Karlsruher Institut für Technologie
Lehrstuhl für Programmierparadigmen
Sprachtechnologie und Compiler WS 2010/2011
Dozent: Prof. Dr.-Ing. G. Snelting
Übungsleiter: Matthias Braun `matthias.braun@kit.edu`

Übungsblatt 2 | Ausgabe: 24.4.2012 | Besprechung: 1.5.2012

Aufgabe 1: *Praxis: Flex*

Unter `http://pp.info.uni-karlsruhe.de/lehre/SS2012/compiler/uebung/minicalc.zip` finden Sie ein C-Programm zum Berechnen von einfachen mathematischen Ausdrücken.

1.1 *Scanner*

Der enthaltene Scanner wurde in flex geschrieben, ist allerdings unvollständig. Erweitern Sie diesen um folgende Tokentypen:

- **T_PLUS**: das '+' Zeichen
- **T_MINUS**: das '-' Zeichen
- **T_STAR**: das '*' Zeichen
- **T_SLASH**: das '/' Zeichen
- **T_LBRACE**: das '(' Zeichen
- **T_RBRACE**: das ')' Zeichen
- **T_NUMBER**: Eine Zahl. Diese beginnt mit einer Ziffernfolge. Danach kann ein Punkt und optional weitere Ziffern folgen. Alternativ darf die Zahl nur mit einem Punkt beginnen, wenn danach mindestens eine Ziffer folgt. Enthält die Zahl einen Punkt, so darf der Buchstabe **e** oder **E** angehängt werden, gefolgt von einem optionalen + oder **-** und weiteren Ziffern.

Zusätzlich soll bei **T_NUMBER** Tokens der Wert der Zahl im Token vermerkt werden.

Hinweise:

- In flex enthält die Variable yytext den erkannten String.
- Die Funktion scanf ermöglicht es einen String in eine Gleitpunktzahl zu konvertieren: sscanf(string , "%f", &variable);
- Der parser Code enthält eine Debughilfe.

Aufgabe 2: *Kontextfreie Grammatik 1*

2.1 *Einfache Grammatiken 1*

Betrachten Sie die folgende kontextfreie Grammatik und den String **aa+a***:

$$S \rightarrow S\ S\ + \mid S\ S\ * \mid \mathbf{a}$$

- Geben Sei eine Linksableitung für den String an.
- Geben Sie eine Rechtsableitung für den String an.
- Geben Sie einen Parse-Baum für den String an.
- Ist die Grammatik mehrdeutig oder nicht? Begründen Sie Ihre Antwort.
- Beschreiben Sie die von dieser Grammatik generierte Sprache.

2.2 *Einfache Grammatiken 2*

Wiederholen Sie die vorige Aufgabe mit

$$S \rightarrow S\ (\ S\ )\ S \mid \varepsilon$$

und dem String (()()).

2.3 *Entwerfen von Grammatiken*

Entwerfen Sie Grammatiken für:

- Die Menge aller Strings aus **0** und **1** mit derselben Anzahl von **0** und **1**
- Die Menge aller Strings aus **0** und **1** mit einer ungleichen Anzahl von **0** und **1**
- Die Menge aller Strings aus **0** und **1** in der **011** nicht als Teilstring vorkommt

Aufgabe 3: *Kontextfreie Grammatik 2*

Gegeben folgende Grammatik:

Satz	→	*Subjekt Prädikat* **.**
Subjekt	→	*Nomen Attribut*
Nomen	→	*Artikel Adjektive Substantiv* | *Pronomen*
Prädikat	→	*Verb Adverben Adjektiv* | *Verb Objekte Adverben Attribut*
Adjektive	→	ε | *Adjektive Adjektiv*
Objekte	→	*Objekt* | *Objekt Objekt*
Objekt	→	*Nomen Attribut*
Adverben	→	ε | *Adverben Adverb*
Attribut	→	ε | *Präposition Substantiv*
Artikel	→	**das**|**dem**|**den**|**der**|**des**|**die**|**dieser**|**ein**|**eine**|**einem**|**eines**|**kein**|*PossesivPronomen*
Verb	→	**bin**|**bringt**|**fliegen**|**gehe**|**hat**|**ist**|**traf**
Adverb	→	**damals**|**gerne**|**hier**|**hinterher**|**leider**|**links**|**morgen**|**nämlich**|**vielleicht**
Substantiv	→	**Blau**|**Bruder**|**CD**|**Fliegen**|**Gelb**|**Gewehr**|**Jäger**|**Junge**|**Leben**|**Satz**|**Schloss**|**Verb**
Adjektiv	→	ε | **fliegende**|**neue**|**schnell**|**schön**|**teure**
Pronomen	→	**du**|**er**|**es**|**ich**|**sie**|*PossesivPronomen*
PossesivPronomen	→	**dein**|**mein**|**sein**|**seinem**
Präposition	→	**an**|**in**|**mit**

3.1 *Parse-Bäume*

Geben Sie je einen Parse-Baum für folgende Sätze an:

1. **das Leben ist nämlich schön.**
2. **Blau ist das neue Gelb.**
3. **schnell fliegende Fliegen fliegen Fliegen hinterher.**
4. **der Junge bringt seinem Bruder die teure CD.**
5. **dieser Satz kein Verb.**

3.2 *Mehrdeutigkeit*

Ist die Grammatik eindeutig? Geben Sie eine Begründung (falls eindeutig) oder ein Gegenbeispiel (falls nicht eindeutig) an!

Aufgabe 4: *Grammar Engineering*

4.1 *Elimination der Linksrekursion*

Es folgt eine Grammatik für reguläre Ausdrücke über die Symbole **a** und **b**:

$$
\begin{aligned}
\mathit{rexpr} &\rightarrow \mathit{rexpr} + \mathit{rterm} \mid \mathit{rterm} \\
\mathit{rterm} &\rightarrow \mathit{rterm}\ \mathit{rfactor} \mid \mathit{rfactor} \\
\mathit{rfactor} &\rightarrow \mathit{rfactor}\ \mathbf{*} \mid \mathit{rprimary} \\
\mathit{rprimary} &\rightarrow \mathbf{a} \mid \mathbf{b}
\end{aligned}
$$

- Führen Sie eine Linksfaktorisierung dieser Grammatik durch.
- Macht die Linksfaktorisierung sie für Top-Down-Parsing geeignet?
- Eliminieren Sie anschließend die Linksrekursion aus der ursprünglichen Grammatik.
- Eignet sich die resultierende Grammatik für Top-Down-Parsing.

4.2 *Dangling-Else*

Die folgende Grammatik soll das aus der Vorlesung bekannte Problem des „Dangling-Else" beseitigen:

$$
\begin{aligned}
\mathit{stmt} &\rightarrow \mathbf{if}\ \mathit{expr}\ \mathbf{then}\ \mathit{stmt} \\
&\mid \mathit{matchedStmt} \\
\mathit{matchedStmt} &\rightarrow \mathbf{if}\ \mathit{expr}\ \mathbf{then}\ \mathit{matchedStmt}\ \mathbf{else}\ \mathit{stmt} \\
&\mid \mathbf{other}
\end{aligned}
$$

Zeigen Sie, dass sie immer noch mehrdeutig ist.