P. Thiemann, S. Wehr Wintersemester 2006/2007

Compilerbau

http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2006ws/

Übungsblatt 2

8.11.2006

Aufgabe 1 (2 Punkte)
Berechne $D(\underline{d}\underline{d}^*, d)$ und $D((\underline{\varepsilon}|\underline{-})\underline{d}\underline{d}^*, d)$. Dabei sei $D : RE(\Sigma) \times \Sigma \to RE(\Sigma)$ wie in der Vorlesung definiert. Wende auch die Vereinfachungsregeln auf das Ergebnis an.

Aufgabe 2 (6 Punkte)
In der Vorlesung wurde die Ableitung eines regulären Ausdrucks $r \in RE(\Sigma)$ bezüglich eines Symbols $a \in \Sigma$ als Funktion $D : RE(\Sigma) \times \Sigma \to RE(\Sigma)$ zusammen mit einer Hilfsfunktion $E : RE(\Sigma) \to RE(\Sigma)$ definiert. Beweise:

(i) $L(E(r)) = L(r) \cap \{\epsilon\}$

(ii) Für alle $a \in \Sigma$ gilt: $w \in L(D(r, a))$ genau dann wenn $aw \in L(r)$.

(iii) $L(r) = L(E(r)) \cup \bigcup_{a \in \Sigma} aL(D(r, a))$

In der letzten Gleichung wird $aL(D(r, a))$ als Abkürzung für $\{a\} \cdot L(D(r, a))$ verwendet.

Aufgabe 3 (4 Punkte)
Implementiere die in der Vorlesung vorgestellte Funktion `accepts_empty : 'a regexp -> bool` in OCaml.

Aufgabe 4 (4 Punkte)
Scannergeneratoren, wie etwa Lex, benutzen als Beschreibungssprache für Tokens in der Regel nicht nur einfache reguläre Ausdrücke, sondern erweitern diese um zusätzliche Operatoren. Die folgende Tabelle zeigt die erweiterte Beschreibungssprache von Lex (wobei c für ein beliebiges Zeichen, r für einen regulären Ausdruck und s für eine Zeichenkette steht):

Ausdruck	Bedeutung	Beispiel
c	bel. (Nichtsonder-)Zeichen c	`a`
`\`c	Zeichen c	`\*`
`"`s`"`	Zeichenkette s	`"**"`
`.`	bel. Zeichen außer *newline*	`a.*b`
`^`	Zeilenanfang	`^abc`
`$`	Zeilenende	`abc$`
`[`s`]`	bel. Zeichen aus der Klasse s	`[abc]`
`[^`s`]`	bel. Zeichen nicht aus der Klasse s	`[^abc]`
r`*`	null oder mehr rs	`a*`
r`+`	ein oder mehr rs	`a+`
r`?`	null oder ein r	`a?`

$r\{m,n\}$	m bis n Wiederholungen von r	`a{1,5}`
$r_1 r_2$	erst r_1 dann r_2	`ab`
$r_1 \vert r_2$	r_1 oder r_2	`a|b`
(r)	r	`(a|b)`
r_1/r_2	r_1 falls gefolgt von r_2	`abc/123`

(i) Die Spezialbedeutung der Operatorsymbole (`\ " . ^ [ ] * + ? { } | /`) muss ausgeschaltet werden, falls diese selber übereinstimmen sollen. Es gibt zwei Möglichkeiten, dies zu bewerkstelligen. Falls kein `"`-Zeichen in einer Zeichenkette s vorkommt, kann diese mit `"`s`"` exakt auf Übereinstimmung überprüft werden (etwa in `"**"`). Alternativ können die Zeichen separat mit Hilfe von `\` angegeben werden (etwa in `\*\*`).

Gib einen regulären Ausdruck für Lex an, der die Zeichenkette `"\` beschreibt.

(ii) Eine Zeichenklasse, die mit dem Symbol `^` beginnt, beschreibt eine *Komplementär-Zeichenklasse*. Mit einer Komplementär-Zeichenklasse erkennt man alle Zeichen, die *nicht* in einer bestimmten Zeichenklasse enthalten sind (`[^a]` etwa beschreibt alle Zeichen außer `a`).

Zeige, dass zu jedem regulären Ausdruck mit Komplementär-Zeichenklassen ein äquivalenter regulärer Ausdruck ohne Komplement-Zeichenklassen existiert.

(iii) Der reguläre Ausdruck $r\{m,n\}$ erkennt m bis n Wiederholungen des Musters r. Beispielsweise erkennt `a{1,5}` Zeichenketten mit ein bis fünf `a`s.

Zeige, dass zu jedem regulären Ausdruck mit Wiederholungsoperator ein äquivalenter regulärer Ausdruck ohne Wiederholungsoperator existiert.

Abgabe: 15.11.2006
Die Abgabe erfolgt vor Beginn der Übungsstunde. Code soll in gedruckter Form abgegeben werden. Für Plagiate werden keine Punkte vergeben.