Syntax and Semantics

Written Exam, June 01, 2015, 13:00 - 16:00

Navn:

Stud.nr.:

Read this before you start!

- This exam contains 6 exercises. Each exercise is compulsory and has an equal contribution to the final grade.
- Each exercise should be solved in the space available on the page of the exercise and on the back page. Additional pages might be added if needed - for this contact the exam supervisors. Use page numbers to indicate the order in which the examiner should read your solutions.
- For preparing the final solutions you might use additional paper drafts. However, do hand in only the pages with the final solutions of the exercises.
- The solutions of the exercises must be written in English.
- During the exam no written materials are allowed. You are not allowed to use any electronic device.

Exercises

Exercise 1. Consider the following languages over the alphabet $\Sigma = \{a, b\}$.

$$L_1 = \left\{ w \;\middle|\; \begin{array}{l} w \text{ has an even number of } a\text{'s and an odd number of } b\text{'s,} \\ \text{and } w \text{ does not contain the substring } ab \end{array} \right\},$$

$$L_2 = \{w \mid w \text{ contains exactly one } b\}.$$

1. Prove that L_1 is regular by constructing an automaton that will recognize this language and guarantees that it is regular.
2. Prove that $L_1 \cap L_2$ is regular by constructing an automaton that will recognize this language and guarantees that it is regular.
3. Identify a regular expression that characterizes $L_1 \cap L_2$ by using the algorithm of conversion of an automaton into a regular expression presented at the course. (*If the automaton constructed at point 2. does not correspond to the type of automaton required by the algorithm of conversion, firstly convert your automaton into the required type.*)

Ad hoc solutions will not be considered.

Exercise 2. Consider the following context-free grammar with start variable S over the final alphabet $\Sigma = \{[,], a, :, *\}$:

$$S \to T \mid S : T$$
$$T \to T * F * T \mid \varepsilon$$
$$F \to a \mid [S]$$

1. Which of the following strings are described by this grammar and which are not? For those that are described, provide a syntax tree.

   (a) $** a * *a* :$

   (b) $[*a*]$

   (c) $: *a* :$

2. Convert the grammar into an equivalent one in Chomsky normal form by using the algorithm presented at the course. Ad hoc solutions will not be considered.

3. Use the construction presented in the course to construct a push-down automaton that recognizes the language generated by this grammar. Ad hoc solutions will not be considered.

Exercise 3. Prove that the language

$$L = \{a^n \mid n \text{ is a prime natural number}\}$$

over the alphabet $\{a\}$ is not a context-free language, where a^n, as usual, denotes the word formed by n consecutive occurrences of a.

Exercise 4. Consider the language **Bims** in which the Boolean expressions are given by the following grammar:

$$b ::= a_1 \neq a_2 \mid a_1[div]a_2 \mid b_1 \leftrightarrow b_2 \mid b_1[eor]b_2 \mid \bot,$$

where a denotes arithmetic expressions, $\neq$ denotes the relation *"not equal"*, $a_1[div]a_2$ expresses the fact that "a_1 *is a divisor of* a_2", $\leftrightarrow$ denotes the classical logical equivalence, $b_1[eor]b_2$ expresses the fact that "*exactly one of* b_1 *and* b_2 *is true*" and $\bot$ denotes *"false"*.

1. Give a big-step semantics for these Boolean expressions, assuming that you already have the big-step semantics for arithmetic expressions.

2. Describe the corresponding transition system using the formal definition provided at the course.

Exercise 5. Consider the language **Bump** with call-by-name parameter passing. Hüttel's book presents transition rule for procedure calls assuming the case of mixed scope rules with dynamic scope rules for variables and static scope rules for procedures.
Provide the similar transition rule for procedure calls assuming static scope rules for variables and dynamic scope rules for procedures.

Exercise 6. Consider the following statement in **Bip**.

```
begin
    var x:=2;
    var y:=6;
    proc p is x:=x+1;
    proc q is call p;
    begin
        var x:=8;
        proc p is x:=x+1;
        call q;
        y:=x
    end
end.
```

1. What is the value of y after the statement is executed assuming fully dynamic scope rules for both variables and procedures? Motivate your answer.

