Department of Information and Computing Sciences
Utrecht University

INFOB3TC – Solutions for the Exam

Andres Löh

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

Please keep in mind that often, there are many possible solutions, and that these example solutions may contain mistakes.

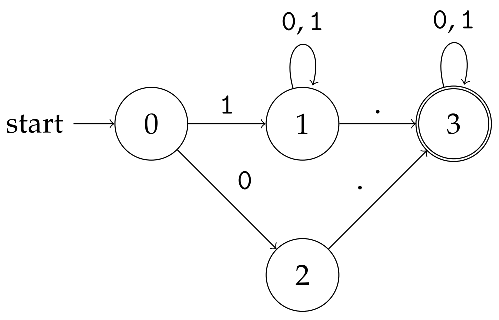
DFAs

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

```
lex, lex₀, lex₁, lex₂, lex₃ :: String → Bool
lex = lex₀
lex₀ ('1' : xs) = lex₁ xs
lex₀ ('0' : xs) = lex₂ xs
lex₀ _          = False
lex₁ ('0' : xs) = lex₁ xs
lex₁ ('1' : xs) = lex₁ xs
lex₁ ('.' : xs) = lex₃ xs
lex₁ _          = False
lex₂ ('.' : xs) = lex₃ xs
lex₂ _          = False
lex₃ ('0' : xs) = lex₃ xs
lex₃ ('1' : xs) = lex₃ xs
lex₃ []         = True
lex₃ _          = False
```

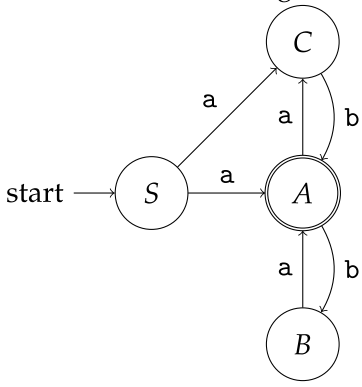
Give a deterministic finite state automaton that accepts the same language as the Haskell code. •

Solution 1.



○

Consider the following nondeterministic finite state automaton:

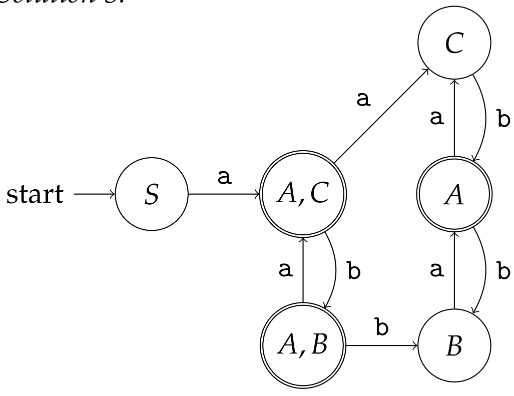


2 (4 points)**.** Give two different words of length at least 5 that are accepted by the automaton. ●

Solution 2. For example `ababa` and `abaab`. ○

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. ●

Solution 3.



○

Regular expressions

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

```
data Regex s = Empty
             | Epsilon
             | Const s
             | Sequ  (Regex s) (Regex s)
             | Plus  (Regex s) (Regex s)
             | Star  (Regex s)
```

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

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

into a value of type *Regex Char*. •

Solution 4.

```
Star (Plus (Sequ (Const 'a') (Const 'a')) (Const 'b'))
```

○

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

```
regexParser :: Eq s ⇒ Regex s → 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. •

Solution 5.

```
regexParser Empty        = empty
regexParser Epsilon      = succeed []
regexParser (Const s)    = (:[]) <$> symbol s
regexParser (Sequ r1 r2) = (++) <$> regexParser r1 <*> regexParser r2
regexParser (Plus r1 r2) = regexParser r1 <|> regexParser r2
regexParser (Star r)     = concat <$> many (regexParser r)
```

Handling the end of input was not required, but ok if done. ○

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.) •

Solution 6. The algebra needs to parameters, because *Regex* already has one:

```
type RegexAlgebra s r = (r, r, s → r, r → r → r, r → r → r, r → r)
```

The *fold* function is straightforward to define:

```
foldRegex :: RegexAlgebra s r → Regex s → r
foldRegex (empty, epsilon, const, sequ, plus, star) r = fold r
  where
```

