PHILIPPS-UNIVERSITÄT MARBURG D-35032 Marburg
Fachbereich Mathematik und Informatik Hans Meerwein Straße
Prof. Dr. R. Loogen
J.Berthold

15.Dezember 2005

8. Übung zu „Grundlagen des Compilerbaus“, WS 2005/06

Abgabe schriftlicher Aufgaben: Do, 12.Januar 2006 (vor der Vorlesung)
Besprechung mündlicher Aufg.: ab 19.Dezember 2005 in der Übung

---

Hinweise: Die Bearbeitungszeit für dieses Blatt ist verlängert und erstreckt sich über die Weihnachtspause. Daher sind statt 12 Punkten 18 Punkte erreichbar. Im ersten Tutorium im Januar können Fragen zu den schriftlichen Aufgaben geklärt werden.
Der Hintergrund zu Aufgabe 5 wird erst in der kommenden Woche Thema der Vorlesung.

Frohe Weihnachten
und alles Gute in 2006

Mündliche Aufgaben

8.1 **Happy**
Schreiben Sie zu den gegebenen Grammatiken Eingabedateien für den Parsergenerator HAPPY, so dass die erzeugten Parser bei korrekten Eingaben (Tokentyp `Char`) die Differenz (Anzahl `a`) - (Anzahl `b`) zurückliefern (Typ `Int`).

$G_{palindrom}: S \rightarrow aXa \mid bXb$
$X \rightarrow S \mid \varepsilon$

$G_{aUndB}: S \rightarrow a \mid aS \mid aSb$

Happy meldet für beide Grammatiken Konflikte (verwenden Sie die Option `-i`, um zusätzliche Informationen zu erhalten). Woran liegt das?

Welches Verhalten zeigen die erzeugten Parser? Geben Sie, falls möglich, jeweils eine Grammatik für die gleiche Sprache an, mit der HAPPY einen konfliktfreien Parser erzeugen kann.

8.2 **Konfliktauflösung mit Operatorpräzedenzen**
Gegeben sei die folgende Grammatik für Boolesche Ausdrücke mit konstantem Wert.

S → B
B → B ∧ B | (B) | ¬B | t | f

Erstellen Sie eine Analysetabelle, in der die Shift/Reduce-Konflikte über Präzedenzregeln der Operatoren aufgelöst werden. Erläutern Sie Ihre Lösung.

8.3 **Erste Ableitung mit Attributgrammatik**
Erweitern Sie die Grammatik G_{AE} der Vorlesung um Attribute, welche den eingegebenen Ausdruck einmal nach der Variablen a differenzieren. Die Ableitung soll als Zeichenkette aufgebaut und nicht vereinfacht werden.
Hinweis: Benutzen Sie zwei Attribute *exp* und *diff*, welche den Ausdruck selbst und seine Ableitung enthalten.

Bitte wenden!

Schriftliche Aufgaben

8.4 Erkennung einer Teilsprache von XML

12 Punkte

Die Grammatik in Abbildung 1 beschreibt eine Teilsprache von XML.[1] Interne DTDs, Processing Instructions u.ä. wurden hier zur Vereinfachung weggelassen, weiter müssen Textdaten stets mit `CDATA` begrenzt sein. Als Zeichensatz werden nur druckbare ASCII-Zeichen (alex: `$ printable`) verwendet.

Erstellen Sie mit HAPPY und ALEX einen Parser für diese Sprache, welcher die Ergebnisse in einer geeigneten Haskell-Datenstruktur speichert.

(a) **Grammatik-Transformation, Scanner und Parser:** Auf der Webseite zur Vorlesung finden Sie eine Alex-Datei `XMLSX.x` für einen XML-Scanner.

Die Terminale der Grammatik sind einfache Zeichen, weiter können Leerzeichen eine Bedeutung tragen. Aus diesem Grund wurden Teile der Erkennung in den Scanner verlagert.

Ändern Sie die Grammatik passend zu den Token des bereitgestellten Scanners. Ersetzen Sie ferner die speziellen Notationen der Grammatik, soweit sie im Parser verbleiben müssen. Sie können den Scanner auch nach Bedarf modifizieren. / 4

(b) Definieren Sie einen geeigneten **Datentyp** für die Repräsentation von XML-Dokumenten. / 2

(c) Erstellen Sie mit HAPPY einen passenden XML-Parser. / 4

Fügen Sie der Grammatik Attribute hinzu, mit denen die Wohlgeformtheit des Dokuments überprüft werden kann / 2