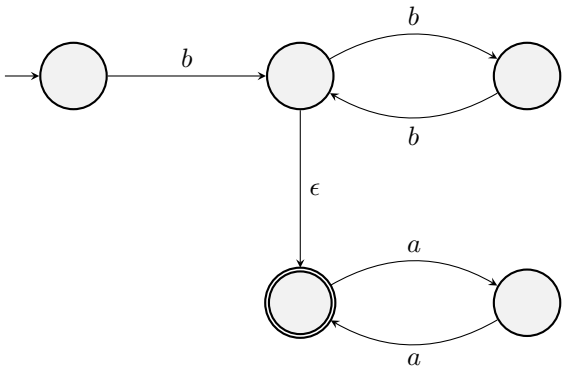
2. What is the value of y after the statement is executed assuming dynamic scope rules for procedures and static scope rules for variables? Motivate your answer.

3. What is the value of y after the statement is executed assuming fully static scope rules for both procedures and variables? Motivate your answer.

Exercise 1

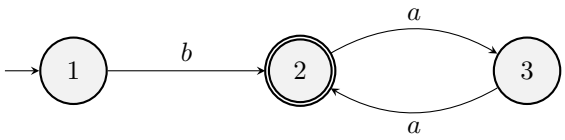
1.)

Since w does not contain the substring ab all a's must be after all b's. An NFA for L_1:



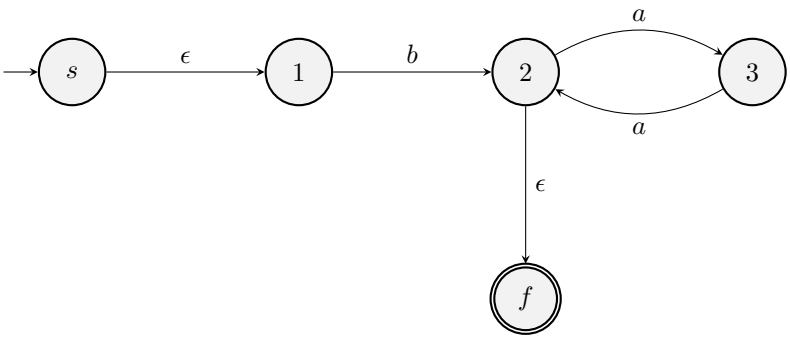
2.)

An NFA for $L_1 \cap L_2$:

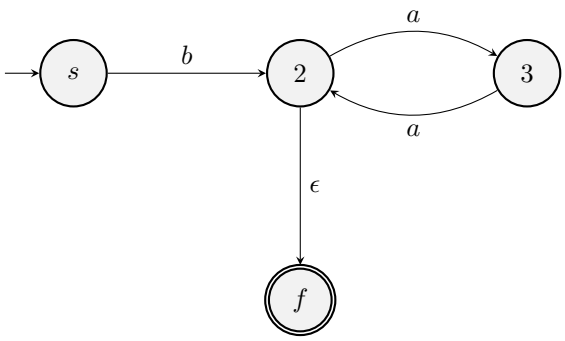


3.)

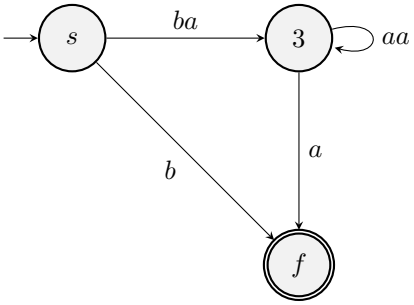
GNFA for $L_1 \cap L_2$ ($\emptyset$-transitions are not shown):



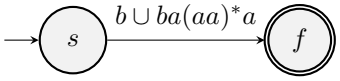
With 1 removed:



With 2 removed:



With 3 removed:



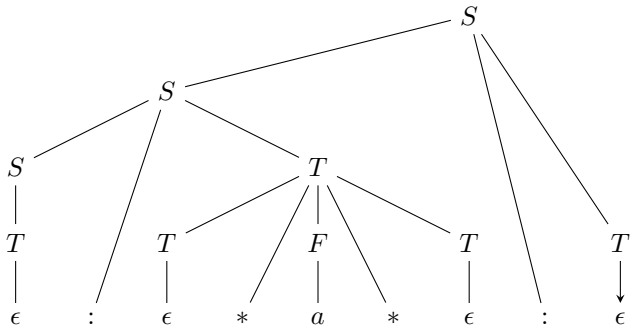
Exercise 2

1.)

a) Has no derivation

b) Has no derivation

c) Has a derivation



2.)

Step 1: Insert new start rule

$$
\begin{aligned}
&S_0 \to S \\
&S \to T \mid S : T \\
&T \to T * F * T \mid \epsilon \\
&F \to a \mid [S]
\end{aligned}
$$

Step 2: Remove $A \to \epsilon$ rules

$$
\begin{aligned}
&S_0 \to S \mid \epsilon \\
&S \to T \mid S : T \mid : T \\
&T \to T * F * T \mid *F * T \mid T * F* \mid *F* \\
&F \to a \mid [S] \mid []
\end{aligned}
$$

Step 3: Remove $A \to B$ rules

$$
\begin{aligned}
&S_0 \to S : T \mid : T \mid T * F * T \mid *F * T \mid T * F* \mid *F* \mid \epsilon \\
&S \to S : T \mid : T \mid T * F * T \mid *F * T \mid T * F* \mid *F* \\
&T \to T * F * T \mid *F * T \mid T * F* \mid *F* \\
&F \to a \mid [S] \mid []
\end{aligned}
$$

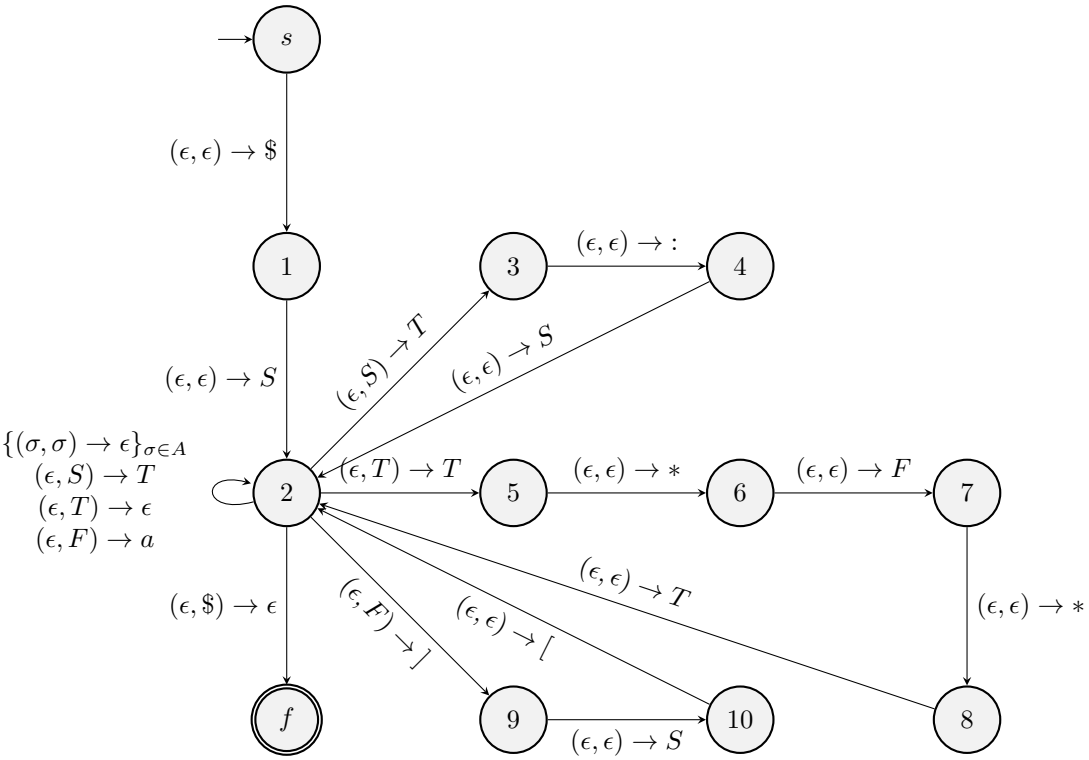
Step 4: Split $A \to u_1 u_2 ... u_k$ rules where $k \geq 2$

