Problem Set 2

Instructions

Resources useful to solve the exercises in this problem set are the following:

Guille Godoy's video lectures

- Gramáticas incontextuales
- Operaciones sobre gramáticas
- Depuración de gramáticas (1)
- Depuración de gramáticas (2)
- Depuración de gramáticas (3)

Books

- (Sipser 2013, § 2.1 Content-free Grammars)
- (Cases and Màrquez 2003, § 2 and § 3)
- (Hopcroft, Motwani, and Ullman 2007, § 5)

Cases, Rafel, and Lluís Màrquez. 2003. *Llenguatges, Gramàtiques i Autòmats : Curs Bàsic*. 2a ed. Edicions UPC.

Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. 2007. *Introduction to Automata Theory, Languages, and Computation*. 3rd edition. Pearson Addison Wesley.

Sipser, Michael. 2013. *Introduction to the Theory of Computation*. 3rd edition. Cengage Learning.

All exercises

Exercise 2.1 (Just context-free, or actually regular?)**.** Later in the course we will see that no algorithm exists that always terminates and gives the answer to the question "*Given a context-free grammar, does it generate a regular language?*".

For each of the context-free grammars below, find what is the language generated and prove whether it is a regular language **or not**.

(a) $\begin{cases} S \to AB \mid CD \\ A \to 0A0 \mid 0 \\ B \to 1B1 \mid \lambda \\ C \to 0C0 \mid \lambda \\ D \to 1D1 \mid \lambda \end{cases}$

(b) $\begin{cases} S \to aA \mid bB \mid \lambda \\ A \to Sa \mid Sb \\ B \to Sb \end{cases}$

(c) $\begin{cases} S \to AB \\ A \to 0A0 \mid 1 \\ B \to 1B1 \mid 0 \end{cases}$

(d) $\{ S \to 0S0 \mid 0S1 \mid \lambda$

(e) $\begin{cases} S \to AB \\ A \to 0A0 \mid 0A1 \mid \lambda \\ B \to 0B \mid 1B \mid \lambda \end{cases}$

(f) $\begin{cases} S \to A \mid B \\ A \to 0S0 \mid 1S1 \mid \lambda \\ B \to 0S1 \mid 1S0 \mid \lambda \end{cases}$

(g) $\begin{cases} S \to A \mid B \\ A \to 0A0 \mid 1A1 \mid \lambda \\ B \to 0B1 \mid 1B0 \mid \lambda \end{cases}$

(h) $\begin{cases} S \to aSa \mid bSb \mid X \\ X \to aXb \mid bXa \mid a \mid b \mid \lambda \end{cases}$

(i) $\begin{cases} S \to WXW' \\ X \to aX \mid bX \mid \lambda \\ W \to aW \mid bW \mid \lambda \\ W' \to W'a \mid W'b \mid \lambda \end{cases}$

Exercise 2.2 (Ambiguous context-free grammars)**.** Show that the following grammars are ambiguous.

(a)
$$\begin{cases} S & \to & aSb \mid B \\ B & \to & bAa \mid bCb \mid \lambda \\ A & \to & aAbA \mid bAaA \mid \lambda \\ C & \to & Aaa \mid aAa \mid aaA \end{cases}$$

(b)
$$\begin{cases} S & \to & aU_1 \mid aS \mid bZ_1 \mid bS \\ Z_1 & \to & aU_2 \mid bF \\ U_1 & \to & bU_2 \\ U_2 & \to & bF \mid b \\ F & \to & aF \mid bF \mid a \mid b \end{cases}$$

(c)
$$\begin{cases} S & \to & AaBA \mid ABaA \mid ACA \mid AbabA \\ B & \to & bb \\ C & \to & bB \\ A & \to & aA \mid bA \mid \lambda \end{cases}$$

(d)
$$\begin{cases} S & \to & aU_1 \mid aS \mid bZ_1 \mid bS \\ Z_1 & \to & aU_2 \mid bZ_2 \\ U_1 & \to & bU_2 \\ U_2 & \to & bF \\ Z_2 & \to & aF \mid bF \\ F & \to & aF \mid bF \mid \lambda \end{cases}$$

Exercise 2.3 (On Dyck languages)**.** In this exercise we consider the *Dyck language*, that is the language of well-balanced parentheses (and variations of it). More precisely, given the alphabet $\Sigma = \{(,)\}$, the *Dyck language* $\mathsf{dyck}(1)$ is

$$\mathsf{dyck}(1) = \{w \in \Sigma^* \mid \text{for every prefix } u \text{ of } w \;\; |u|_(\geq |u|_) \wedge |w|_(= |w|_)\}.$$

Similarly, let $\mathsf{dyck}(s)$ be the *Dyck language* on s pairs of parentheses, i.e. the language of correctly nested sequences of s distinct types of parentheses. For instance, given the two types of parentheses (,), and [,], the word ([]) $\in \mathsf{dyck}(2)$, ()[] $\in \mathsf{dyck}(2)$ but ([)] $\notin \mathsf{dyck}(2)$.

