Theoretical-Computer-Science

This repository contains basic notes about Theoretical-Computer-Science course of Università Della Calabria. You can use this repo for review but not for study as proofs of the theorems are missing

Table of Contents

Languages

What is a language? It is simple. A set of strings. Now the question is obvious. What are strings? Strings are a sequence of simbol of an alphabet. Mathematically we denote with this simbol Σ an alphabet, and with Σ^* all the strings from the alphabet.

Now we can define formally a language given an alphabet Σ like the set $L= (w \mid w \in \Sigma ^*)$

Example: $\Sigma =${ 0,1 }, $L=${11,01,1}

Grammars

Grammars generate languages. Formally a grammar is a quadruple $G = (V,T,P,S)$

- V is the set of non-terminal symbols
- T is the set of terminal symbols
- S is the initial non-terminal symbol
- P is a set of productions

What is a production? Simple a rule that allows you to replace the left side with the right. Formally a production is $\alpha A \beta => \alpha \gamma \beta$ e $A \in V$, e $ `\alpha , \gamma , \beta \in (V \cup T )^*` $, With $(V \cup T)^*$ we identify strings of non-terminal and terminal symbols.

Chomsky hierarchy

We can divide grammars in 4 types base on the language they generate:

- Type 3 grammar: grammar production are of the type, $ `A => a` $ $ `A \in V,\ a \in T` $ $ `A => Ba, A,B \in V, a \in T` $ $ `A => aB, A,B \in V, a \in T` $
- Type 2 grammar: grammar production are of the type, $ `A => \Gamma` $ $ `A \in V, \Gamma \in (V \cup T)^*` $
- Type 0 grammar: grammar production are of the type, $ `\alpha A \beta => \alpha \gamma \beta` $ $ `A \in V` $, $ `\alpha , \gamma , \beta \in (V \cup T )^*` $

Regular languages

Type 3 grammars

Deterministic finite state automata

Non deterministic finite state automata

Regular expression

Proprieties of regular languages

Pumping lemma for regular languages