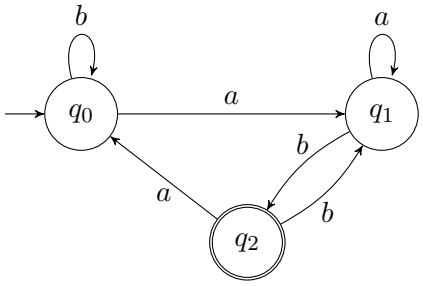
Lecture L&M 4

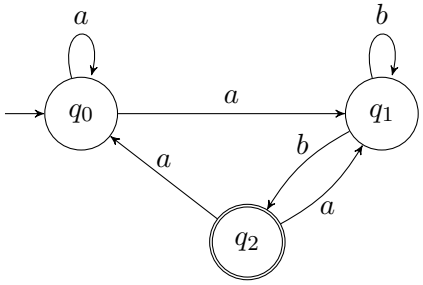
Composition of Automata, Context-Free Grammars

4.1 Projection and cartesian product

Let M_1 be the automaton defined by the following state diagram:

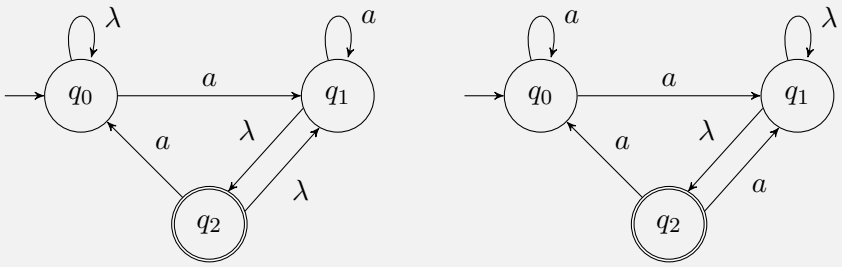


Let M_2 be the automaton defined by the following state diagram:



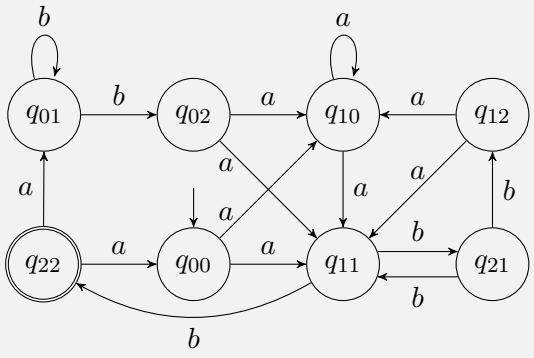
A. Draw the state diagrams of $M_1 \restriction \{a\}$ and $M_2 \restriction \{a\}$.

On the left the projection on $\{a\}$ of M_1, on the right the one of M_2:



B. Draw the state diagram of $M_1 \times M_2$.

The product automaton is given below. We write q_{ij} for the pair of states q_i from M_1 and q_j from M_2.



Actually there are also a steps from (the unreachable node) q_{20} to q_{00} and q_{01} (not shown).

A state q_{ij} from the product automaton is accepting if q_i and q_j are *both* accepting (in the respective automata). This is necessary because we want the language of the product to be the intersection of the languages of the original automata.

4.2 Context-free grammars

Do the following exercises from the book:

C. Let G be the grammar

$$\begin{aligned} S &\rightarrow abSc \mid A \\ A &\rightarrow cAd \mid cd. \end{aligned}$$

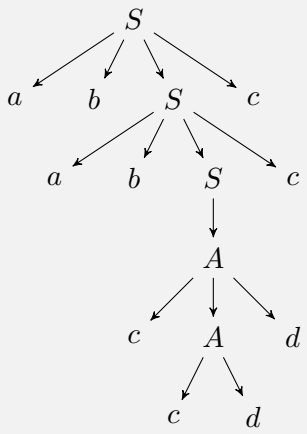
(a) Give the derivation of *ababccddcc*.

(b) Build the derivation tree for the derivation in part (a).