(a) Show that the grammar $S \to SS \mid (S) \mid \lambda$ generates exactly $\mathsf{dyck}(1)$. Is it ambiguous?

(b) Show that the grammar $S \to (S)S \mid \lambda$ generates exactly $\mathsf{dyck}(1)$. Is it ambiguous?

(c) Show that the grammar $S \to SS \mid (S) \mid [S] \mid \lambda$ generates exactly $\mathsf{dyck}(2)$. Is it ambiguous?

(d) Show that the grammar $S \to (S)S \mid [S]S \mid \lambda$ generates exactly $\mathsf{dyck}(2)$. Is it ambiguous?

(e) Construct an unambiguous grammar that generates $\mathsf{dyck}(s)$ for an arbitrary s.

(f) Let $\sigma = \{(,), [,]\}$ and L be the language of all words over Σ^* such that, ignoring the symbols [,], the words are well-parenthesised on (,), and, similarly, ignoring the symbols (,), the words are well-parenthesised on [,]. In particular $\mathsf{dyck}(2) \subseteq L$, but the containment is strict. For instance, ([)] $\in L \setminus \mathsf{dyck}(2)$. Show that L is not a context-free language.

> **Tip**
>
> Use the fact that the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ is not context-free.

> **Note**
>
> The Dyck languages are important because, in a sense, they are the *most complicated* context-free languages. Indeed, the **Chomsky–Schützenberger representation theorem** states that a language L is context-free if an only if L is the image under an homomorphism of some $\mathsf{dyck}(s)$ intersected with a regular language.

Exercise 2.4 (Context-free closure operations and ambiguity)**.** Given unambiguous context-free grammars G_1 and G_2,

(a) could the construction to obtain the grammar for the union $G_1 \cup G_2$ give an ambiguous grammar?
(b) could the construction to obtain the grammar for the concatenation $G_1 \cdot G_2$ give an ambiguous grammar?
(c) could the construction to obtain the grammar for the Kleene star G_1^* give an ambiguous grammar?
(d) could the construction to obtain the grammar for the reverse G_1^R give an ambiguous grammar?
(e) given also a homomorphism σ, could the construction to obtain the grammar for $\sigma(G_1)$ give an ambiguous grammar?

Exercise 2.5 (Depuration of grammars)**.**

(1) Given a grammar G, describe *an algorithm to eliminate λ-productions* (with only one possible exception) from G (without changing the generated language). Make sure that when G is unambiguous the algorithm described gives an unambiguous grammar. What is the cost of the algorithm?

(2) Given a grammar G, describe *an algorithm to eliminate unary production rules* from G (without changing the generated language). Make sure that when G is unambiguous the algorithm described gives an unambiguous grammar. What is the cost of the algorithm?

(3) Given a grammar G, describe *an algorithm to eliminate useless symbols* from G (without changing the generated language). Make sure that when G is unambiguous the algorithm described gives an unambiguous grammar. What is the cost of the algorithm?

(4) Apply the depuration algorithm (remove λ-productions, unary productions, unproductive and unreachable symbols) to the following grammars:

(a) $\left\{ \begin{array}{lcl} S & \to & SS \mid (S) \mid \lambda \end{array} \right.$

(b) $\left\{ \begin{array}{lcl} S & \to & (S)S \mid \lambda \end{array} \right.$

(c) $\left\{ \begin{array}{lcl} S & \to & AA \\ A & \to & AA \mid \lambda \end{array} \right.$

(d) $\left\{ \begin{array}{lcl} S & \to & A \\ A & \to & B \\ B & \to & c \end{array} \right.$

(e) $\left\{ \begin{array}{lcl} S & \to & AB \\ A & \to & a \mid \lambda \\ B & \to & b \mid \lambda \end{array} \right.$

(f) $\left\{ \begin{array}{lcl} S & \to & AB \\ A & \to & aAb \mid \lambda \\ B & \to & bBc \mid \lambda \end{array} \right.$

(g) $\left\{ \begin{array}{lcl} S & \to & BC \mid \lambda \\ A & \to & aA \mid \lambda \\ B & \to & bB \\ C & \to & c \end{array} \right.$

(h) $\left\{ \begin{array}{lcl} S & \to & X \mid Y\lambda \\ X & \to & Xc \mid A \\ A & \to & aAb \mid \lambda \\ Y & \to & aY \mid B \\ B & \to & bBc \mid \lambda \end{array} \right.$

(i) $\left\{ \begin{array}{lcl} S & \to & A \mid B \mid C \\ A & \to & SaSbS \mid \lambda \\ B & \to & SbSaS \mid \lambda \\ C & \to & Cc \mid \lambda \end{array} \right.$

Exercise 2.6 (On the Chomsky normal form)**.**

(a) Given a context-free grammar G, describe a polynomial time procedure to obtain a grammar G' producing the same language of G and in Chomsky Normal Form.

