Theoretical Computer Science 1
Exercise Sheet 6

René Maseli
Thomas Haas
TU Braunschweig
Winter Semester 2023/24

Release: 2024-01-19 Due: 2024-02-01 23:59

Hand in your solutions to the Vips directory of the StudIP course until Thursday, February 1st 2024 23:59. You should provide your solutions either directly as .pdf file or as a readable scan/photo of your handwritten notes. Submit your results as a group of four and state **all** members of your group with **student id, name and course**.

Homework Exercise 1: The syntax of programming languages as grammar [9 points]

The syntax of a programming language is usually formulated with a context-free grammar (oftentimes expressed in EBNF or a syntax diagram). In this exercise you will construct a grammar which describes the syntax of a simple programming language.

a) [3 points] Give a context-free grammar G such that its language $\mathcal{L}(G)$ consists of the set of syntactically correct programs as described below.

- Use the terminals $\Sigma := \{\texttt{id}, \texttt{num}, \texttt{;}, \texttt{op}, \texttt{=}, \texttt{(}, \texttt{)}, \texttt{if}, \texttt{else}, \texttt{while}, \texttt{end}, \texttt{break}\}$.
  Hereby, `id`, `num` and `op` are placeholders for possible variable names, natural numbers and binary operators (including ==). The other symbols represent single keywords and symbols.
- An **expression** consists of variables, numbers and parenthesized binary operations, e.g. `(x+2)`, `(z<500)`, `(x*(y/3))`, `(x==(y+1))`.
- A **program** is either
  - empty
  - a variable definition (e.g. `x=(y+1)`)
  - a conditional branching (e.g. `if x y=(z/x) else y=z end`)
  - a loop (e.g. `while (x<100) x=(x*12) end`)
  - a break `break` **but only inside a loop**
  - a `;`-delimited sequence of programs (e.g. `x=12 ; y=500`)

b) [1 point] Derive the following program from your grammar in part a) starting from the initial symbol. Additionally to the start symbol and the resulting word, give at least three intermediate words of your derivation sequence.
`while (x<y) x=(x<<1) ; if (y==z) break else y=(y+1) end end`
(First you have to replace each variable with `id`, each number with `num` and the operations by `op`.)

c) [2 points] Use the pumping lemma to prove that $\mathcal{L}(G)$ is not regular.

d) [3 points] Modify G into another grammar G', such that the programming language prohibits obvious dead code: `break` jumps out of the current program sequence, so it should not preceed any other program instruction. This also applies to conditional branches, if both subprograms end with `break`. This can even nest indefinetly.

Homework Exercise 2: CFG, CNF, CYK [10 points]

The Cocke-Younger-Kasami-algorithm (CYK algorithm) assumes as input a context-free grammar (CFG) in Chomsky normal form (CNF). This means that all production rules are of the form $X \to YZ$ (for non-terminals Y and Z) or of the form $X \to a$ (for a terminal a).

a) [6 points] Use the procedure introduced in the lecture to construct a grammar G_a in CNF, which satisfies $\mathcal{L}(G_a) = \mathcal{L}(G) \setminus \{\varepsilon\}$. The grammar $G = \langle\{S, X, Y\}, \{a, b, c\}, P, S\rangle$ is defined by the following productions:

$$S \to XcX \qquad X \to a \mid YX \mid YbYb \qquad Y \to bc \mid X \mid XX$$

With the use of your grammar and the CYK algorithm, decide whether the word *abcbca* is produced by G.

b) [4 points] Use the CYK algorithm to decide, whether the words *babaa* and *baba* are produced by the grammar with the following productions.

$$S \to AB \mid BC \qquad A \to a \mid CC \qquad B \to b \mid BA \qquad C \to a \mid AB$$

Homework Exercise 3: Greibach normal form [10 points]

Transform the following context-free grammar G into GNF.

$$S \to WS \mid UU \qquad U \to b \mid c \mid UW \qquad V \to c \qquad W \to a \mid VW \mid VU$$

a) [4 points] For each pair of non-terminal $X \in N$ and terminal $s \in \Sigma$, give a regular expression for the language $L_{X,s} \subseteq N^*$ of the suffixes of terms $\alpha \in N^*$ which are produced by the strong left derivation: $X \Rightarrow^*_{SL} s\alpha$.

b) [2 points] Find for all non-empty languages $L_{X,s}$ a right-linear grammar $G_{X,s}$ over **terminal symbols** N and with initial symbol $T_{X,s}$. Create new non-terminals, if required.

c) [2 points] Transform the union of G and all your grammars into pseudo-GNF, by letting this new grammar G' guess each next terminal symbol, like has been shown in the lecture. Ensure, that $\mathcal{L}(G') = \mathcal{L}(G)$ holds. A proof is not necessary. Try to avoid useless non-terminals.

d) [2 points] Eliminate all ε-productions from G', to form a grammar G'' in GNF, which satisfies $\mathcal{L}(G'') = \mathcal{L}(G') \setminus \{\varepsilon\}$.

