

# INFOB3TC – Exam 2

Johan Jeuring

Friday, 30 January 2015, 11:00–13:00

## Preliminaries

- The exam consists of 9 pages (including this page). Please verify that you got all the pages.
- Fill out the answers **on the exam itself**.
- Write your **name** and **student number** here:

- The maximum score is stated at the top of each question. The total amount of points you can get is 90.
- Try to give simple and concise answers. Write readable text. Do not use pencils or pens with red ink. You may use Dutch or English.
- When writing grammar and language constructs, you may use any set, sequence, or language operations covered in the lecture notes.
- When writing Haskell code, you may use Prelude functions and functions from the following modules: *Data.Char*, *Data.List*, *Data.Maybe*, and *Control.Monad*. Also, you may use all the parser combinators from the *uu-tc* package. If you are in doubt whether a certain function is allowed, please ask.

*Good luck!*

## Questions

### Regular expressions and languages

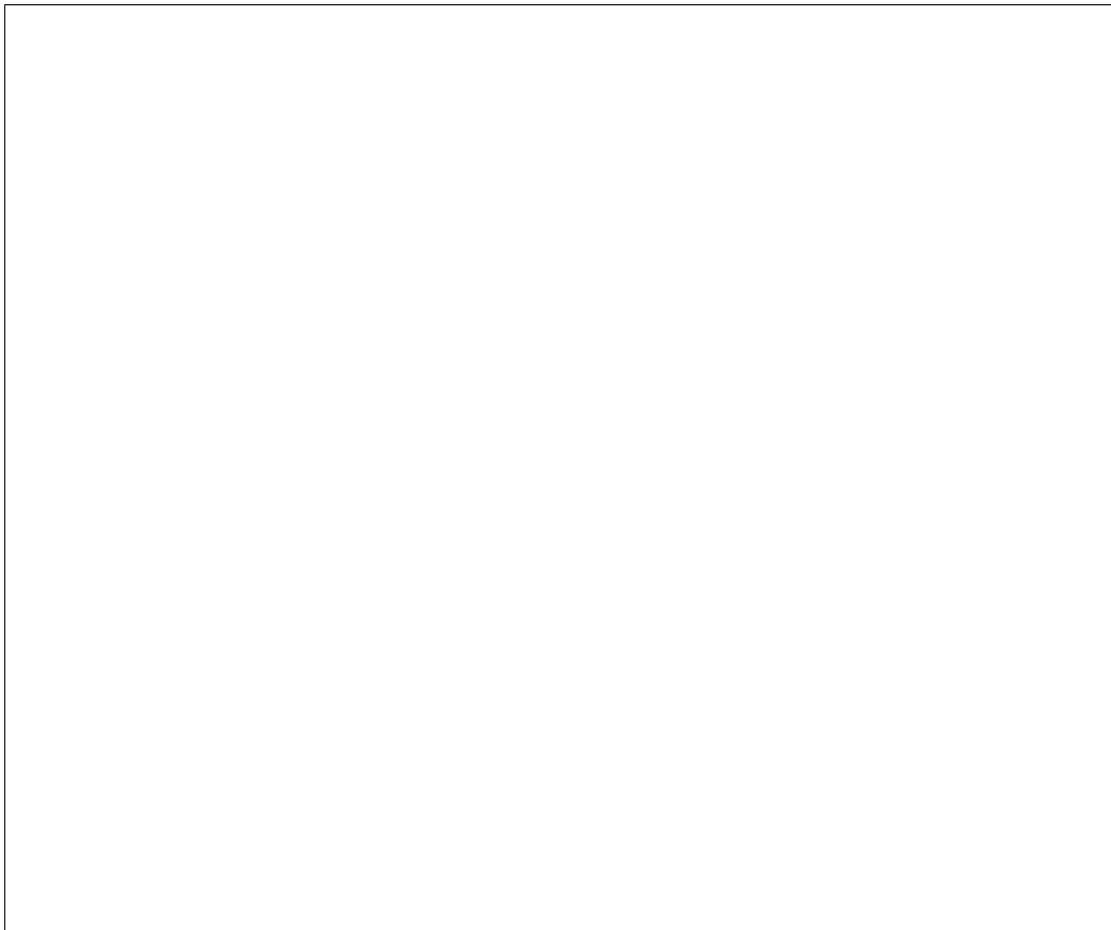
1 (5+5 points). Consider the grammars for the regular languages  $L_1$  and  $L_2$ :

$$L_1: S \rightarrow bA \mid aS$$

$$A \rightarrow aS \mid \varepsilon$$

$$L_2: S \rightarrow bS \mid Sa \mid \varepsilon$$

Give a regular expression for each language. •



2 (10+10 points). For each language definition below, show whether or not the language is regular. If it is regular, give one of the following:

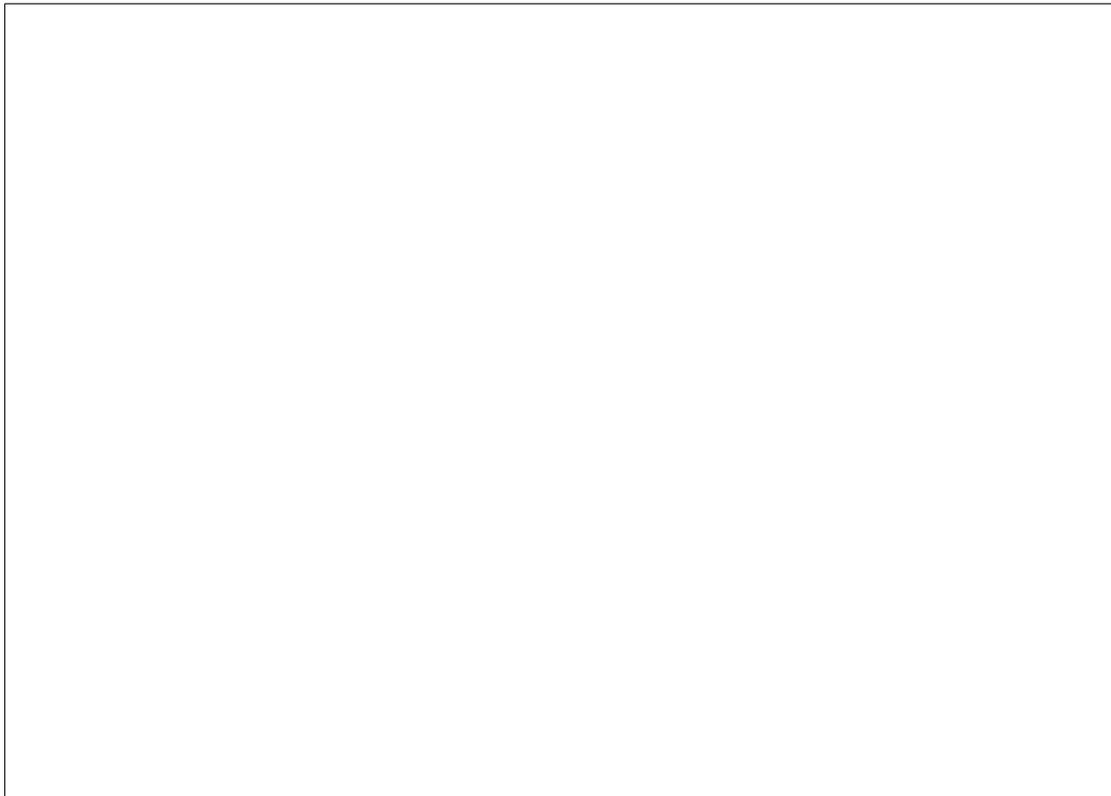
- (a) a regular grammar in an acceptable form,
- (b) a regular expression, or
- (c) a finite state automaton.

If the language is not regular, prove that using the pumping lemma for regular languages.

(a)  $\{ o^m p^n \mid n = m + 1 \}$

(b)  $\{ 3^j 7^k \mid j > 2, k < 5 \}$

•



## LL parsing

In these exercises we will look at the grammar

$$\begin{aligned}
 M &\rightarrow \langle E \rangle M \mid \varepsilon \\
 E &\rightarrow Q \mid Q;E \\
 Q &\rightarrow 0 \mid 1 \mid M
 \end{aligned}$$

3 (15 points). Complete the table below by computing the values in the columns for the appropriate rows. Use *True* and *False* for property values and set notation for everything else.

NT	Production	<i>empty</i>	<i>emptyRhs</i>	<i>first</i>	<i>firstRhs</i>	<i>follow</i>	<i>lookAhead</i>
<i>M</i>	$M \rightarrow \langle E \rangle M$						
	$M \rightarrow \varepsilon$						
<i>E</i>	$E \rightarrow Q$						
	$E \rightarrow Q;E$						
<i>Q</i>	$Q \rightarrow 0$						
	$Q \rightarrow 1$						
	$Q \rightarrow M$						

4 (10 points). Is the above grammar LL(1)? Explain how you arrived at your answer. If the grammar is not LL(1), transform the grammar such that is LL(1) and complete a new table with only the rows that differ from the old table. ●



5 (5 points). Show the steps that a parser for the above LL(1) grammar (after transformation if necessary) goes through to recognize the following input sequence:

<0;<1>>

For each step (one per line), show the stack, the remaining input, and the action (followed by the relevant symbol or production) performed. If you reach a step in which you cannot proceed, note the action as "error." •