(c) Use set notation to define $L(G)$.

a. $\begin{aligned} S &\Rightarrow abSc \\ &\Rightarrow ababScc \\ &\Rightarrow ababAcc \\ &\Rightarrow ababcAdcc \\ &\Rightarrow ababccddcc \end{aligned}$

b.



c. For example: $L(G) = \{(ab)^n c^m d^m c^n \mid n \geq 0, m > 0\}$

D. Let G be the grammar

$$\begin{array}{lcl} S & \rightarrow & SAB \mid \lambda \\ A & \rightarrow & aA \mid a \\ B & \rightarrow & bB \mid \lambda. \end{array}$$

(a) Give a leftmost derivation of *abbaab*.

(b) Give two leftmost derivations of *aa*.

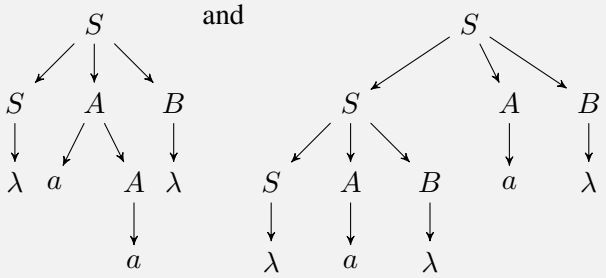
(c) Build the derivation tree for the derivations in part (b).

(d) Give a regular expression for $L(G)$.

a. For example:
$$\begin{aligned} S &\Rightarrow SAB \\ &\Rightarrow SABAB \\ &\Rightarrow ABAB \\ &\Rightarrow aBAB \\ &\Rightarrow abBAB \\ &\Rightarrow abbBAB \\ &\Rightarrow abbAB \\ &\Rightarrow abbaAB \\ &\Rightarrow abbaaB \\ &\Rightarrow abbaabB \\ &\Rightarrow abbaab \end{aligned}$$

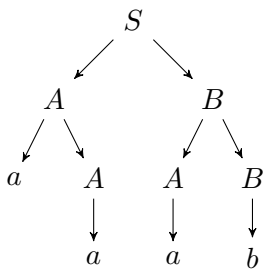
b. For example:
$$\begin{aligned} S &\Rightarrow SAB \\ &\Rightarrow AB \\ &\Rightarrow aAB \\ &\Rightarrow aaB \\ &\Rightarrow aa \end{aligned}$$
and
$$\begin{aligned} S &\Rightarrow SAB \\ &\Rightarrow SABAB \\ &\Rightarrow ABAB \\ &\Rightarrow aBAB \\ &\Rightarrow aAB \\ &\Rightarrow aaB \\ &\Rightarrow aa \end{aligned}$$

c.



d. For example: $(a^+b^*)^*$

E. Let DT be the derivation tree



(a) Give a leftmost derivation that generates the tree DT.

(b) Give a rightmost derivation that generates the tree DT.

a) $S \Rightarrow AB \Rightarrow aAB \Rightarrow aaB \Rightarrow aaAB \Rightarrow aaaB \Rightarrow aaab$

b) $S \Rightarrow AB \Rightarrow AAB \Rightarrow AAb \Rightarrow Aab \Rightarrow aAab \Rightarrow aaab$

F. Construct a grammar over $\{a, b\}$ whose language is $\{a^m b^n \mid 0 \leq n \leq m \leq 3n\}$.

We must ensure the right number of optional a:

$$\begin{array}{lcl} S & \rightarrow & aAASb \mid \lambda \\ A & \rightarrow & a \mid \lambda \end{array}$$

An alternative:

$$\begin{array}{lcl} S & \rightarrow & ASb \mid \lambda \\ A & \rightarrow & a \mid aa \mid aaa \end{array}$$

G. For the following grammar, give a regular expression or set-theoretic definition for the language of the grammar. Show that the grammar is ambiguous and construct an equivalent unambiguous grammar.

$$\begin{array}{lcl} S & \rightarrow & aSA \mid \lambda \\ A & \rightarrow & bA \mid \lambda \end{array}$$

b.
- A regular expression: $a^+b^* \cup \lambda$.
- The string aab has two (leftmost) derivations:

$S \Rightarrow aSA \Rightarrow aaSAA \Rightarrow aaAA \Rightarrow aabA \Rightarrow aab$

$S \Rightarrow aSA \Rightarrow aaSAA \Rightarrow aaAA \Rightarrow aaA \Rightarrow aab$

- The following grammar is unambiguous:

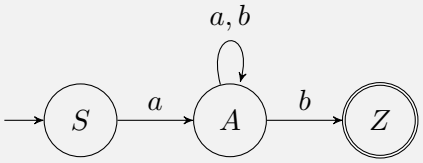
$$\begin{array}{lcl} S & \rightarrow & aS \mid aB \mid \lambda \\ B & \rightarrow & bB \mid b \end{array}$$

This follows from the fact that the options for S lead to the (disjoint) sub-languages $a(a^+b^* \cup \lambda)$, ab^+ and λ, and those of B to strings with various numbers of bs.

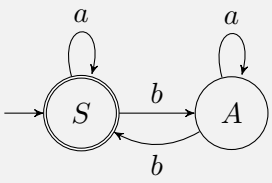
4.3 From regular grammars to NFA

For each of the following grammars G, construct an NFA M such that $\mathcal{L}(M) = \mathcal{L}(G)$.

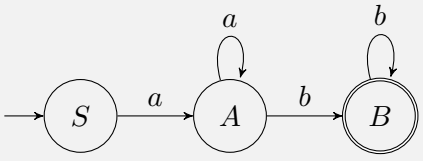
H. G is the regular grammar: $\begin{array}{lcl} S & \rightarrow & aA \\ A & \rightarrow & aA \mid bA \mid b. \end{array}$



I. G is the regular grammar $\begin{array}{lcl} S & \rightarrow & aS \mid bA \mid \lambda \\ A & \rightarrow & aA \mid bS. \end{array}$



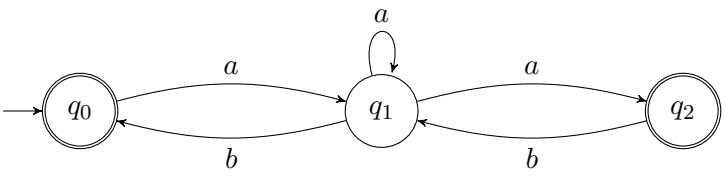
J. *(extra)* G is the regular grammar $\begin{array}{lcl} S & \rightarrow & aA \\ A & \rightarrow & aA \mid bB \\ B & \rightarrow & bB \mid \lambda. \end{array}$



4.4 From NFA to regular grammars

Now, for each of the following NFA M, construct a regular grammar G such that $\mathcal{L}(G) = \mathcal{L}(M)$.

K. M is the NFA:



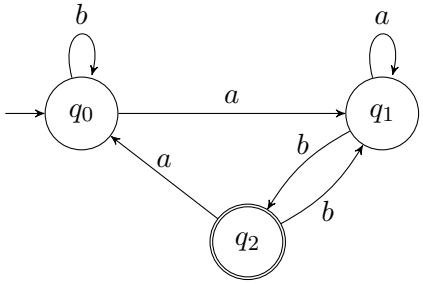
States become variables, transitions become rules, and accepting states produce an extra λ-rule. The initial state becomes the start variable.

The CFG $G := (\{Q_0, Q_1, Q_2\}, \{a, b\}, P, Q_0)$, with the following set of rules P:

$$\begin{array}{lcl} Q_0 & \rightarrow & aQ_1 \mid \lambda \\ Q_1 & \rightarrow & aQ_1 \mid bQ_0 \mid aQ_2 \\ Q_2 & \rightarrow & bQ_1 \mid \lambda \end{array}$$

Note that G is automatically a regular grammar.
Also note that for, for example, Q_0, there are actually two rules.