8.5 Attributgleichungssysteme

6 Punkte

Gegeben sei die kontextfreie attributierte Grammatik G mit der Attributmenge $Att = \{s, s_1, s_2, i_1, i_2\}$ und der folgenden Attributzuordnung:

$$syn(S) := \{s\},\ syn(A) := \{s_1, s_2\},\ inh(A) := \{i_1, i_2\}$$

Der Wertebereich in den Attributgleichungen sei stets $\mathbb{N}$ und *succ* die Nachfolgerfunktion.

$$
\begin{array}{lll}
G: S \to AA & (\pi_1) & \pi_1: \begin{array}{lcl} s.0 &=& succ(s_2.2) \\ i_1.1 &=& 1 \\ i_2.1 &=& succ(s_1.2) \\ i_1.2 &=& succ(s_2.1) \\ i_2.2 &=& succ(s_1.1) \end{array} \\
A \to a & (\pi_2) & \pi_2: \begin{array}{lcl} s_1.0 &=& succ(i_2.0) \\ s_2.0 &=& succ(i_1.0) \end{array} \\
A \to b & (\pi_3) & \pi_3: \begin{array}{lcl} s_1.0 &=& succ(i_1.0) \\ s_2.0 &=& succ(i_2.0) \end{array} \\
A \to c & (\pi_4) & \pi_4: \begin{array}{lcl} s_1.0 &=& succ(i_1.0) \\ s_2.0 &=& succ(i_1.0) \end{array}
\end{array}
$$

(a) Bestimmen Sie zu den Ableitungsbäumen der Wörter *aa*, *ab* und *bc* die jeweiligen Attributgleichungssysteme und berechnen Sie den Wert des Attributs s daraus.

(b) Ermitteln Sie unter Verwendung von Abhängigkeitsgraphen, ob in den Ableitungsbäumen aus a) stets alle Attributwerte berechenbar sind.

[1] vgl. *http://www.w3.org/TR/REC-xml*.

Auszug aus der XML-Spezifikation

Notation:

- Alternativen mit `|`, optionale Elemente mit `?`
- beliebige Wiederholung mit `*`, mindestens einmal mit `+`
- Zeichenmengen mit `[...]`, Komplementbildung mit `^`, Mengendifferenz mit '-'
- `#xN`: Zeichen mit ISO/IEC-Codierung N

```
Letter ::= [a-zA-Z]
Digit  ::= [0-9]

[1]  document ::= prolog element Misc*
[2]  Char     ::= ( vereinfacht: alex-macro $printable )
[3]  S        ::= (#x20 | #x9 | #xD | #xA)+
[4]  NChar     ::= Letter | Digit |'.' | '-' | '_' | ':'
[5]  Name     ::= (Letter | '_' | ':') (NChar )*
[10] AttValue ::= '"' ([^<&"] | Reference )* '"'
[11] SystemLit::= ('"' [^"]* '"')
[14] CharData ::= [^<&]* - ( [^<&]* ']]>' [^<&]* )
[15] Comment  ::= '<!--' ( (Char - '-') | ('-'(Char - '-')) )* '-->'
[18] CDSect   ::= CDStart CData CDEnd
[19] CDStart  ::= '<![CDATA['
[20] CData    ::= Char* - ( Char* ']]>' Char* )
[21] CDEnd    ::= ']]>'
[22] prolog   ::= XMLDecl? Misc* (dtd Misc*)?
[23] XMLDecl  ::= '<?xml' Version S? '?>'
[24] Version  ::= S 'version' Eq '"' VNum '"'
[25] Eq       ::= S? '=' S?
[26] VNum     ::= '1.0'
[27] Misc     ::= Comment | S
[28] dtd      ::= '<!DOCTYPE' S Name (S ExtID ) S? '>'
[39] element  ::= EmptyElem | STag content ETag
[40] STag     ::= '<' Name (S Attribute )* S? '>'
[41] Attribute::= Name Eq AttValue
[42] ETag     ::= '</' Name S? '>'
[43] content  ::= CharData* (( element | Reference | CDSect | Comment ) CharData* )*
[44] EmptyElem::= '<' Name (S Attribute )* S? '/>'
[66] CharRef  ::= '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'
[67] Reference::= EntityRef | CharRef
[68] EntityRef::= '&' Name ';'
[75] ExtID    ::= 'SYSTEM' S SystemLit

Wohlgeformtheit (kleiner Auszug!):
[39]    Der Name in STag und ETag muss gleich sein
[40,44] Attributnamen dürfen in einem Tag nicht mehrfach vorkommen
[66]    Die Codezahl muss im erlaubten Bereich liegen.
        (hier: $printable, normalerweise: Unicode)
```

Abbildung 1: Grammatik einer XML-Teilsprache