z:t]: impl(p(X,Y) & p(X,Z), Y = Z)
```

`if-then-else` expressions

Examples

1. $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(max(x, y) \doteq \texttt{if } x \geq y \texttt{ then } x \texttt{ else } y)$
2. $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(\texttt{if } max(x, y) \doteq x \texttt{ then } x \geq y \texttt{ else } y \geq x)$

Translation

$(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(max(x, y) \doteq g(x, y))$

- $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(x \geq y \Rightarrow g(x, y) \doteq x)$
- $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(x \ngeq y \Rightarrow g(x, y) \doteq y)$

`if-then-else` expressions

Examples

1. $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(max(x, y) \doteq \texttt{if } x \geq y \texttt{ then } x \texttt{ else } y)$
2. $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(\texttt{if } max(x, y) \doteq x \texttt{ then } x \geq y \texttt{ else } y \geq x)$

Translation

$(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(max(x, y) \doteq g(x, y))$

- $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(x \geq y \Rightarrow g(x, y) \doteq x)$
- $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(x \ngeq y \Rightarrow g(x, y) \doteq y)$

`if-then-else` expressions

Examples

1. $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(max(x, y) \doteq \texttt{if } x \geq y \texttt{ then } x \texttt{ else } y)$
2. $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(\texttt{if } max(x, y) \doteq x \texttt{ then } x \geq y \texttt{ else } y \geq x)$

Translation

$(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(max(x, y) \doteq g(x, y))$

- $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(x \geq y \Rightarrow g(x, y) \doteq x)$
- $(\forall x : \mathbb{Z})(\forall y : \mathbb{Z})(x \ngeq y \Rightarrow g(x, y) \doteq y)$

if-then-else expressions

TPTP

```
tff(max, type, max: ($int * $int) > $int).
tff(max_definition, axiom,
    ![X:$int, Y:$int]:
      (max(X,Y) = $ite($greatereq(X,Y),X,Y))).

tff(max_property, hypothesis,
    ![X:$int, Y:$int]:
      $ite(max(X,Y) = X, $greatereq(X,Y),
                         $greatereq(Y,X)).
```

`let-in` expressions

$$
\begin{aligned}
&\texttt{let } f_1(x_1^1 : \sigma_1^1, \ldots, x_{n_1}^1 : \sigma_{n_1}^1) = s_1; \\
&\qquad \ldots \\
&\qquad f_m(x_1^m : \sigma_1^m, \ldots, x_{n_m}^m : \sigma_{n_m}^m) = s_m \\
&\texttt{in } t
\end{aligned}
$$

`let-in` expressions: Examples

Code fragment

```
array[3] := 5;
array[2] + array[3];
```

In FOOL

$\texttt{let}\ array(i : \mathbb{Z}) = \texttt{if}\ i \doteq 3\ \texttt{then}\ 5\ \texttt{else}\ array(i)$
$\texttt{in}\ array(2) + array(3)$

Translation

$g(2) + g(3)$

- $(\forall i : \mathbb{Z})(g(i) \doteq \texttt{if}\ i \doteq 3\ \texttt{then}\ 5\ \texttt{else}\ array(i))$.

TPTP

```
$let(array(I:$int) := $ite(I = 3, 5, array(I)),
    $sum(array(2), array(3))).
```

`let-in` expressions: Examples

Code fragment

```
array[3] := 5;
array[2] + array[3];
```

In FOOL

let $array(i : \mathbb{Z}) = \texttt{if}\ i \doteq 3\ \texttt{then}\ 5\ \texttt{else}\ array(i)$
in $array(2) + array(3)$

Translation

$g(2) + g(3)$

- $(\forall i : \mathbb{Z})(g(i) \doteq \texttt{if}\ i \doteq 3\ \texttt{then}\ 5\ \texttt{else}\ array(i))$.

TPTP

```
$let(array(I:$int) := $ite(I = 3, 5, array(I)),
    $sum(array(2), array(3))).
```

`let-in` expressions: Examples

Code fragment

```
array[3] := 5;
array[2] + array[3];
```

In FOOL

$\texttt{let}\ array(i : \mathbb{Z}) = \texttt{if}\ i \doteq 3\ \texttt{then}\ 5\ \texttt{else}\ array(i)$
$\texttt{in}\ array(2) + array(3)$

Translation

$g(2) + g(3)$

- $(\forall i : \mathbb{Z})(g(i) \doteq \texttt{if}\ i \doteq 3\ \texttt{then}\ 5\ \texttt{else}\ array(i))$.

TPTP

```
$let(array(I:$int) := $ite(I = 3, 5, array(I)),
    $sum(array(2), array(3))).
```

`let-in` expressions: Examples

Code fragment