Homework Exercise 4: Pushdown automata [11 points]

Construct pushdown automata for the following languages and state which acceptance condition (empty stack or final states) you assume.

Remark Note that whenever multiple symbols are pushed, the last gets to be the new top.

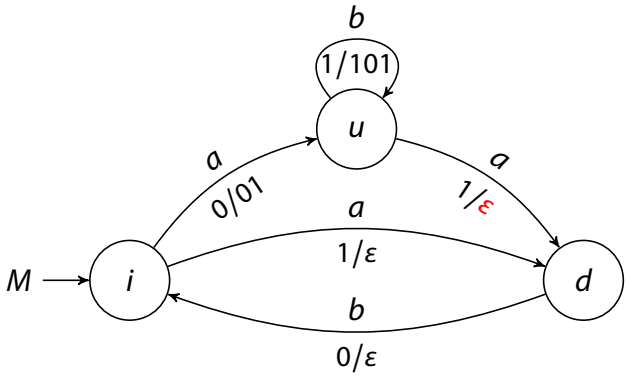
a) [2 points] $L_1 = \{ w \in \{a, b, (,)\}^* \mid w \text{ is correctly parenthesized } \}$.

b) [2 points] $L_2 = \{ w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b \}$.

c) [2 points] Can you construct a PDA, which accepts $L_1 \cap L_2$?
If not, what is the intuitive problem here?

$$L_1 \cap L_2 = \{ w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b \text{ and } w \text{ is correctly parenthesized } \}$$

d) [5 points] Consider the Pushdown Automaton $M = \langle \{i, u, d\}, \{a, b\}, \{0, 1\}, i, 0, \delta \rangle$, with empty-stack acceptance and whose transition relation δ is given by the following diagram.
Find a contextfree grammar G with $\mathcal{L}(M) = \mathcal{L}(G)$, by using the triple construction from the lecture.

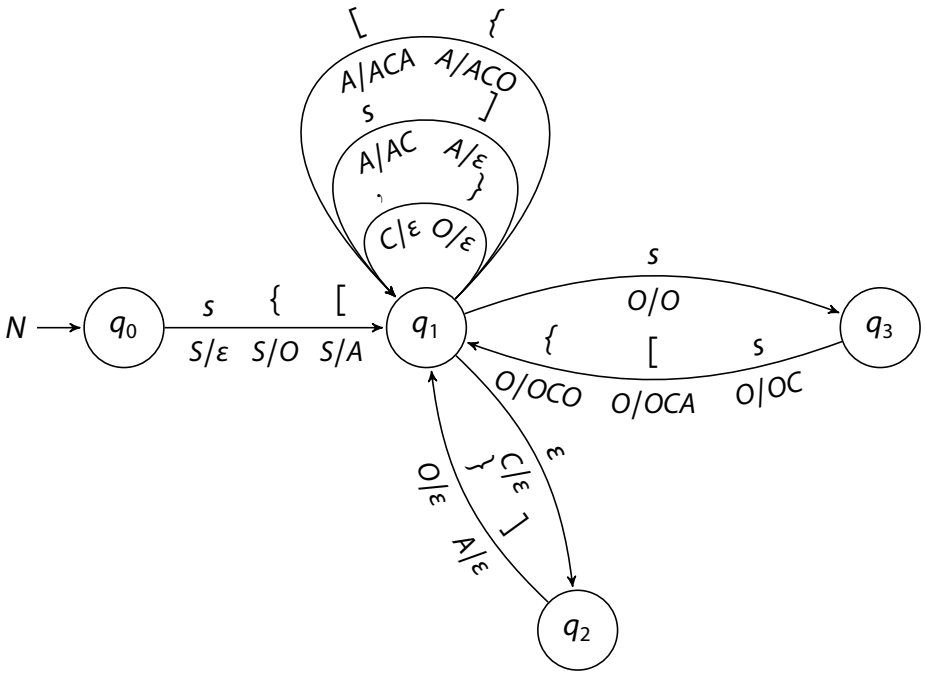


Exercise 5:

JavaScript Object Notation (JSON) is a description language for structured collections of serializable data, which is applied in numerous web technologies. Alongside some primitive datatypes, they can also express lists (arrays) and associative containers (objects).

