Programmeertaal Pico
(deelopdracht 2, deadline: 16 maart 2011, 24.00u)

Introductie

Pico is een imperatief programmeertaaltje dat twee datatypen heeft: de natuurlijke getallen en strings. De operaties op deze datatypen zijn: concatenatie voor strings en optellen en aftrekken voor de natuurlijke getallen. Aftrekken van natuurlijke getallen wordt gedefinieerd via aftrekken van gehele getallen:

$$m - n = \text{ if } m > n \text{ then } m - n \text{ else } 0$$

Er zijn 3 soorten statements: assignment, if-then-else en while. De guards in de conditionele statements if en while zijn booleans gerepresenteerd door natuurlijke getallen. De natuurlijke getallen worden afgebeeld op de booleans door:

$$\varphi.0 = false,\ \varphi.(n+1) = true$$

Pico heeft geen input/output mechanisme, de statespace wordt opgespannen door de gedeclareerde variabelen met hun waarden. In deze opgave gaan we Pico programma's analyseren via een representatie in Haskell. Voor een formele beschrijving van Pico gebruiken we een contextvrije grammatica, aangevuld met contextcondities. Een mogelijke beschrijving is:

Contextvrije grammatica in EBNF

```
PROG    ::= "begin" DECLS ";" STATLS "end"
DECLS   ::= "declare" IDLS
IDLS    ::= ε | IDTS
IDTS    ::= PICO-ID ":" TYPE ( "," PICO-ID ":" TYPE)*
TYPE    ::= "natural" | "string"
STATLS  ::= ε | STATS
STATS   ::= STAT (";" STAT)*
STAT    ::= VAR ":=" EXPR
STAT    ::= "if" EXPR "then" STATLS "else" STATLS "fi"
STAT    ::= "while" EXPR "do" STATLS "od"
EXPR    ::= NatCon
EXPR    ::= StrCon
EXPR    ::= VAR
EXPR    ::= EXPR "+" EXPR
EXPR    ::= EXPR "-" EXPR
EXPR    ::= EXPR "||" EXPR
EXPR    ::= "(" EXPR ")"
```

```
VAR      ::= PICO-ID
PICO-ID::= CHAR (CHAR | DIGIT)*
CHAR   ::= "a" | "b" | ... | "z"
DIGIT   ::= "0" | "1" | ... | "9"
```

De nonterminals NatCon resp. StrCon genereren op niet nader gespecificeerde wijze de natuurlijke getallen resp. strings.

Contextcondities

c1: Variabelen moeten gedeclareerd worden en verschillen van de reserved words.

c2: Alle namen in een declaratielijst moeten verschillend zijn.

c3: Gedeclareerde variabelen worden default geinitialiseerd: natuurlijke getallen op 0, strings op [].

c4: De terminals "+" resp. "-" staan voor optellen resp. aftrekken van natuurlijke getallen. De operanden en het resultaat van optelling en aftrekking moeten van type $\mathbb{N}$ zijn.

c5: De terminal "||" staat voor string concatenatie. De operanden en het resultaat van concatenatie moeten van type String zijn.

c6: De guards in de conditionele statements if en while moeten van type $\mathbb{N}$ zijn.

Voorbeeld van Pico programma (pico1)

```
begin  declare in : natural,
               out : natural,
               a : natural,
               b : natural,
               h : natural;
       in := 11;
       a := 1;
       while in do
           h := a;
           a := b;
           b := h + b;
           in := in-1
       od;
       out := b
end
```

Haskell environment voor Pico programma's

```
module Pico where
type Prog = ([Decl],[Stat])
type Decl = (Name, Type)
type Name = String

data Type = N | S deriving (Show, Eq)

-- waarden bij type N en S
data Val  = N1 Integer | S1 String deriving (Show,Eq)

data Stat = Assign Name Expr
          | If Expr [Stat] [Stat]
          | While Expr [Stat]
          deriving (Show,Eq)

data Expr = NCon Integer
          | SCon String
          | Var Name
          | Bin Op Expr Expr
          deriving (Show,Eq)

data Op   = Plus | Min | Cat deriving (Show,Eq)

type State = [(Name,Val)]

-- aan te vullen met opdrachten
```

