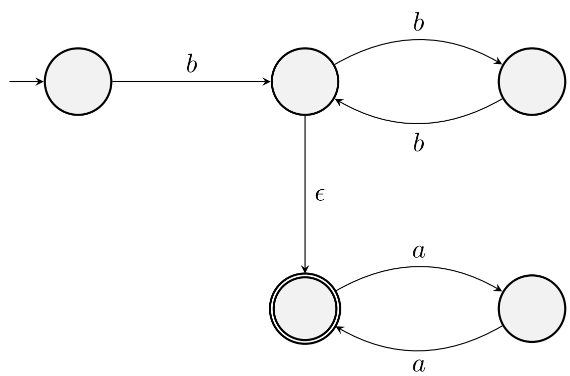
Exercise 1

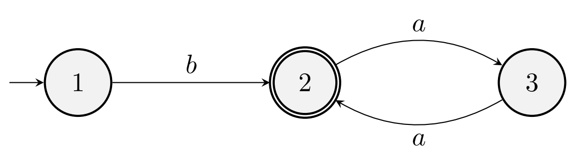
1.)

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



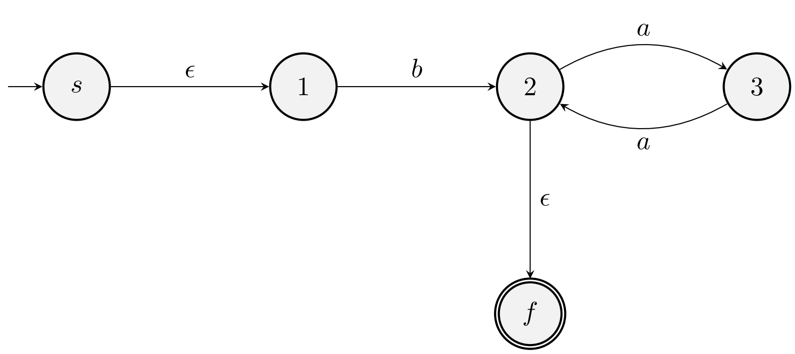
2.)

An NFA for $L_1 \cap L_2$:

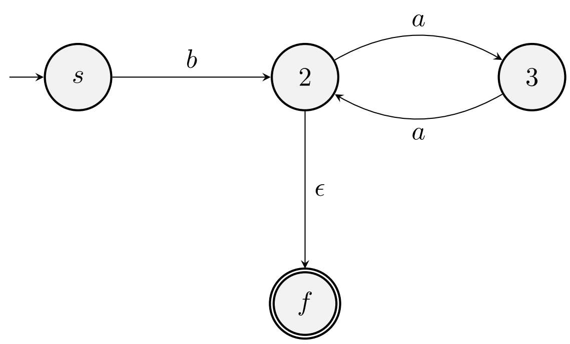


3.)

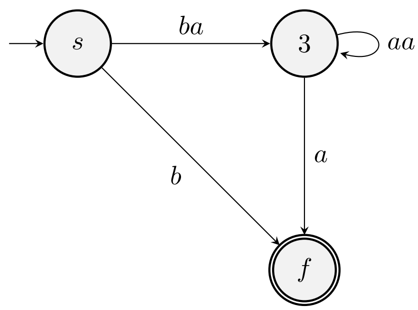
GNFA for $L_1 \cap L_2$ ($\emptyset$-transitions are not shown):



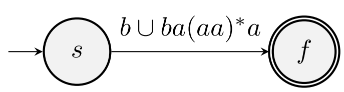
With 1 removed:



With 2 removed:



With 3 removed:



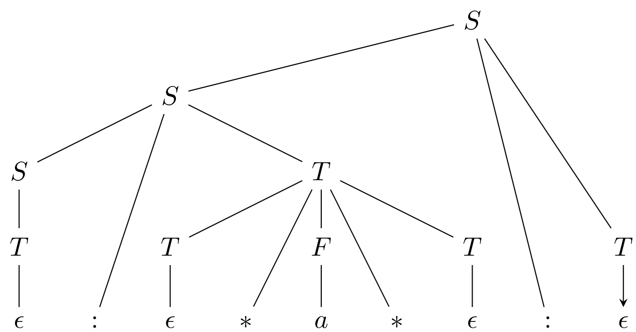
Exercise 2

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

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

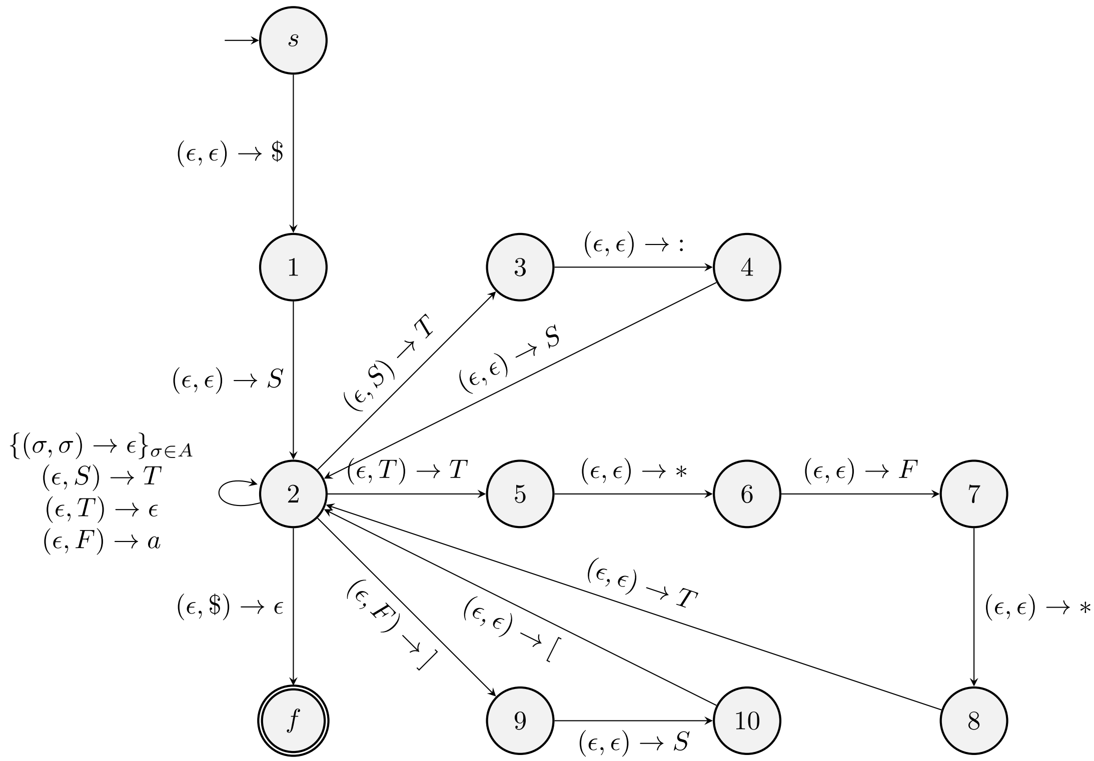
Step 4: Split $A \to u_1u_2...u_k$ rules where $k \geq 2$

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

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.)

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

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

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

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

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

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

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

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

$$[\text{LIT-FALSE}_{BSS}] \qquad s \vdash \perp \to_B \mathit{ff}$$

2.)

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

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

Exercise 5

Solution not provided because call-by-name is not exam relevant.

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