```
fold Empty         = empty
fold Epsilon       = epsilon
fold (Const s)     = const s
fold (Sequ r₁ r₂)  = sequ (fold r₁) (fold r₂)
fold (Plus r₁ r₂)  = plus (fold r₁) (fold r₂)
fold (Star r)      = star (fold r)
```

For completeness, *regexParser* as a fold:

```
regexParserAlgebra :: Eq s ⇒ RegexAlgebra s (Parser s [s])
regexParserAlgebra = (empty,
                      succeed [],
                      λs → (:[]) <$> symbol s,
                      λx₁ x₂ → (++) <$> x₁ <*> x₂,
                      (<|>),
                      λx → concat <$> many x)
regexParser :: Eq s ⇒ Regex s → Parser s [s]
regexParser = foldRegex regexParserAlgebra
```

○

Regular vs. context-free

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

$$S \rightarrow \texttt{a}S\texttt{a} \mid B$$
$$B \rightarrow \texttt{b} \mid \texttt{b}B$$

(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. ●

Solution 7. The language described by the grammar is $\{\texttt{a}^n\texttt{b}^m\texttt{a}^n \mid m, n \in Nat, m > 0\}$. The language is not regular. We use the pumping lemma. Assuming the language is regular, there must be a number n of states of a corresponding finite state automaton. We now choose the word a^nba^n which can be derived from S and therefore is in the language. The pumping lemma states that for any subword of length at least n, we can pump. So we choose the subword a^n, and know that there must be an $m > 0$ such that $a^{m+n}ba^n$ is also in the language. But that is not the case. Hence, the assumption that the language is regular must be wrong. ○

8 (10 points)**.** Consider the language over alphabet $A = \{\texttt{a}, \texttt{b}\}$ 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. ●

Solution 8. A context-free grammar for the language (with start symbol S) is as follows:

$$\begin{aligned} S &\longrightarrow \mathtt{b}A \mid \mathtt{a}S \\ A &\longrightarrow \mathtt{b}B \mid \mathtt{a}A \\ B &\longrightarrow \mathtt{b}C \mid \mathtt{a}B \\ C &\longrightarrow \mathtt{b}C \mid \mathtt{a}C \mid \varepsilon \end{aligned}$$

This grammar is in fact a regular grammar, so the language is clearly regular. One of many possible regular expressions for the language is

$$((\mathtt{a}+\mathtt{b}))^{*}\mathtt{ba}^{*}\mathtt{ba}^{*}\mathtt{b}((\mathtt{a}+\mathtt{b}))^{*}$$

○

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. ●

Solution 9. No, that is not possible. The complement of a regular language is always regular, so if the complement of L is regular, then the complement of the complement of L must be regular. But the complement of the complement of L is L itself. ○

LL and LR parsing

Consider the following grammar, with start symbol S:

$$\begin{aligned} S &\longrightarrow OSS \mid V \\ O &\longrightarrow \mathtt{+} \mid \varepsilon \\ V &\longrightarrow \mathtt{1} \mid \mathtt{x} \end{aligned}$$

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

Solution 10. Only O can derive the empty string.

$$\begin{aligned} \mathit{first}\ S &= \{\mathtt{+},\mathtt{1},\mathtt{x}\} \\ \mathit{first}\ O &= \{\mathtt{+}\} \\ \mathit{first}\ V &= \{\mathtt{1},\mathtt{x}\} \end{aligned}$$

