Alphi

Naam: Lissens
Voornaam: Tobiah
Richting: 2de bachelor Informatica
year: 2016-2017

Inhoud

Inleiding

In dit project wordt de eenvoudige programmeertaal Alphi opgesteld. Hierbij is het de bedoeling verschillende basiselementen van een imperatieve programmeertalen te implementeren. zoals bv: toekenning, variablen en volgorde van bewerkingen. De taal die hieronder wordt uitgewerkt heet Alphi wat staat voor alphanumerical. Deze taal maakt enkel gebruik van alphanumerische karakters met de uitzondering dat whitespace ook is toegestaan. Eerst zal de syntax worden vastgelegd. Vervolgens wordt de semantiek vastgelegd en worden voorbeeld programma's gegeven. Hierna worden de implementatie aspecten besproken.

Syntax

BNF notatie van de Alphi taal

```
Pgm  = Stmt

Stmt    = <Var> "Is" <Exp> "Stop"
        | "Command" <Output> <Exp> "Stop"
          <Stmt> <Stmt>
          "If" <Exp> "Begin" Stmt "End"
          "While" <Exp> "Begin" Stmt

Exp    ::= <BExp>
         | <NExp>
         | "Command" <Input>

NExp    = <Num>
           <NVar>
           <NExp> "Add" <NExp>
           <NExp> "Sub" <NExp>
         | <NExp> "Mul" <NExp>
         | <NExp> "Div" <NExp>
         | "Open" NExp "Close"

BExp    = <Bool>
         | <BVar>
            "Not" <BExp>
           <NExp> "Gt" <NExp>
         | <NExp> "Lt" <NExp>
         | <NExp> "Eq" <NExp>
           <BExp> "And" <BExp>
         | <BExp> "Or" <BExp>
         | "Open" BExp "Close"

Input  ::= "OpenMBot"
         | "CloseMBot"
         | "SensorR"
         | "SensorL"
         | "Ultra"

Output ::= "Print"
         | "MotorR"
         | "MotorL"
         | "Led1"
         | "Led2"

Bool     = "True" | "False"
Num      =  Int | Float
Int     ::= ["0"-"9"]+
Float   ::= <Int>"Point"<Int>
```

```
Var      = NVar | BVar
NVar    ::= "N"<Letter>+
BVar    ::= "B"<Letter>+
Letter  ::= ["a"-"Z"]
```

Semantiek

Korte omschrijving van wat wat is. 1. Expressies 1. Numeric

```
  Volgorde van bewerkingen
Voor bewerkingen op hetzelfde niveau wordt van links naar rechts geevalueerd.
  Hoe hoger het niveau hoe eerder ze geivalueerd worden

  Niveau 1(literalen)
  Float, Int
  Vb: 10Point4, 4

  Niveau 2
  Add, Sub

  Niveau 3
  Mul, Div, Mod

  Niveau 4 (haakjes)
  Open Close

  vb: 4 Sub 4 Mul 4 = -12
      4 Mul 4 Mul 2 Sub 3 = 29
```

2. Boolean

```
  Volgorde Van bewerkingen
  Voor bewerkingen op hetzelfde niveau wordt van links naar rechts geevalueerd.
  Hoe hoger het niveau hoe eerder ze geivalueerd worden

  Niveau 1
  True, False

  Niveau 2
  And, Or

  Niveau 3
  Gt, Lt , Eq

  Niveau 4
  Not
```

```
Niveau 5 (Haakjes)

vb: True And True            = True
    4 Lt 5 And False         = False
    4 Lt 5 And True          = True
    4 Lt 5 And False Or True = True
```

3. Commands
4. Input
    1. SensorL/SensorR

```
Gebruik: Lees een booleanaanse waarde uit de linkse/rechtse lichtsensor.
False = sensor ziet wit
True  = sensor ziet zwart
Vb: lees waarde uit de linkse lichtsensor.
Bvalue Is Command SensorL Stop
```