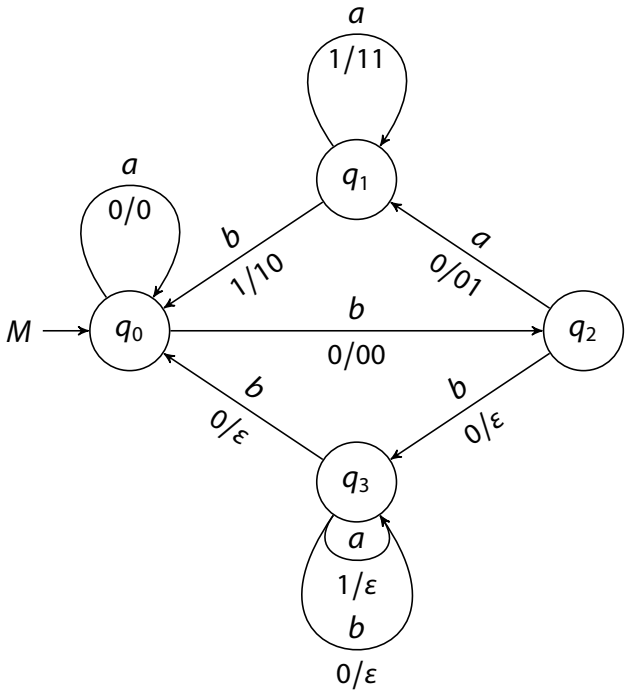
a) Construct pushdown automata M for the following language L and state which acceptance condition (empty stack or final states) you assume. Do not just give context-free grammars. You do not need to prove the correctness of your construction.
Consider a simplified variant of JSON over $\{a, b, \{, \}\}$: ‚Objects' start and end with fitting curly braces ‚{' and ‚}'. Inside, there is an arbitrary number of key-value pairs. Keys are words of $a.b^*$ and may not be unique inside the same object. Values are either words of $a.b^*$, or again objects. The automaton M shall accept exactly the well-formed objects.
For example, $\{abbababb\{\}\} \in L$ and $\{abb\{ab\{aba\}a\{abbab\}\}\} \in L$ have to be accepted, but neither $\{ababab\} \notin L$, $abb\{aa\} \notin L$ nor $\{ab\{\} \notin L$.

b) Describe the behavior of the following pushdown automaton N, with empty-stack acceptance, by explaining the role of all states and stack symbols with one sentence, each.



Exercise 6:

Consider the Pushdown Automaton $M = \langle \{q_0, q_1, q_2, q_3\}, \{a, b\}, \{0, 1\}, q_0, 0, \delta \rangle$, with empty-stack acceptance and whose transition relation δ is given by the following diagram.



a) Consider just the states on their own, to answer those two questions. Which two states are befitting destinations $q \in Q$ in triples like $\langle p, s, q \rangle$? Which five pairs of $p \in Q$ and $s \in \Gamma$ are enabled?

b) Find a contextfree grammar G with $\mathcal{L}(M) = \mathcal{L}(G)$, by using the triple construction from the lecture.

Exercise 7:

Given the two CFG $G = \langle\{S, W, X\}, \{a, b\}, P_G, S\rangle$ and $H = \langle\{S, U, V\}, \{a, b, c\}, P_H, S\rangle$.

$$P_G : S \to \varepsilon \mid bW$$
$$W \to a \mid XXb$$
$$X \to SS \mid ab$$

a) Use the procedure introduced in the lecture to construct a grammar G_a without ε productions, which satisfies $\mathcal{L}(G_a) = \mathcal{L}(G) \setminus \{\varepsilon\}$.

b) Use G_a and the procedure from the lecture to construct a grammar G_b in CNF with $\mathcal{L}(G_b) = \mathcal{L}(G) \setminus \{\varepsilon\}$.

c) Use G_b and the CYK algorithm to decide whether the word *bbaab* is produced by G.

d) Use G_b and the CYK algorithm to decide whether $bbababb \in \mathcal{L}(G)$ is true.

$$P_H : S \to UVab \mid bU$$
$$U \to aV \mid aUSc$$
$$V \to \varepsilon \mid bSc \mid U$$

e) Use the procedure from the lecture to construct a grammar H_e in CNF, that satisfies $\mathcal{L}(H_e) = \mathcal{L}(H) \setminus \{\varepsilon\}$.

f) Use H_e and the CYK algorithm to decide whether the word *aaabca* is produced by H.