$$\begin{aligned} \mathit{follow}\ S &= \{\mathtt{+},\mathtt{1},\mathtt{x}\} \\ \mathit{follow}\ O &= \{\mathtt{+},\mathtt{1},\mathtt{x}\} \\ \mathit{follow}\ V &= \{\mathtt{+},\mathtt{1},\mathtt{x}\} \end{aligned}$$

With this, we can see that the grammar is not LL(1), because

$$\begin{aligned} \mathit{lookahead}\ (S \to OSS) &= \{\mathtt{+},\mathtt{1},\mathtt{x}\} \\ \mathit{lookahead}\ (S \to V) &= \{\mathtt{1},\mathtt{x}\} \end{aligned}$$

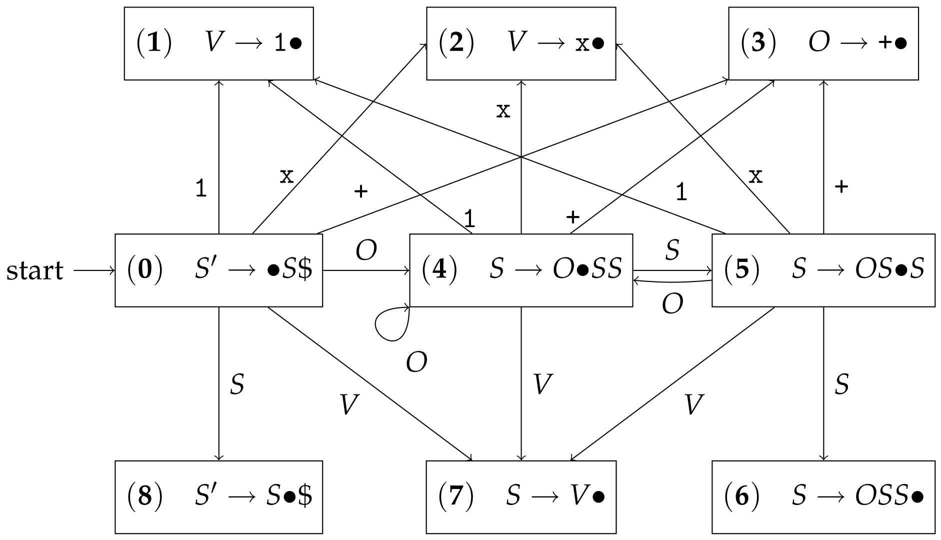
and these lookahead sets are not disjoint. ○

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? ●

Solution 11. The states (**1**), (**2**), (**3**), (**6**), (**7**) are all reduce states, and do not gain extra items by computing the closure. The state (**8**) is the special end state.

The states (**0**), (**4**), (**5**) gain the following items for closure

$$\begin{aligned} S &\to \bullet OSS \\ S &\to \bullet V \\ O &\to \bullet + \\ O &\to \bullet \\ V &\to \bullet 1 \\ V &\to \bullet x \end{aligned}$$

Therefore, all three states have a shift-reduce conflict. SLR(1) parsing would not be of any help here, because we can shift for all three symbols in the alphabet, and the follow set of O also contains all three symbols. ○

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. ●

Solution 12.

stack	input	remark
$(\mathbf{0})$	`+11$`	we always shift, thus move to $(\mathbf{3})$
$(\mathbf{0})\mathtt{+}(\mathbf{3})$	`11$`	reduce by $O \rightarrow \mathtt{+}$
$(\mathbf{0})O(\mathbf{4})$	`11$`	we always shift, thus move to $(\mathbf{1})$
$(\mathbf{0})O(\mathbf{4})\mathtt{1}(\mathbf{1})$	`1$`	reduce by $V \rightarrow \mathtt{1}$
$(\mathbf{0})O(\mathbf{4})V(\mathbf{7})$	`1$`	reduce by $S \rightarrow V$
$(\mathbf{0})O(\mathbf{4})S(\mathbf{5})$	`1$`	we always shift, thus move to $(\mathbf{1})$
$(\mathbf{0})O(\mathbf{4})S(\mathbf{5})\mathtt{1}(\mathbf{1})$	$	reduce by $V \rightarrow \mathtt{1}$
$(\mathbf{0})O(\mathbf{4})S(\mathbf{5})V(\mathbf{7})$	$	reduce by $S \rightarrow V$
$(\mathbf{0})O(\mathbf{4})S(\mathbf{5})S(\mathbf{6})$	$	reduce by $S \rightarrow OSS$
$(\mathbf{0})S(\mathbf{8})$	$	success

○

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. ●

Solution 13. Yes, one example is the word `11` which can be derived as follows

$$S \Rightarrow OSS \Rightarrow^* \varepsilon VV \Rightarrow^* \mathtt{11}$$

Trying to parse this word yields

stack	input	remark
$(\mathbf{0})$	`11`	we always shift, thus move to $(\mathbf{1})$
$(\mathbf{0})\mathtt{1}(\mathbf{1})$	`1`	reduce by $V \rightarrow \mathtt{1}$
$(\mathbf{0})V(\mathbf{7})$	`1`	reduce by $S \rightarrow V$
$(\mathbf{0})S(\mathbf{8})$	`1`	failure

○

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

Solution 14. I ask this question because it is the first time I'm setting an exam for the course, and I'm genuinely interested how difficult the students perceive the exam to be, and how good the students are in judging their own performance. Obviously, the answer to this question has no relevance for the final result. ○