    2. Ultra “‘ Gebruik: Lees een numerische waarde uit de ultrasonesensor False = sensor ziet wit True = sensor ziet zwart Vb: lees waarde uit de ultrasonesensor. Nvalue Is Command Ultra Stop

4. OpenMBot/CloseMBot

```
Gebruik: Open/Sluit de connectie met de robot.
Vb: Maak robot klaar voor communicatie
Command OpenMBot Stop
```

2. Output
    1. Print

```
Gebruik: Print een expressie uit naar standaard out.
Vb: Toon 3.
Command Print 3 Stop
```

    2. MotorR/MotorL

```
Gebruik: Zet motor aan met snelheid [-255,...,255]
Vb: Rij vooruit met snelheid 100
Command MotorR 100 Stop
Command MotorL 100 Stop
```

    3. Led1/Led2

```
Gebruik: Geef Led1/Led2 kleur
1 -> rood
2 -> groen
3-> blauw
```

```
Vb: Geef led1 kleur Groen
Command Led1 2 Stop

3. Statements

  1. AssignStatement

Gebruik: Ken een waarde aan een variable Toe

Vb: Zet variable Tobiah op 20
NTobiah Is 20 Stop

  2. While

Gebruik: Herhaalt de statements tussen Begin en End tot de Expression voor Begin naar False evalu

Vb: Programma dat tot 10 telt.
Nx IS 0 Stop
While Nx Lt 10 Begin
    NX Is NX Add 1 Stop
End

  3. If
```

Gebruik: Voert Stuk code tussen Begin en End uit indien de Experessie voor Begin naar True evalueert.

Vb: Programma dat bij oneven getallen 1 optelt. If Nvar Mod 2 Eq 1 Begin Nvar Is Nvar Add 1 Stop End

```
## Programma's
Korte beschrijvingen van het programma
1. demo_police.alp (zie Appendix Broncode)
   Start teller.
   Indien teller even zet Led1 op rood en led 2 op blauw.
   Indien teller oneven zet led2 op rood en led1 op blauw.
   Verhoog Teller met 1
   Begin bij stap 2.

2. demo_line.alp (zie Appendix Broncode)
  Lees beide lichtsensoren uit.
  Indien beide sensoren Zwart zien rij de robot rechtdoor.
  Indien links wit ziet en rechts zwart draai alleen de linker motor.
  Indien rechts wit ziet en links zwart draai alleen de rechter motor.
  Indien Beide wit zien rij achteruit.
  Begin terug bij stap 1.

3. demo_ultra.alp (zie Appendix Broncode)
```

Lees Ultrasonesensor uit.
Indien afstand Groter dan 40 rij rechtdoor.
Indien afstand Kleiner dan 40 draai de linkermotor vooruit en de rechtermotor achteruit.
Begin terug bij stap 1.

Implementatie

Hier worden kort de interessante functies aangeraakt.

1. Parsen (Parser)

1. Base.hs
Hier werd de Parser monad geimplementeerd het grootste deel van de code komt uit de slides over
Wel belangrijk te noteren dat de option uit alternative and de mplus uit de monadplus anders ge

Monadplus:
Zie Parser.Base line?
mplus p1 p2 = probeer parser 1 en ook parser2.

Alternative:
Zie Parser.Base line?
option p1 p2 = indien parser 1 faalt probeer parser2.

2. Util.hs
Hier werden alle Parser functies geimplementeerd die nergens anders een plaats hadden.

3. NumericalParser.hs
Hier staan alle numerical expressie parsers.

4. BooleanParser.hs
Hier staan alle boolean expressie parsers.

5. StatementParser.hs
Hier staan alle Statemenparsers en dit is dus eveneens de programma parser.

2. Evalueren (Evaluator)
Bij het evalueren word er gebruik gemaakt van een StateT monad transformer waarin een IO monad z

1. NumericEval.hs / BoolEval.bs

Het idee Hierbij is te pattern matchen op de datastructuur En op deze manier kunnen we elk geva
Voor functies die veel voorkomen word een abstractere versie aangemaakt in Evaluator.Util.hs
een mooi voorbeeld hiervan is de functie EvalBOp zie Evaluator.Util.hs line ?.
EvalBOp is een functie die 6 argumenten neemt.

