Termination analysis of first order programs

Oleksandr Shturmov

January 20, 2012

Overview

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>*                                ⟨v, X⟩
     <clause> ::= <name> <pattern>* ':=' <expression>                 c = ⟨v, P, x⟩
    <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.

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