Termination analysis of first order programs

Oleksandr Shturmov

January 22, 2012

Overview

$$H(M,x) = \begin{cases} \textit{true} & M \text{ halts on } x \\ \textit{false} & M \text{ does not halt on } x. \end{cases}$$

$$H(M,x) = \begin{cases} \textit{true} & M \text{ halts on } x \\ \textit{false} & M \text{ does not halt on } x. \end{cases}$$

$$F(M) = \begin{cases} \textit{true} & \texttt{if } H(M,M) \texttt{ then } \textit{false} \\ \textit{false} & \texttt{if } H(M,M) \texttt{ then } \textit{true}. \end{cases}$$

"Unfortunately, many have drawn too strong of a conclusion about the prospects of automatic program termination proving and falsely believe we are always unable to prove termination, rather than more benign consequence that we are unable to always prove termination."

[Cook et al., 2011]

The syntax of Δ

```
          <program> ::= <clause>+ <expression>
       <expression> ::= <element> ( ‘.’ <expression> ) ?
          <element> ::= ‘0’ | ‘(’ <element> ‘)’ | <name> | <application>
      <application> ::= <name> <expression>*
           <clause> ::= <name> <pattern>* ‘:=’ <expression>‘;’
          <pattern> ::= <pattern-element> ( ‘.’ <pattern> ) ?
  <pattern-element> ::= ‘0’ | ‘_’ | ‘(’ <pattern> ‘)’ | <name>
             <name> ::= [‘a’-‘z’] ( [‘-’ ‘a’-‘z’]* [‘a’-‘z’] ) ?
```

(14)

Symbols

Description	**Instance**	**Finite list**	**Space**
Expression	x	X	$\mathbb{X}$
Element (of an expression)	e	E	$\mathbb{E}$
Function	f	F	$\mathbb{F}$
Clause	c	C	$\mathbb{C}$
Pattern	p	P	$\mathbb{P}$
Value (think "binary")	b	B	$\mathbb{B}$
Name (think "variable")	v	V	$\mathbb{V}$
Program (p was taken)	r	R	$\mathbb{R}$

(15)

The syntax of Δ

`<program>`	`::= <clause>+ <expression>`	
`<expression>`	`::= <element> ( ‘.’ <expression> ) ?`	x
`<element>`	`::= ‘0’ | ‘(’ <element> ‘)’ | <name> | <application>`	e
`<application>`	`::= <name> <expression>*`	$\langle v, X \rangle$
`<clause>`	`::= <name> <pattern>* ‘:=’ <expression>‘;’`	$c = \langle v, P, x \rangle$
`<pattern>`	`::= <pattern-element> ( ‘.’ <pattern> ) ?`	p
`<pattern-element>`	`::= ‘0’ | ‘_’ | ‘(’ <pattern> ‘)’ | <name>`	p
`<name>`	`::= [‘a’-‘z’] ( [‘-’ ‘a’-‘z’]* [‘a’-‘z’] ) ?`	v

(14)

Functions in Δ

$$f = \langle v, C \rangle \quad \text{s.t.} \quad \forall \ \langle v_1, P_1, \_ \rangle, \langle v_2, P_2, \_ \rangle \in C \ (v_1 = v_2 = v) \wedge (|P_1| = |P_2|)$$

Pattern matching is ensured **exhaustive** at compile time.

$$\forall\, b \in \mathbb{B}\ \exists\, c \in C\ c \succ b$$

Programs in Δ

$$r = \langle F, x \rangle$$

Demo

Disjoint shapes

$$s_1 \cap s_2 = \emptyset \quad \text{iff} \quad B_1 \cap B_2 = \emptyset$$

where

$$s_1, s_2 \in \mathbb{S} \wedge B_1 = \{b \mid b \in \mathbb{B} \wedge b \succ s_1\} \wedge B_2 = \{b \mid b \in \mathbb{B} \wedge b \succ s_2\}$$

References

[Cook et al., 2011] B. Cook, A. Podelski & A. Rybalchenko, *Proving program termination*, Communications ACM Vol. 54(5), 2011, pp. 88–98.