Fortgeschrittene Funktionale Programmiernug in Haskell

Projekt: Entwicklung einer Programmiersprache in Haskell

Tutoren: Jonas Betzendahl (`jbetzend@techfak...`), Stefan Dresselhaus (`sdressel@techfak...`)

Aufgabenstellung:

Ziel dieses Projektes ist die Implementation einer eigenen DSL (**D**omain **S**pecific **L**anguage) zur Erstellung von Turtle Graphics[1], ähnlich der Programmiersprache ***Logo***.

Mindestanforderungen:

Ihre Abgabe soll in der Lage sein, eine Datei mit Befehlen für die Schildkröte (s.u.) einzulesen, zu parsen, in eine interne Repräsentation zu überführen und daraus entsprechenden Output zu generieren.

Sie können mit einer Text-Only Ausgabe beginnen, die nur auf `stdout` printet, was die Schildkröte tun würde. In der Endabgabe muss allerdings eine grafische Ausgabe generiert werden. Die Wahl der Bibliothek für diese Zwecke (`sdl2`, `gtk2hs`, ...) ist Ihnen frei gestellt.

Wenn Sie wünschen, können Sie anstatt einer grafischen Oberfläche auch nur ein Bild ausgeben (z.B. ein `.png` oder ein animated `.gif`). Falls Sie sich dazu entscheiden, implementieren Sie außerdem den Operator `<%>` aus den Zusatzaufgaben.

Ihre Programmiersprachen muss zur Abgabe mindestens die folgenden Befehle unterstützen:

forward n: bewegt die Schildkröte n Einheiten vorwärts
turn d: dreht die Schildkröte d Grad im Uhrzeigersinn
die: „tötet“ die Schildkröte (hält das Programm an)
forever: führt ein Programm endlos immer wieder aus
color r g b oder **color c**: Verändert die Farbe (Farbwerte oder Name)
penup/pendown: hebt/senkt den Stift sodass (nicht) gezeichnet wird

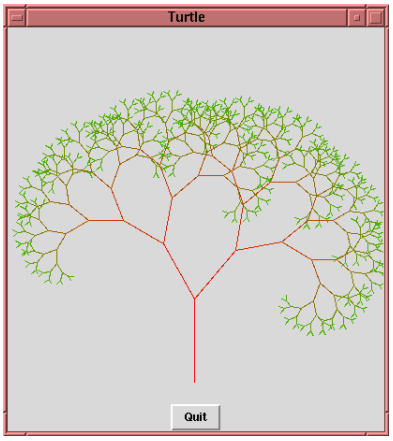

Abbildung 1: Ein möglicher Output der Turtle. Credit: *Chalmers University*

Sie werden für diese außerdem einen Kombinator wie `>%>` implementieren müssen. Dieser führt zwei Programme hintereinander aus. So können sie auch einzelne Befehle zu komplexeren Programmen zusammenfügen. Welche der oben aufgeführten Befehle lassen sich auf eine Kombination von simpleren Befehlen zurück führen?

Zusatz:

Bei besonderer Motivation können außerdem die folgenden Features noch eingebaut werden:

- **Parallel Turtles:**
  Implementieren sie den Operator `<%>` analog zu `>%>`. Dieser startet beide Programme parallel (es gibt also danach eine zusätzliche Turtle).
- **Schildkröten-Funktionen:**
  Geben Sie Ihrer Sprache die Möglichkeit, Funktionen auszudrücken (s. Code rechts), die dann wie Befehle aufgerufen werden können.

```
to spiral :size :angle
  if :size > 100 [stop]
  forward :size
  right :angle
  spiral :size + 2 :angle
end
```

Abgabemodalitäten:

Eine gültige Abgabe ist ein `cabal`-Projekt, das fehlerfrei in einer Sandbox installiert werden kann und eine funktionierende ausführbare Datei generiert (Testumgebung ist im Zweifelsfall wie immer das GZI). Bitte reichen Sie Ihre Projekte spätestens bis zum **Freitag, den 18.09.2015** ein. Dazu schicken Sie alle Dateien, die zu Ihrem Projekt gehören (eventuell modulo einer vernünftigen `.gitignore`) in einem Dateiarchiv an beide Tutoren.

Falls gewünscht, kann Ihnen für die Entwicklung des Projekts ein privates Repository auf `GitHub` zur Verfügung gestellt werden. Dann kann auch direkt dort abgegeben werden. Kontaktieren Sie dafür bitte die Tutoren.

Sollten Sie Rückfragen haben oder Hilfestellung benötigen, wenden Sie sich bitte ebenfalls an die Tutoren.

[1] `http://en.wikipedia.org/wiki/Turtle_graphics`