Weakest pre-condition calculus for Lang0

Program Verification - 2012/2013

João Paulo Pizani Flor

Department of Information and Computing Sciences, Utrecht University

Tuesday 19th March, 2013

Universiteit Utrecht

Table of Contents

The Lang0 language datatype

```
1  data L0Value = I Integer | B Bool
2
3  data L0Instruction
4      = SetLocal Int L0Value | LoadLocal Int
5      | StoreLocal Int | LoadParam Int
6      | StoreParam Int | PushLiteral L0Value | Pop
7      | L0Add | L0Sub | L0Mul | L0Div | L0Equ
8      | L0Lt | L0Gt | L0Lte | L0Gte | L0Return
9
10 data L0Statement
11     = Inst L0Instruction
12     | IfThenElse L0Statement L0Statement
13     | Stmts [L0Statement]
14
15 type PName = String
16 type NPar  = Int
17 type NLoc  = Int
18 data L0Program = L0Program PName NPar NLoc L0Statement
```


Universiteit Utrecht

The Spec0 language datatype

```haskell
type Variable = Char

data Sp0Expr
    = Return | Var Variable | Integer Integer
    | Boolean Bool | Param Int | Local Int | Stack Int
    | Add Sp0Expr Sp0Expr | Sub Sp0Expr Sp0Expr
    | Mul Sp0Expr Sp0Expr | Div Sp0Expr Sp0Expr
    | Aeq Sp0Expr Sp0Expr | Lt  Sp0Expr Sp0Expr
    | Lte Sp0Expr Sp0Expr | Gt  Sp0Expr Sp0Expr
    | Gte Sp0Expr Sp0Expr

data Sp0Formula
    = Verum | Falsum | Exp Sp0Expr | Not Sp0Formula
    | And Sp0Formula Sp0Formula
    | Or Sp0Formula Sp0Formula
    | Implies Sp0Formula Sp0Formula
    | Exists Variable Sp0Formula
    | ForAll Variable Sp0Formula
```


Universiteit Utrecht

WP rules for (compound) statements

```
1   wp :: L0Program -> Sp0Formula -> Sp0Formula
2   wp (L0Program _ _ _ body) q = wp_ body q
3
4   wp_ :: L0Statement -> Sp0Formula -> Sp0Formula
5   wp_ (Inst i)          q = wp_inst  i  q
6   wp_ (Stmts ss)        q = wp_stmts ss q
7   wp_ (IfThenElse t e) q = theni `And` elsei
8       where theni   = stackZe     `Implies` wp_ t q
9             elsei   = Not stackZe `Implies` wp_ e q
10            stackZe = Exp $ Aeq (Stack 0) (Integer 0)
11
12  wp_stmts :: [L0Statement] -> Sp0Formula -> Sp0Formula
13  wp_stmts [x] q      = wp_ x q
14  wp_stmts (x:z:zs) q = wp_ x (wp_stmts (z:zs) q)
```


Universiteit Utrecht

WP rules for (some) instructions

```
1   wp_inst :: L0Instruction -> Sp0Formula -> Sp0Formula
2   wp_inst (SetLocal k v) q = rewr (localToLit   k v) q
3   wp_inst (LoadLocal k)  q = rewr (stackToLocal 0 k) q
4   wp_inst (StoreParam k) q = rewr (paramToStack k 0) q
5   wp_inst Pop            q = rewr (stackToStack 0 1) q
6   wp_inst L0Return       q = rewr returnToStack      q
7   ...
8
9   returnToStack :: Sp0Expr -> Sp0Expr
10  returnToStack e = if isReturn e then (Stack 0) else e
11
12  type BinOperator = Sp0Expr -> Sp0Expr -> Sp0Expr
13
14  binOpTransform :: BinOperator -> Sp0Expr -> Sp0Expr
15  binOpTransform op e
16      | isStack e ((==) 0) = op (Stack 0) (Stack 1)
17      | otherwise          = e
```


Universiteit Utrecht

Tools

Programming language Haskell

Theorem prover Z3 (Microsoft Research)

Language bindings The z3 package (on hackage)

- Haskell bindings for Z3
- High-level API
- We need to translate values from our `Spec0` type to `Z3.Lang.Prelude.{Expr,Z3}`