$$
\begin{aligned}
&S_0 \to SA_1 \mid :T \mid TA_2 \mid *A_3 \mid TA_5 \mid *A_6 \mid \epsilon \\
&S \to SA_1 \mid :T \mid TA_2 \mid *A_3 \mid TA_5 \mid *A_6 \\
&T \to TA_2 \mid *A_3 \mid TA_5 \mid *A_6 \\
&F \to a \mid [A_7 \mid [B_1 \\
&A_1 \to :T \\
&A_2 \to *A_3 \\
&A_3 \to FA_4 \\
&A_4 \to *T \\
&A_5 \to *A_6 \\
&A_6 \to F* \\
&A_7 \to S] \\
&B_1 \to]
\end{aligned}
$$

Step 5: Remove $A \to uB$ and $A \to Bu$ rules

$$
\begin{aligned}
&S_0 \to SA_1 \mid B_2T \mid TA_2 \mid B_3A_3 \mid TA_5 \mid B_3A_6 \mid \epsilon \\
&S \to SA_1 \mid B_2T \mid TA_2 \mid B_3A_3 \mid TA_5 \mid B_3A_6 \\
&T \to TA_2 \mid B_3A_3 \mid TA_5 \mid B_3A_6 \\
&F \to a \mid B_4A_7 \mid B_4B_1 \\
&A_1 \to B_2T \\
&A_2 \to B_3A_3 \\
&A_3 \to FA_4 \\
&A_4 \to B_3T \\
&A_5 \to B_3A_6 \\
&A_6 \to FB_3 \\
&A_7 \to SB_1 \\
&B_1 \to] \\
&B_2 \to : \\
&B_3 \to * \\
&B_4 \to [
\end{aligned}
$$

3.)



where $A = \{a, :, *, [,]\}$

Exercise 3

Assume that L is a context free language. That means there exist a $p \geq 1$ such that any $s \in L$ where $|s| \geq p$ can be split $s = uvxyz$ which fulfils the following conditions of the pumping lemma of context free languages.

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Let $s = a^q$ where q is prime and $q \geq p$. This is always possible since there's an infinite amount of primes. Clearly $|s| \geq p$ and $s \in L$. Setting $i = |s| + 1 = q + 1$, Condition 3. ensures that $|uv^{q+1}xy^{q+1}z| = |uvxyz| + q \cdot |uv| = q + q \cdot |uv| = q(1 + |uv|)$ is prime. This, in turn, implies that $|vy| = 0$, which contradicts Condition 1.

Exercise 4

1.)

$$[\text{NEQ-TRUE}_{BSS}] \qquad s \vdash a_1 \neq a_2 \rightarrow_B tt \qquad \text{if} \quad \begin{array}{l} s \vdash a_1 \rightarrow_A v_1 \\ s \vdash a_2 \rightarrow_A v_2 \\ v_1 \neq v_2 \end{array}$$

$$[\text{NEQ-FALSE}_{BSS}] \qquad s \vdash a_1 \neq a_2 \rightarrow_B ff \qquad \text{if} \quad \begin{array}{l} s \vdash a_1 \rightarrow_A v_1 \\ s \vdash a_2 \rightarrow_A v_2 \\ v_1 = v_2 \end{array}$$

$$[\text{DIV-TRUE}_{BSS}] \qquad s \vdash a_1[div]a_2 \rightarrow_B tt \qquad \text{if} \quad \begin{array}{l} s \vdash a_1 \rightarrow_A v_1 \\ s \vdash a_2 \rightarrow_A v_2 \\ v_1 | v_2 \end{array}$$