```
array[3] := 5;
array[2] + array[3];
```

In FOOL

let $array(i : \mathbb{Z}) = \texttt{if } i \doteq 3 \texttt{ then } 5 \texttt{ else } array(i)$
in $array(2) + array(3)$

Translation

$g(2) + g(3)$

- $(\forall i : \mathbb{Z})(g(i) \doteq \texttt{if } i \doteq 3 \texttt{ then } 5 \texttt{ else } array(i))$.

TPTP

```
$let(array(I:$int) := $ite(I = 3, 5, array(I)),
     $sum(array(2), array(3))).
```

`let-in` expressions: Examples

Code fragment

```
a, b := b, a
```

TPTP

```
$let(a := b; b := a, f(a,b))
```

Experimental Results

TPTP Problems

Translated from some of the THF0 problems of TPTP

Prover	Solved	Time
Vampire	134	3.59
Vampire⋆	134	7.28
Satallax	134	23.93
Leo-II	127	27.42
Isabelle	128	893.80

A total of 134 problems

Algebraic Datatypes Problems

Translated from SMT-LIB problems generated by Isabelle

Prover	Solved	Time
Vampire	59	26.580
Z3	57	4.291
Vampire⋆	56	26.095
CVC4	53	25.480

A total of 152 problems

Outline

Polymorphic Theory of Arrays

A union of theories of arrays parametrised by two sorts: sort τ of indexes and sort σ of values

Theory of arrays with indexes τ and values σ

- Sort $array(\tau, \sigma)$
- Function symbol $select : array(\tau, \sigma) \times \tau \to \sigma$
- Function symbol $store : array(\tau, \sigma) \times \tau \times \sigma \to array(\tau, \sigma)$
- Three theory axioms
  1. read-over-write 1
  2. read-over-write 2
  3. extensionality

Polymorphic Theory of Arrays

Example

$\texttt{let}\ array = store(array, 3, 5)$
$\texttt{in}\ select(array, 2) + select(array, 3)$

TPTP

```
tff(array_type, type, array: $array($int,$int)).
$let(array := $store(array,3,5),
     $sum($select(array,2), $select(array,3))).
```

Outline

Program Analysis with Vampire

Code fragment

```
res := x;
if (x > y)
    then max := x;
    else max := y;
if (max > 0)
    then res := res + max;
    else res := res − max;
```

FOOL

```
$let(res := x,
  $let(max := $ite($greater(x,y),
                   x, y),
    $let(res := $ite($greater(max,0),
                     $sum(res,max),
                     $diff(res,max)),
      res))).
```

Program Analysis with Vampire

Code fragment

```
res := x;
if (x > y)
    then max := x;
    else max := y;
if (max > 0)
    then res := res + max;
    else res := res - max;
```

FOOL

```
$let(res := x,
  $let(max := $ite($greater(x,y),
                   x, y),
    $let(res := $ite($greater(max,0),
                     $sum(res,max),
                     $diff(res,max)),
      res)))).
```

Program Analysis with Vampire

Code fragment

```
if (x > y) then
    t := x;
    x := y;
    y := t;
```

FOOL with tuples

$$
\begin{array}{l}
\texttt{let } (x, y, t) = \\
\quad \texttt{if } x > y \texttt{ then} \\
\qquad \texttt{let } (x, y, t) = (x, y, x) \texttt{ in} \\
\qquad\quad \texttt{let } (x, y, t) = (y, y, t) \texttt{ in} \\
\qquad\qquad \texttt{let } (x, y, t) = (x, t, t) \texttt{ in } (x, y, t) \\
\quad \texttt{else } (x, y, t) \\
\texttt{in } (x, y, t)
\end{array}
$$

Program Analysis with Vampire

Code fragment

```
if (x > y) then
    t := x;
    x := y;
    y := t;
```

FOOL with tuples

$$
\begin{array}{l}
\texttt{let } (x, y, t) = \\
\quad \texttt{if } x > y \texttt{ then} \\
\qquad \texttt{let } (x, y, t) = (x, y, x) \texttt{ in} \\
\qquad\quad \texttt{let } (x, y, t) = (y, y, t) \texttt{ in} \\
\qquad\qquad \texttt{let } (x, y, t) = (x, t, t) \texttt{ in } (x, y, t) \\
\quad \texttt{else } (x, y, t) \\
\texttt{in } (x, y, t)
\end{array}
$$

Outline

Future Work

- Improved translation of FOOL
- Add tuples to FOOL
- Parser for SMT-LIB syntax
- Support for polymorphism and TFF1