Department of Information and Computing Sciences
Utrecht University

INFOB3TC – Exam

Andres Löh

Wednesday, 3 February 2010, 09:00–12:00

Preliminaries

- The exam consists of 4 pages (including this page). Please verify that you got all the pages.
- Write your **name** and **student number** on all submitted work. Also include the total number of separate sheets of paper.
- For each task, the maximum score is stated. The total amount of points you can get is 100.
- Try to give simple and concise answers. Write readable. Do not use pencils or pens with red ink.
- You may answer questions in Dutch or English.
- When writing Haskell code, you may use Prelude functions and functions from the *Data.List*, *Data.Maybe*, *Data.Map*, *Control.Monad* modules. Also, you may use all the parser combinators from the `uu-tc` package. If in doubt whether a certain function is allowed, please ask.

Good luck!

1

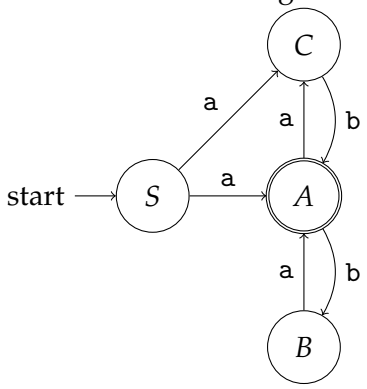
DFAs

1 (10 points)**.** Consider the following Haskell lexer:

$$
\begin{aligned}
&lex, lex_0, lex_1, lex_2, lex_3 :: String \to Bool \\
&lex = lex_0 \\
&lex_0\ (\texttt{'1'} : xs) = lex_1\ xs \\
&lex_0\ (\texttt{'0'} : xs) = lex_2\ xs \\
&lex_0\ \_ \qquad\quad = False \\
&lex_1\ (\texttt{'0'} : xs) = lex_1\ xs \\
&lex_1\ (\texttt{'1'} : xs) = lex_1\ xs \\
&lex_1\ (\texttt{'.'} : xs) = lex_3\ xs \\
&lex_1\ \_ \qquad\quad = False \\
&lex_2\ (\texttt{'.'} : xs) = lex_3\ xs \\
&lex_2\ \_ \qquad\quad = False \\
&lex_3\ (\texttt{'0'} : xs) = lex_3\ xs \\
&lex_3\ (\texttt{'1'} : xs) = lex_3\ xs \\
&lex_3\ [] \qquad\quad = True \\
&lex_3\ \_ \qquad\quad = False
\end{aligned}
$$

Give a deterministic finite state automaton that accepts the same language as the Haskell code. •

Consider the following nondeterministic finite state automaton:



2 (4 points)**.** Give two different words of length at least 5 that are accepted by the automaton. •

3 (10 points)**.** Transform the automaton given above into an equivalent deterministic finite-state automaton. It is not necessary to include any unreachable states, but the correspondence with the original automaton should be made obvious, for example by using the standard construction and naming nodes adequately. •

Regular expressions

Consider the following Haskell datatype that describes regular expressions over an alphabet type s:

$$
\begin{aligned}
\textbf{data}\ \mathit{Regex}\ s = {} & \mathit{Empty} \\
\mid {} & \mathit{Epsilon} \\
\mid {} & \mathit{Const}\ s \\
\mid {} & \mathit{Sequ}\ \ (\mathit{Regex}\ s)\ (\mathit{Regex}\ s) \\
\mid {} & \mathit{Plus}\ \ (\mathit{Regex}\ s)\ (\mathit{Regex}\ s) \\
\mid {} & \mathit{Star}\ \ (\mathit{Regex}\ s)
\end{aligned}
$$

4 (4 points)**.** Translate the regular expression

$$(\mathtt{aa} + \mathtt{b})^*$$

into a value of type *Regex Char*. •

5 (10 points)**.** Define a function

$$\mathit{regexParser} :: \mathit{Eq}\ s \Rightarrow \mathit{Regex}\ s \rightarrow \mathit{Parser}\ s\ [s]$$

using the parser combinators such that *regexParser r* is a parser for the language described by the regular expression *r*. The parser should return the list of symbols recognized. •

6 (10 points)**.** Define an algebra type *RegexAlgebra* and a fold function *foldRegex* for the *Regex* type. (Note that *regexParser could* be written as a call to *foldRegex*. It is, however, *not* part of the task to do so.) •

Regular vs. context-free

7 (10 points)**.** Consider the language described by the following grammar with start symbol S:

$$
\begin{aligned}
S &\rightarrow \mathtt{a}S\mathtt{a} \mid B \\
B &\rightarrow \mathtt{b} \mid \mathtt{b}B
\end{aligned}
$$

(a) Give a characterization of the words that belong to the language.