(b) Given a grammar G in Chomsky normal form and a word w produced by G, in how many steps is w produced? (as a function of $|w|$)

> **Note**
>
> Recall that a context-free grammar G is in *Chomsky normal form* if all of its production rules are of the form:
>
> $$A \to BC, \text{or}$$
> $$A \to a, \text{or}$$
> $$S \to \lambda,$$
>
> where A, B, and C are nonterminal symbols, the letter a is a terminal symbol, and S is the start symbol. Moreover, neither B nor C may be the start symbol S, and the production rule $S \to \lambda$ can only appear if λ is in the language produced by G.

Exercise 2.7 (Membership in a context-free language is decidable in polynomial time)**.** Consider the following decision problem:

$\texttt{Membership}_{\texttt{CFL}}$: given an input $x \in \{0,1\}^*$ and a context-free grammar G, determine whether $x \in L(G)$.

Show that $\texttt{Membership}_{\texttt{CFL}}$ can be decided in polynomial time w.r.t. $|x|$ and the size of G. Do this describing how the *Cocke-Kasami-Younger algorithm* (CKY) works.

> **Caution**
>
> The CKY algorithm assumes as input a grammar in Chomsky normal form.

Exercise 2.8 (Some decidable properties of context-free languages)**.** Let L_G be the (context-free) language produced by the context-free grammar G. Given as input the grammar G, describe an algorithm to decide whether

(a) L_G is empty.
(b) L_G is infinite.
(c) L_G contains some word of even length.
(d) L_G contains infinite words of even length.

What is the asymptotic cost of the algorithm proposed as a function of the number of symbols in G?

> **!** Important
>
> Later in the course we will see that a lot of very natural questions on context-free grammar do not have an algorithm to decide them. Not that we didn't find an algorithm yet. **No algorithm exists that always terminate and gives the answer.** For instance:
>
> - Given a context-free grammar, is it ambiguous?
> - Given a context-free grammar, does it describe a regular language?
> - Given a context-free grammar G with terminals $\{a, b\}$, is $L_G = \{a, b\}^*$?
> - Given two context-free grammars G_1 and G_2, is $L_{G_1} \cap L_{G_2}$ empty?
> - Given two context-free grammars G_1 and G_2, is $L_{G_1} = L_{G_2}$?
> - Given two context-free grammars G_1 and G_2, is $L_{G_1} \subseteq L_{G_2}$?

Exercise 2.9 (Complement of context-free sometimes is context-free)**.** In general context-free languages are not closed under complement, that is given a context-free language L it is not true in general that $\overline{L}$ is also context-free.

This exercise is about some context-free languages whose complement is actually also context-free.

(a) Give a non-ambiguous context-free grammar generating $L = \{a^n b^n \mid n \in \mathbb{N}\}$ and a context-free grammar generating $\overline{L}$. Can you make this latter grammar non-ambiguous?

> **Tip**
>
> Use RACSO to test whether your proposed grammar for $\overline{L}$ is non-ambiguous.

(b) Give a non-ambiguous context-free grammar generating $L = \{w \in \{a, b\}^* \mid w = w^R\}$ and a context-free grammar generating $\overline{L}$. Can you make this latter grammar non-ambiguous?

> **Tip**
>
> Use RACSO to test whether your proposed grammar for $\overline{L}$ is non-ambiguous.

Exercise 2.10 (Sufficient conditions for unambiguity)**.** Consider the following conditions for a context-free grammar:

(1) Every production has at most one non-terminal symbol on its right-hand side.
(2) The languages generated by two different productions of the same non-terminal symbol are always disjoint.

Show that any context-free grammar satisfying the above conditions must be unambiguous.

Exercise 2.11 (On regular grammars)**.** A context-free grammar $G = (V, \Sigma, P, S)$ is *right linear* if all its productions are of the form $A \to wB$ or $A \to w$, where $A, B \in V$ and $w \in \Sigma^*$. Analogously, when all the productions of G are of the form $A \to Bw$ or $A \to w$, we say that G is *left linear*. A grammar which is either *right linear* or *left linear* is called *regular*. If the grammar G contains productions of the form $A \to wB$, $A \to Bw$, or $A \to w$, we say that G is *linear*.

The goal of this exercise is to show that regular languages can be generated by unambiguous grammars.

(a) *Normal form.* Prove that for every right-linear grammar $G = (V, \Sigma, P, S)$ there exists an equivalent grammar whose productions are all of the form $A \to aB$ or $A \to \lambda$, where $A, B \in V$ and $a \in \Sigma$. Observe that a symmetric transformation can be applied to left-linear grammars.

(b) *Linear grammars.* Construct a linear grammar generating a non-regular language.

(c) *Regular grammars.* Prove that a language is regular if and only if it is generated by a regular grammar.

> **Tip**
>
> Show this only for left-linear o right-linear grammars (it can be done independently for any of them and they are symmetric). Use the normal form above.

(d) *Unambiguity of regular languages.* Show that the regular languages can be generated by unambiguous grammars.

> **Tip**
>
> Use the construction from the previous item to transform a DFA into a regular grammar and show that the obtained grammar is unambiguous.