L. M is the NFA:

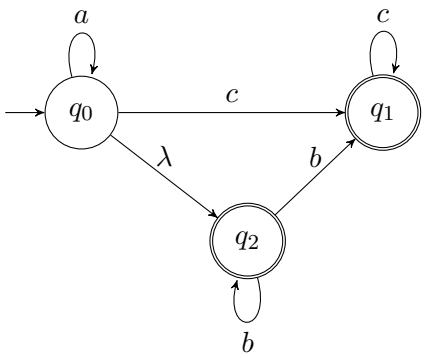


The CFG $G := (\{Q_0, Q_1, Q_2\}, \{a, b\}, P, Q_0)$, with the rules P:

$$\begin{array}{lcl} Q_0 & \rightarrow & bQ_0 \mid aQ_1 \\ Q_1 & \rightarrow & aQ_1 \mid bQ_2 \\ Q_2 & \rightarrow & aQ_0 \mid bQ_1 \mid \lambda \end{array}$$

M. *(extra)* M is the NFA:

(*Note:* First remove the λ-transitions!)



This construction only works for NFA, and not for NFA-λ!
This is not stated clearly in the book, and was not said during the lecture.

Note: if you were to do this with the λ-steps present, you would obtain:

$$\begin{array}{lcl} Q_0 & \to & aQ_0 \mid cQ_1 \mid \lambda Q_2 \\ Q_1 & \to & cQ_1 \mid \lambda \\ Q_2 & \to & bQ_2 \mid bQ_1 \mid \lambda \end{array}$$

Apparently $L(G) = L(M)$ applies, but the grammar is not regular, since λQ_2 is not of the form a, λ of aX. Simplifying λQ_2 to Q_2 offers no help either.

Option 1 would be to replace Q_2 in the rhs of Q_0 by the rhs of Q_2. That preserves the language, and after that, it's regular: You will obtain CFG $G_1 := (\{Q_0, Q_1, Q_2\}, \{a, b, c\}, P_1, Q_0)$, with rules P_1:

$$\begin{array}{lcl} Q_0 & \to & aQ_0 \mid cQ_1 \mid \underline{bQ_2 \mid bQ_1 \mid \lambda} \\ Q_1 & \to & cQ_1 \mid \lambda \\ Q_2 & \to & bQ_2 \mid bQ_1 \mid \lambda \end{array}$$

The official method is to remove the λ transitions. This can be done by determinising the NFA-λ. The input transition function will then be as follows:

	a	b	c
q_0	$\{q_0\}$	$\{q_1, q_2\}$	$\{q_1\}$
q_1	$\varnothing$	$\varnothing$	$\{q_1\}$
q_2	$\varnothing$	$\{q_1, q_2\}$	$\{q_1\}$

The automaton that is being constructed by this has states

$$\begin{aligned} Q_0 &= \{q_0, q_1\} \\ Q_1 &= \{q_1\} \\ Q_2 &= \{q_1, q_2\} \\ Q_3 &= \varnothing \end{aligned}$$

The initial state is Q_0 and the accepting states are $\{Q_0, Q_1, Q_2\}$.

The CFG that corresponds to this is $G_2 := (\{Q_0, Q_1, Q_2, Q_3\}, \{a, b, c\}, P_2, Q_0)$, with rules P_2:

$$\begin{array}{lcl} Q_0 & \to & aQ_0 \mid bQ_2 \mid cQ_1 \mid \lambda \\ Q_1 & \to & cQ_1 \mid aQ_3 \mid bQ_3 \mid \lambda \\ Q_2 & \to & bQ_2 \mid cQ_1 \mid aQ_3 \mid \lambda \\ Q_3 & \to & aQ_3 \mid bQ_3 \mid cQ_3 \end{array}$$

Note that $L(Q_3) = \varnothing$, because you can never get rid of the symbol Q_3: there is no terminating rule for Q_3. That's why you can simplify this to $G_3 := (\{Q_0, Q_1, Q_2\}, \{a, b, c\}, P_3, Q_0)$, with rules P_3:

$$\begin{array}{lcl} Q_0 & \rightarrow & aQ_0 \mid bQ_2 \mid cQ_1 \mid \lambda \\ Q_1 & \rightarrow & cQ_1 \mid \lambda \\ Q_2 & \rightarrow & bQ_2 \mid cQ_1 \mid \lambda \end{array}$$