Department of Information and Computing Sciences
Utrecht University

INFOB3TC – Exam 3

Sean Leather

Monday, 12 March 2012, 14:00 – 17:00

1 Preliminaries

- The exam consists of 4 pages (including this page). Please verify that you received all pages.
- Write your **name** and **student number** on all submitted work. Also include the total number of separate sheets of paper.
- The maximum score is stated at the top of each question. The total amount of points you can get is 100.
- Give simple and concise answers. Write readable text. Do not use pencils or pens with red ink.
- Write your text in English.
- When writing grammar and language constructs, you may use any set, sequence, or language operation covered during the course.
- When writing Haskell code, you may use functions from the *Prelude* and the following modules: *Data.Char*, *Data.List*, *Data.Maybe*, and *Control.Applicative*. If you are in doubt whether a certain function is allowed, please ask.
- Use your time efficiently. Look over all the questions, and answer the ones you know first.

Good luck!

2 Questions

2.1 Grammar Transformation

1 (20 points)**.** Consider the following grammar over the alphabet $\{\texttt{m}, \texttt{n}, \texttt{8}, \texttt{\#}, \texttt{@}\}$:

$$
\begin{aligned}
S &\longrightarrow \texttt{m}\,\texttt{8}\,P \mid \texttt{m}\,\texttt{8}\,P\,S \\
P &\longrightarrow P\,Q\,\texttt{\#} \mid \texttt{n} \\
Q &\longrightarrow \texttt{@}\,P
\end{aligned}
$$

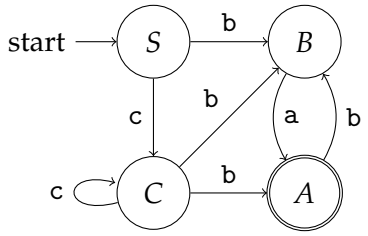
Transform this into a minimal grammar from which we can immediately derive the simplest and most efficient parser. You may use any of the following transformations:

Inline nonterminal	Remove duplicate productions
Introduce nonterminal	Remove left-recursion
Introduce $\cdot^*$	Remove unreachable production
Introduce $\cdot^+$	Left-factoring
Introduce $\cdot?$	

•

2.2 Regular Languages

2 (20 points)**.** Consider the following nondeterministic finite state automaton (NFA):



(a) Construct a deterministic finite state automaton (DFA) from the NFA.

(b) Give a regular grammar for the DFA.

•

2.3 LL Parsing

3 (20 points)**.** Consider the grammar in the table below:

NT	Production	*empty*	*emptyRhs*	*first*	*firstRhs*	*follow*	*lookAhead*
S							
	$S \rightarrow \texttt{1}\, B\, \texttt{9}$						
B							
	$B \rightarrow \texttt{2}\, C$						
C							
	$C \rightarrow B\, \texttt{3}$						
	$C \rightarrow \texttt{5}$						

(a) Complete the table by computing the values in the columns for the appropriate rows. Use *True* and *False* for property values and set notation for everything else.

(b) Is the grammar LL(1)? Explain how you arrived at your answer. If it is is not, transform the grammar so that it is LL(1). Give the rows of the table that differ. You don't need to write the whole table again.

•

2.4 Parser Combinators

4 (20 points)**.** The following grammar describes JSON, the JavaScript Object Notation, a data-interchange format.

Object → { } | { *Members* }
Members → *Pair* | *Pair* , *Members*
Pair → *String* : *Value*
Value → *String* | *Number* | *Object* | *Array* | `true` | `false` | `null`
Array → [] | [*Elements*]
Elements → *Value* | *Value* , *Elements*

For *String* and *Number*, you may assume the standard Haskell *String* and *Double* formats and values.

(a) Give an abstract syntax for the grammar using a family of Haskell datatypes.

(b) Define parsers for the datatypes using the parser combinators covered in the course.

(c) Define the algebra type and fold functions for the datatypes.

•

2.5 Context-Free Grammars

5 (20 points). Consider the following grammar G over the alphabet $\{\texttt{0}, \texttt{a}, \texttt{b}, [,]\}$:

$$S \to \texttt{0} \mid \texttt{00} \mid [\,T\,]$$
$$R \to \texttt{a} \mid \texttt{b}$$
$$T \to \varepsilon \mid R\,R\,T$$

(a) Is the grammar context-free? Regular? Why?

(b) Give the language $L(G)$ of the grammar G without referring to G itself.

(c) Is the language $L(G)$ context-free? Regular? Why?

•