Haskell representatie van het voorbeeld programma

```
pico1 = (dl,sl)
dl = [("in",N), ("out", N), ("a", N), ("b", N), ("h", N)]
sl = [ Assign "in" (NCon 11)
     , Assign "a" (NCon 1)
     , While (Var "in")
         [ Assign "h" (Var "a")
         , Assign "a" (Var "b")
         , Assign "b" (Bin Plus (Var "h") (Var "b"))
         , Assign "in" (Bin Min (Var "in") (NCon 1))
         ]
     , Assign "out" (Var "b")
     ]
```

Opdrachten

Ga uit van één van de twee programma's **picoStart.lhs** en **picoStart.hs**. In de ".lhs" versie is het mogelijk tekst en code in een document aan het systeem aan te bieden, in de ".hs" versie moet tekst via commentaar worden ingevoegd.
Lever het aangevulde programma vóór de deadline in via PEACH.

1. Kennismaken met Pico

In dit onderdeel gaan we na hoe Pico operaties uitgebreid zouden kunnen worden.
Bij de opdrachten **1**.2, **1**.4 en **1**.6 wordt een wiskundige notatie verwacht.

1. Bestudeer het voorbeeldprogramma en geef het verband aan tussen de variabelen "in" en "out".

2. Definieer vermenigvuldiging van 2 natuurlijke getallen in termen van optelling. Geef ook een geaccumuleerde versie.

3. Geef een Pico programma voor de vermeningvuldiging van 13 en 37.
   Geef de bijbehorende Haskell representatie van dit Pico programma.

4. Geef een geaccumuleerde versie voor de berekening van n!

5. Schrijf een Pico programma voor de berekening van 13! (**pico2**).
   Geef de bijbehorende Haskell representatie van dit Pico programma.

6. Definieer deling van 2 natuurlijke getallen (deler 0 uitgezonderd) in termen van aftrekken. Geef ook een geaccumuleerde versie.

7. Geef een Pico programma voor de deling van 1311 door 23 (**pico3**).
   Geef de bijbehorende Haskell representatie van dit Pico programma.

2. Programma analyse

In dit onderdeel gaan we (Haskell versies van) Pico programma's analyseren m.b.v. Haskell programma's.

1. De gebruikte variabelen in een programma zijn alle variabelen die voorkomen in de programma body (de statementlist die volgt op de declaratielist). Geef een inductieve definitie ter berekening van de gebruikte variabelen. Geef ook een geaccumuleerde versie.

2. Geef een programma ter controle van contextconditie **c1**.

3. Geef een inductieve en een geaccumuleerde versie voor contextconditie **c2**.

4. Typecheck een Pico programma, dus controleer of aan de contextcondities **c4**, **c5** en **c6** voldaan is.

5. Voor een gegeven statement: geef een algoritme dat berekent hoeveel assignments erin voorkomen, en welke variabelen daarbij betrokken zijn.

6. Voor een gegeven programma p bereken in one-pass het 3-tuple (nrA, nrI, nrW) waarbij nrA, nrI en nrW respectievelijk het aantal assignments, if-statements en while-statements van p voorstellen.

7. Er zijn diverse metrieken om de complexiteit van een programma uit te drukken. Een ervan is de MacCabe complexiteit: i.e. het aantal guards in het programma. Bereken de MacCabe complexiteit bij een gegeven programma.

3. Evaluator voor Pico

In dit onderdeel gaan we (Haskell versies van) syntactisch correcte Pico programma's die aan de contextcondities voldoen, evalueren. Dat wil zeggen: gegeven de initiële state (lijst met gedeclareerde variabelen en hun waarden) kan de state berekend worden die bereikt wordt door programma-executie.

1. Definieer een evaluator voor Pico programma's in Haskell

2. Controleer de evaluaties van de programma's uit opdrachtenserie **1**.