Advanced Automata Theory
Exercise Sheet 12

Emanuele D'Osualdo
Sebastian Schweizer

TU Kaiserslautern

Summer term 2016

Out: July 6 | Due: July 11, 12:00

Exercise 1: Tree Language & Word Language

Let $G = (N, T, \Pi, S)$ be a context-free grammar. A derivation tree of $L(G)$ is a tree where leaves are terminals or ϵ, the other nodes are nonterminals, the root is the starting symbol $S \in N$ and the tree structure reflects the production rules. An **indexed derivation tree** is a derivation tree where each nonterminal is indexed with the number of children it has.

We define

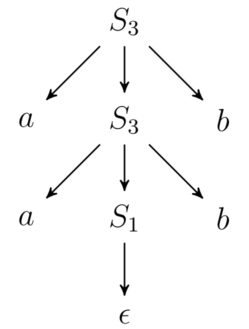
$$\Sigma_G := \{\epsilon/_0\} \cup \{a/_0 \mid a \in T\} \cup \{A_1/_1 \mid A \to \epsilon \in \Pi \text{ with } A \in N\} \cup \\ \{A_k/_k \mid A \to X_0 \dots X_{k-1} \in \Pi \text{ with } A \in N \text{ and } X_i \in N \cup T\}$$

Example:

For $G = (N, T, \Pi, S)$ with
$N = \{S\}, T = \{a, b\}$
$\Pi = \{$
$S \to aSb$
$S \to \epsilon$
$\}$
we have $\Sigma_G =$
$\{\epsilon/_0, a/_0, b/_0, S_1/_1, S_3/_3\}$

Consider the Σ_G-tree

$t =$



t is an indexed derivation tree of $L(G)$.
$\text{yield}(t) = aabb \in L(G)$

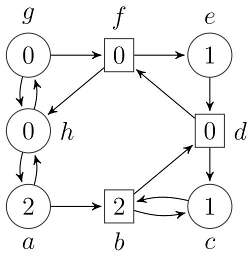
(a) Show that for every context free grammar G the set of indexed derivation trees of $L(G)$ is a regular tree language over Σ_G.

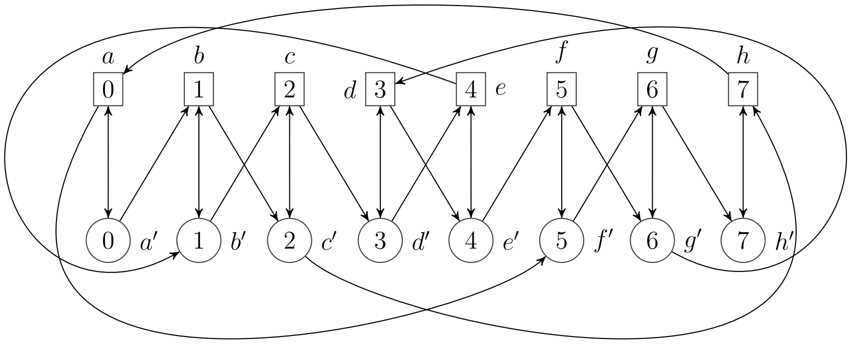
(b) Let L be a regular tree language. Show that $\text{yield}(L) := \{\text{yield}(t) \mid t \in L\}$ is a context-free word language.

Exercise 2: Parity Game Attractors

(a) Compute the attractor of $\{a, b\}$ for player A in the game depicted below on the left.

(b) Compute the attractor of $\{c, c'\}$ for player A in the game depicted on the right.





Reminder: positions of A are ◯s and positions of P are □s.

Exercise 3: Parity Game Strategies

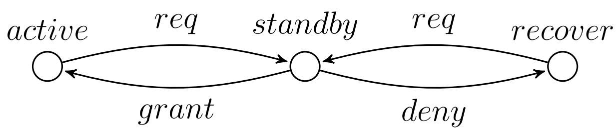
(a) Give a positional strategy of P winning from a in the above right hand side game.

(b) Give a positional strategy of A winning from b in the above right hand side game.

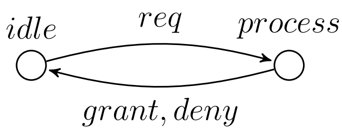
Exercise 4: Construction of Parity Games

Consider the following automata, describing a client server system where the client can request resources and the server may grant or deny them:

Client:



Server:



Transform this system into a parity game that is won by the client if it can reach its *active* state infinitely often.