1) Functie die 2 a's binnen neemt en een a teruggeeft
2) Expressie1 een expressie
3) Expressie2 een expressie
4) Evaluator1
5) unwrapper (m a -> a) functie die een return value unwrapped
6) Constructor Wrapper constructor

Deze functie zal de 2 expressies uitrekenen de return waarden daarvan uitpakken de functie erc

2. StatementEval:

3. RobotLib (Robot.Base.hs)
Hier werd verder gewerkt op de gegeven library zodat er intuitiver gewerkt kan worden gewerkt m
En zodat er een mooi scheiding kan blijven bestaan tussen de robotaansturing en de taal.

Een Intressante functie is de move functie hierbij wordt een device snelheid en motor meegegeve
zodat de motor makkelijker kan aangestuurd worden. De Implementatie kan gevonden worden onder

Conclusie
1. Algemeen:
Een alphanumerical taal maken leek in het begin leuk. Dit bracht echter enkele nadelen met zich m
Het groote nadeel hieraan is dat je geen speciale karakters hebt die kunnen instaan voor bv het e
Verder wordt de taal ook Enorm rap onduidelijk en on leesbaar doordat er weinig tot geen ondersch

2. Syntax definitie:
Hierbij zijn er soms onlogische samenstellingen mogelijk zoals Numerical Expression in de IfCond
Hierdoor moet dit opgevangen worden tijdens het evalueren dit is ongewild.

3. Implementatie:
De parseLibrary is vrij onduidelijk geschreven er onbreekt een mooie volgbare hierachy die bv we

Dit komt voornamelijk doordat er geen eenduidige manier was om dingen te parsen En er op ieder mo

Appendix Broncode:

Inhoud

AlphiExamples

1. demo_police.alp

```
0 commentOpen
1
2 A simple police sirene program
3
4 commentClose
5
6
7 Command OpenMBot Stop
8
9 Ncount Is 0 Stop
10 While True  Begin
11
12   If Ncount Mod 2 Eq 0 Begin
13     Command Led1 1 Stop
14     Command Led2 3 Stop
15   End
16
17   If Ncount Mod 2 Eq 1 Begin
```

```
18     Command Led1 3 Stop
19     Command Led2 1 Stop
20   End
21
22   Ncount Is Ncount Add 1 Stop
23
24 End
25 Command CloseMBot Stop
```

2. demo_line.alp

```
0 commentOpen
1
2 A simple linefollowing program
3
4 commentClose
5
6
7 Command OpenMBot Stop
8 While True Begin
9
10   Bleft Is Command SensorL Stop
11   Bright Is Command SensorR Stop
12
13   If Bleft And Bright Begin
14     Command MotorL 70 Stop
15     Command MotorR 70 Stop
16   End
17
18   If Bleft And Not Open Bright Close Begin
19       Command MotorL 0 Stop
20       Command MotorR 80 Stop
21   End
22
23   If Bright And Not Open Bleft Close Begin
24       Command MotorL 80 Stop
25       Command MotorR 0  Stop
26   End
27
28   If Not Open Bright Or Bleft Close Begin
29     Command MotorL 0 Sub 60 Stop
30     Command MotorR 0 Sub 60 Stop
31   End
32 End
33 Command CloseMBot Stop
```

3. demo_ultra.alp

```
0 commentOpen
1
2 A simple wall evade program
3
4 commentClose
5
6
7 Command OpenMBot Stop
8
9 While True Begin
10   Ndistance Is Command Ultra Stop
11
12   If Ndistance Gt 40 Begin
13     Command MotorL 70 Stop
14     Command MotorR 70 Stop
15   End
16
17   If Ndistance Lt 39 Begin
18     Command MotorL 70 Stop
19     Command MotorR 0 Sub 70 Stop
20   End
21   Command Print Ndistance Stop
22 End
```

‘‘‘

Src