## LR parsing

Consider the following grammar, with start symbol  $S$ :

$$S \rightarrow L = R \mid R$$

$$L \rightarrow * R \mid i$$

$$R \rightarrow L$$

We augment the grammar above in preparation for LR parsing:

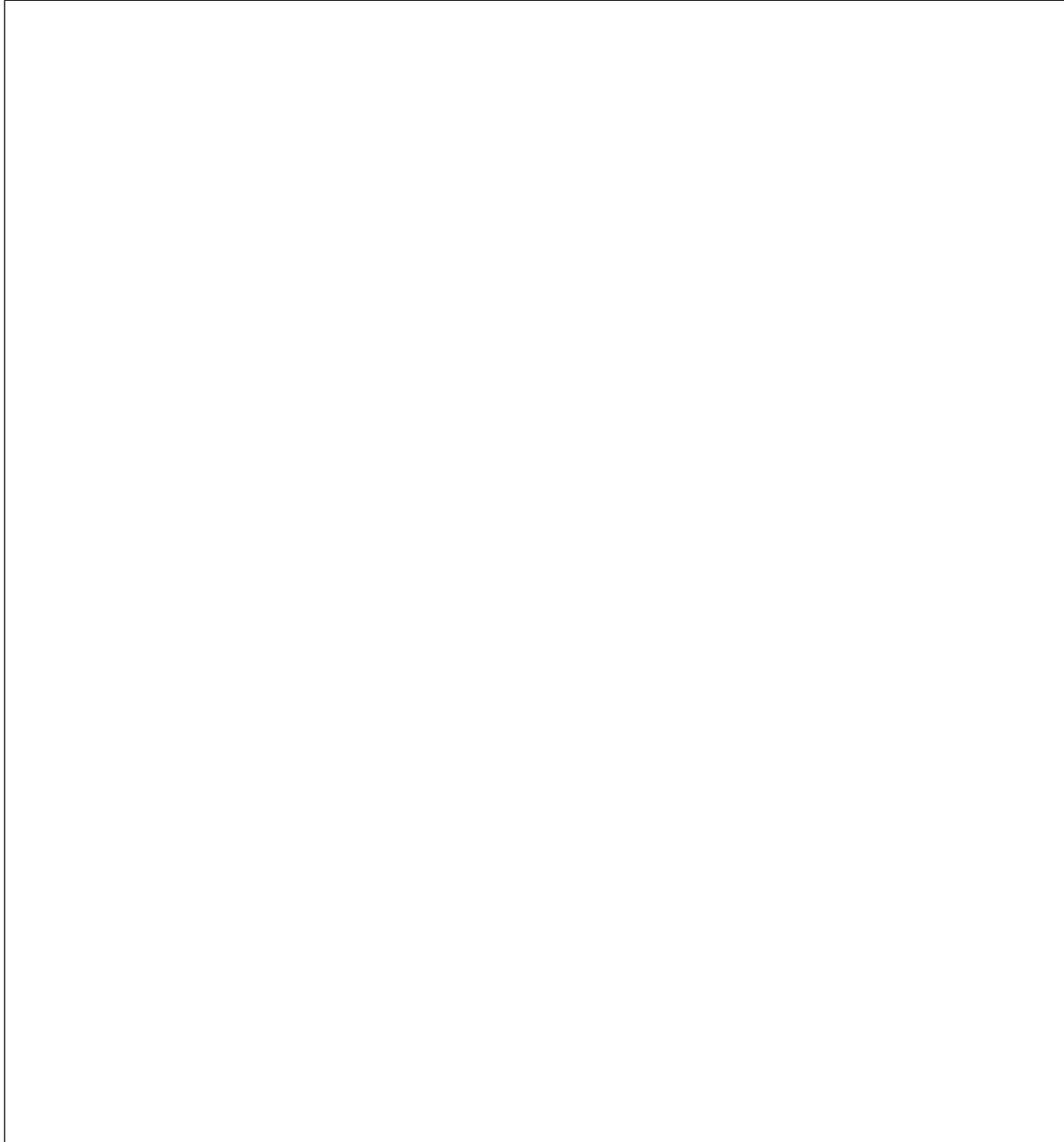
$$S' \rightarrow S\$$$

and  $S'$  becomes the new start symbol.

**6** (10 points). Compute the LR(0) automaton corresponding to the full grammar. Number each state for future reference. ●



7 (10 points). Classify each state in your LR(0) automaton as a shift state, reduce state, or shift-reduce conflict state. Also mark potential reduce-reduce conflicts. If there are conflicts, would applying SLR(1) parsing help to resolve these? •



8 (10 points). Play through the LR parsing process for the word `**i=*i$`. If there is a choice somewhere, make this explicit. Show in each step at which state in your LR(0) automaton you are. ●

