Lessenreeks Haskell: Project

Tom Schrijvers, Pieter Vander Vennet

March 20, 2015

1 Inleiding

Live coding heeft tot doel om de programmeur ogenblikkelijke feedback te geven over het programma dat hij schrijft. Hier zijn een aantal interessante voorbeelden van de visionaire Brett Victor:

- *Inventing on Principle*, `http://vimeo.com/36579366`
- *Learnable Programming*, `http://worrydream.com/LearnableProgramming/`

2 Het Project

In dit project gaan we een eenvoudige tool voor live coding maken. De tool leest een programma geschreven in de programmeertaal While in uit een tekstbestand, en drukt het programma met rechts ervan voorbeeldgedrag.

Hier is een voorbeeld van mogelijke invoer:

prog1.txt

```
x = 3;
y = 0;
while (x > 0) {
  x = x - 1;
  y = y + 1
}
```

uitvoering:

```
$ ./Main prog1.txt > output.txt
```

en overeenkomstige uitvoer:

output.txt

```
x = 3;            :   x = 3
y = 0;            :   y = 0
while (x > 0) {   :   x = 3 | 2 | 1
```

```
                    :   y = 0 | 1 | 2
  x = x - 1;        :   x = 2 | 1 | 0
  y = y + 1         :   y = 1 | 2 | 3
}                   :
```

Dit is maar een voorbeeld van een mogelijke weergave. Je bent vrij om zelf de weergave en het uitvoerformaat (bv. plain text, HTML, ...) te kiezen. De essentie moet wel aanwezig zijn in je uitvoer: de programma-code en de overeenkomstige waarden van de variabelen.

2.1 De While Taal

Syntax De syntax van WHILE programma's ziet er als volgt uit.

Een programma is een *statement*. Een statement is ofwel een toekenning, ofwel een `while`-lus ofwel een sequentie van twee statements.

$$\begin{array}{rcll} s & ::= & v = e & \textit{assignments} \\ & | & \texttt{while}(c)\{s\} & \textit{loops} \\ & | & s_1; s_2 & \textit{sequences} \end{array}$$

Een *expressie* is ofwel een constante, een variabele, een optelling, een aftrekking, een vermenigvuldiging of een expressie tussen haakjes. De prioriteiten van de operatoren zijn de gebruikelijke: haakjes hebben voorrang op $*$, en $*$ op $+$ en $-$.

$$\begin{array}{rcll} e & ::= & n \quad (n \in \mathbb{N}) & \textit{literals} \\ & | & x & \textit{variables} \\ & | & e + e & \textit{additions} \\ & | & e - e & \textit{subtractions} \\ & | & e * e & \textit{multiplications} \\ & | & (e) & \textit{groupings} \end{array}$$

Er is slechts één vorm van *conditionele expressie*:

$$c \quad ::= \quad e > e \qquad\qquad \textit{greater-thans}$$

Je mag ervan uitgaan dat alle invoerprogramma's voldoen aan deze syntax.

Semantiek De operationele semantiek van de taal is de gebruikelijke, imperatieve. Je moet geen rekening houden met randgevallen zoals integer overflow, ongeinitialiseerde variabelen of oneindige lussen.

3 Functionaliteit

We schetsen de mogelijke opsplitsing in een aantal deelfunctionaliteiten:

- Een tekstbestand uitlezen.
- Tekst parsen naar een abstract syntax tree.
- Evaluatie van het programma:
  - Definieer een *programmatoestand* als een associatie van variabelen met waarden.
  - Definieer evaluatie van statements.
  - Associeer met elk statement een programmatoestand.
- Pretty printen van het programma met geassocieerde toestanden.

4 Hints

Je mag gebruik maken van bibliotheken. Hoewel zelf een parser en pretty-printer schrijven een uitstekende oefening is, kan je een parse- en html-bibliotheek gebruiken.

5 Evaluatie

Je project zal worden bekeken en feedback erop zal je toegestuurd worden. Je krijgt geen punten, aangezien dat deze enkel nuttig zijn om op een rapport te zetten. Bij gebrek aan rapport, geen punten :p . Let echter wel op:

- correcte werking
- verzorgdheid van de code
- gebruik van de meest geschikte Haskell-features
- grote "woordenschat": diversiteit aan Haskell-features en library-functies
- verzorgdheid van de uitvoer

Om het maximum qua punten te halen, lever je extra inspanning (*) op een of meer van de volgende gebieden:

- Je genereert zeer complexe uitvoer.
- Je formuleert enkele relevante stellingen over functies die je geschreven hebt, en bewijst ze of voorziet QuickCheck testen.
- ...

6 Indienen

Je werkt het liefst individueel, op die manier ga je het meeste leren. Er is echter niets mis met eens te spieken en van een ander bij te leren.

Je kan indienen door een github te maken en deze door te sturen naar pieter.vandervennet@ugent.be

Indien je geen github-account wenst te maken, stuur je een zip.

Je inzending bestaat uit een `zip`-bestand met volgende inhoud:

- `README.txt`: Een opsomming van de bijkomende bestanden in de `zip` file met hun functie, en een opsomming van de extra's (*) in je project.
- `Main.hs`: Hoofdmodule van je project. Compilatie van dit bestand met `ghc --make Main.hs` genereert de binary executable.
- Eventuele andere bestanden nodig voor je project.

7 Vragen, Opmerkingen en Voorbeeldprogramma's

Met al je inhoudelijke vragen en opmerkingen kan je terecht op de facebook-groep: https://www.facebook.com/groups/443970042422378/

Verdere woensdagen worden ook voorzien om aan het project te werken, dus dan kun je ook al je vragen stellen.

Als je voorbeeldprogramma's in WHILE schrijft, wees dan solidair en deel ze. Je vindt al een paar voorbeelden op https://github.com/pietervdvn/While-tool. De voorbeeld output (Example.html) was van mijn project2 jaar geleden, en is zeker niet bindend.