$$[\text{DIV-FALSE}_{BSS}] \qquad s \vdash a_1[div]a_2 \rightarrow_B ff \qquad \text{if} \quad \begin{array}{l} s \vdash a_1 \rightarrow_A v_1 \\ s \vdash a_2 \rightarrow_A v_2 \\ v_1 \nmid v_2 \end{array}$$

$$[\text{LOGICEQ-TRUE}_{BSS}] \qquad \frac{s \vdash b_1 \rightarrow_B v_1 \quad s \vdash b_2 \rightarrow_B v_2}{s \vdash b_1 \leftrightarrow b_2 \rightarrow_B tt} \qquad \text{if } v_1 = v_2$$

$$[\text{LOGICEQ-FALSE}_{BSS}] \qquad \frac{s \vdash b_1 \rightarrow_B v_1 \quad s \vdash b_2 \rightarrow_B v_2}{s \vdash b_1 \leftrightarrow b_2 \rightarrow_B ff} \qquad \text{if } v_1 \neq v_2$$

$$[\text{XOR-TRUE}_{BSS}] \qquad \frac{s \vdash b_1 \rightarrow_B v_1 \quad s \vdash b_2 \rightarrow_B v_2}{s \vdash b_1[eor]b_2 \rightarrow_B tt} \qquad \text{if } v_1 \neq v_2$$

$$[\text{XOR-FALSE}_{BSS}] \qquad \frac{s \vdash b_1 \rightarrow_B v_1 \quad s \vdash b_2 \rightarrow_B v_2}{s \vdash b_1[eor]b_2 \rightarrow_B ff} \qquad \text{if } v_1 = v_2$$

$$[\text{LIT-FALSE}_{BSS}] \qquad s \vdash \bot \rightarrow_B ff$$

2.)

Assuming $\mathbb{B} = \{tt, ff\}$ is our boolean values for true and false, the big-step transition system for boolean expressions is a triple $(\Gamma_B, \rightarrow_B, T_B)$ where

- $\Gamma_B = \mathbf{Bexp} \cup \mathbb{B}$ is the set of configurations.
- $T_B = \mathbb{B}$ is the set of end-configurations and $T_B \subseteq \Gamma_B$.
- $\rightarrow_B$ is defined by the rules in 1.) above.

Exercise 5

Solution not provided because call-by-name is only partially exam relevant, see last lecture video for more details.

Exercise 6

```
01 begin
02    var x:=2;
03    var y:=6;
04    proc p is x:=x+1;
05    proc q is call p;
06    begin
07        var x:=8;
08        proc p is x:=x+1;
09        call q;
10        y:=x
11    end
12 end
```

1.)

With fully dynamic scope rules, it is the last declared variables and procedures that are used. This means that it is p in line 8 that is used when q is called in line 9. Similarly, it is the last known x that is used inside p, which would be x in line 7. Therefore the sequence when calling q in line 9 is:

1. The q procedure gets called in line 9
2. The q procedure calls the p procedure in line 8
3. The p procedure increments the variable x in line 7 by 1
4. Variable y is set to 9

2.)

With dynamic scope rules for procedures, it is the last p procedure declared that is used inside q. With static scope rules for variables, it is the x known at the time of declaring procedure p that is incremented. Therefore the sequence when calling q in line 9 is the same as with fully dynamic scope rules:

1. The q procedure gets called in line 9
2. The q procedure calls the p procedure in line 8
3. The p procedure increments the variable x in line 7 by 1
4. Variable y is set to 9

3.)

With fully static scope rules, the procedure p which is called inside q in line 5, is the procedure p known at the time procedure q was declared, i.e procedure p declared in line 4. Similarily, the x in the body of procedure p, is the x known at the time of declaring p, i.e x in line 2. That means:

1. The q procedure is called in line 9
2. The q procedure will call p declared in line 4
3. The x in the p procedure is the x known at the time of declaration, i.e. line 2
4. The x of the *outer scope* is set to 3
5. y is set to 8 since the x of the *inner scope* is still 8