(b) Is the language regular? If yes, give a regular expression that describes the language. If not, use the pumping lemma to prove that the language cannot be regular. •

8 (10 points)**.** Consider the language over alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ that contains all words with at least three '`b`'s.

(a) Give a context-free grammar for the language.

(b) Is the language regular? If yes, give a regular expression that describes the language. If not, use the pumping lemma to prove that the language cannot be regular. •

9 (6 points)**.** Let L be a language that is context-free, but not regular. Is it possible that the complement of L is regular? Explain your answer. •

LL and LR parsing

Consider the following grammar, with start symbol S:

$$S \to OSS \mid V$$
$$O \to \texttt{+} \mid \varepsilon$$
$$V \to \texttt{1} \mid \texttt{x}$$

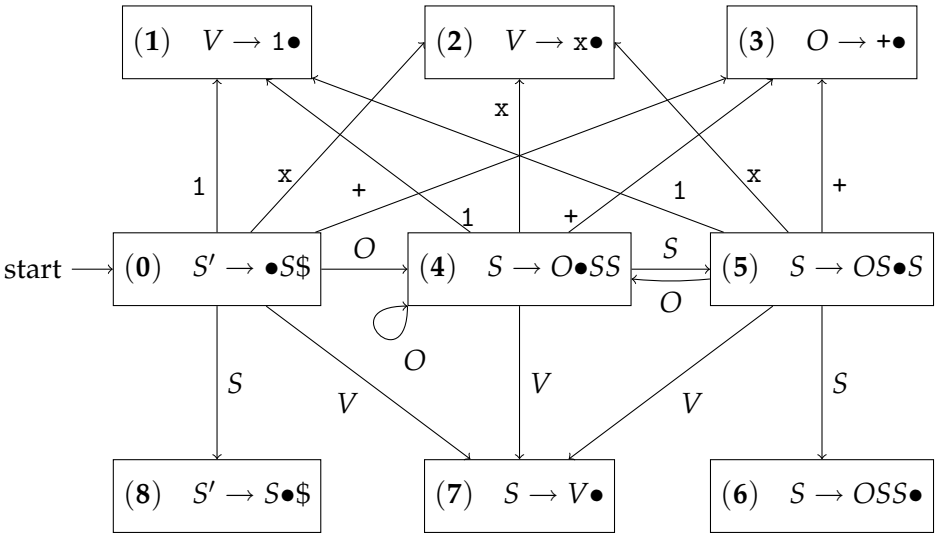
10 (8 points)**.** Compute the *empty* property, the *first* and *follow* sets for all nonterminals. Is the grammar LL(1)? •

We augment the grammar above in preparation for LR parsing:

$$S' \to S\$$$

and S' becomes the new start symbol.

The LR(0) automaton corresponding to the full grammar looks as follows (each state is only labelled by its kernel items, and numbered before the production for future reference):



11 (6 points)**.** Classify each state as a shift state, reduce state, or shift-reduce conflict state. Note that you may have to compute the closure of the item sets to determine that. Also mark potential reduce-reduce conflicts. Would applying SLR(1) parsing help to resolve the potential conflicts? •

12 (6 points)**.** Play through the LR parsing process (no extras, neither SLR nor LALR) for the word `+11`. Resolve potential shift-reduce conflicts by always choosing to shift, and potential reduce-reduce conflicts by picking any of the available reduction. •

13 (6 points)**.** Are there words in the language that cannot be parsed successfully using the simplistic strategy from the previous task? If so, give an example. •

14 (meta question)**.** How many out of the 100 possible points do you think you will get for this exam? •