Christoph Lüth
Dominik Dietrich
Marcus Ermler
Klaus Hartke
Christian Maeder
Dominik Luecke
Ewaryst Schulz

5. Übungsblatt

Ausgabe: 12.01.09
Abgabe: 26.01.09

11 *Suchet, so werdet ihr finden* *7 Punkte*

In dieser Teilaufgabe geht es um einfache Textsuche. Wir wollen innerhalb einer Zeichenkette nach einem oder mehreren Schlüsselbegriffen (also einfach anderen Zeichenketten). Präzise gesagt ist eine Suchanfrage eine Menge von Schlüsselwörtern, entweder durch Konjunktion oder Disjunktion verbunden (wobei Konjunktion stärker bindet). Beispiele für Suchanfragen sind

```
foo & baz
oh no | yes
foo & baz  | bar
```

Modellieren Sie Suchanfragen durch einen algebraischen Datentyp `Query`, und implementieren Sie Funktionen, welche einen `Query` als Zeichenkette in der obigen Syntax einlesen und ausgeben. Leerzeichen am Anfang und Ende von Schlüsselwörtern sollen ignoriert werden.

Eine Suchanfrage ist für einen gegeben Text erfolgreich, wenn

- für `p | q` die Anfrage `p` oder `q` erfolgreich sind,
- für `p & q` sowohl `p` als auch `q` erfolgreich sind,
- und für ein Wort `w` das Wort in dem Text vorkommt (ohne Berücksichtigung von Groß/Kleinschreib

Von den Anfragen oben sind also für den Text `oh no, foo bar!` die letzten beiden erfolgreich (`oh no` gefunden, `bar` gefunden). Die Reihenfolge spielt bei Konjunktion oder Disjunktion keine Rolle (also wäre für `baz foo` die dritte Anfrage erfolgreich).

Implementieren Sie eine Funktion

```
query :: Query-> String-> Bool
```

welche prüft, ob eine Suchanfrage für einen gegebenen Text erfolgreich ist.

12 *Suche I: `hgrep`* *3 Punkte*

Mit Hilfe der Funktion `query` implementieren Sie ein Program, welches (ähnlich wie die Programme aus der `grep`-Familie) eine Datei nach einer Suchanfrage durchsucht.

Das zu entwerfenden Programm `hgrep` sollte die Suchanfrage als erstes Argument, sowie eine Liste von Dateien als weitere Argumente erhalten. Es liest die angegebenen Dateien, sucht dort alle *Zeilen*, welche der Suchanfrage entsprechen, und gibt im Erfolgsfall den Dateinamen, die Zeilennummer, und die Zeile aus:

```
# ./hgrep 'baz & bar | foo' *.*
test.hs(12): foo :: Int-> IO Int
test.hs(13): foo x = do
slides.tex(23): such as the words \texttt{bar} and \texttt{baz}, which
```

Hinweise: Beachten Sie beim Aufruf des Programmes, dass die Shell Muster der Form `*.tex` expandiert, d.h. das in Haskell geschriebene Program muss die Expansion der Muster nicht selber vornehmen. Achten Sie ferner darauf, die Suchanfrage in Hochkommata `'` einzuschließen, damit sie von der Shell nicht interpretiert werden.

13 *Suche II: `webgrep`* *10 Punkte*

In der nächsten Stufe wollen wir das WWW durchsuchen (mit der Idee sind wir nur circa zehn Jahre hinter der Zeit). Dazu verwenden wir das in der Vorlesung vom 14.01. vorgestellte Programm zum Lesen von RSS-Feeds.

Das Programm sollte aus der Kommandozeile aufgerufen werden. Es liest eine Datei namens `feeds.rss`, welches pro Zeile eine Adresse einer RSS-Quelle enthält, und eine Datei `query`, welche eine Suchanfragen erhält.

Dsa Programm parsiert die in `feeds.rss` angegeben Quellen, durchsucht Beschreibungen und Titel der dort gefundenen `item` nach den Suchanfrage, und gibt ein Suchergebnis zurück, welches durch folgenden Datentyp modelliert wird:

```
type Results = [Result]
data Result  = Result String -- Title of item
                      String -- Link of item
                      String -- Description of item
                      String -- Name of channel it appeared in
```

Ein Resultat soll den Titel und den Link auf das gefundene `item` enthalten, dessen Beschreibung, sowie den `channel`, in dem dieses Item gefunden wurde.

Das Ergebnis soll als XHTML-Seite dargestellt werden (wie in der Vorlesung gezeigt). Ein Beispiel für eine Ergebnisdarstellung findet sich auf der Webseite.