Notes on 1657+1658

Daniel Maslowski

November 25, 2013

Contents

1 Introduction **3**

2 Formal languages **4**

2.1 Basic terms 4

2.2 The Chomsky Hierarchy 5

2.3 Pumping lemma 6

3 Automata **7**

4 Computability **8**

4.1 Numberings, notations, and representations 8

4.2 Recursive and recursively enumerable 10

4.3 Hierarchies and separation 12

Abstract

The following document mainly reduces the content of the courses 1657 and 1658 ("Grundlagen der Theoretischen Informatik A/B") of the FernUniversität in Hagen to the definitions, theorems and proofs provided by the authors. They may somewhat differ from common definitions found in textbooks. However, when I felt that a definition, theorem or proof did not satisfy me for some reason, I might as well have chosen to modify them, sometimes slightly, sometimes completely, to gain consistency. See this as a general quick reference for anyone. Hence I neither guarantee correctness nor completeness.

1 Introduction

Theoretical Computer Science consists of several larger topics, three of which are mainly formal languages, automata, and computability.

Although the course starts with some automata, I decided to collect the definitions of all the automata mentioned (and some more) in one section. Thus the order of this summary differs from the original order of the course material where automata are defined just where needed. I don't like this hide-and-seek approach though. In addition, I will start introducing formal languages first, like many textbooks do.

2 Formal languages

2.1 Basic terms

Definition 1 (Formal language)
A formal language $\mathcal{L}$ *is a set of strings (words) on some alphabet* Σ*, i.e.* $\mathcal{L} \subseteq \Sigma^*$.

Definition 2 (Alphabet)
An alphabet $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, ...\}$ *is a set of symbols.*
Its cardinality (number of elements) is usually denoted as $|\Sigma|$.

Definition 3 (String)
A string (word) $\mathbf{x}$ *is an ordered list of symbols, often denoted in bold face.*
The length of a string is usually denoted as $|\mathbf{x}|$.
The empty string $\boldsymbol{\varepsilon}$ *is the only string of length* 0.
For an alphabet Σ*,* Σ^i *is the set of all strings of length* i *on* Σ.

Definition 4 (Kleene closure)
The Kleene closure $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$*, provided by the star operator* **, is the set of all strings of any length on an alphabet* Σ*, including the empty string* $\boldsymbol{\varepsilon}$.
The plus operator $^+$*, similar to the star operator, gives the set of all non-empty strings on an alphabet* Σ*, i.e. the Kleene closure excluding the empty string:* $\Sigma^+ = \Sigma^* \backslash \{\boldsymbol{\varepsilon}\}$.

Definition 5 (Concatenation)
The concatenation of two strings $\mathbf{z} = \mathbf{xy}$ *is the ordered list of symbols that results from adding the symbols in* $\mathbf{y}$ *to* $\mathbf{x}$*, keeping their order. It can be arithmetically seen as a multiplication and an addition:* $\mathbf{z} = \mathbf{x} * |\Sigma|^{|\mathbf{y}|} + \mathbf{y}$.
As a shorthand notation, the i-times repeated concatenation of a string $\mathbf{x}$ *with itself is denoted as* $\mathbf{x}^i$.
For convenience: $\mathbf{x}^0 = \boldsymbol{\varepsilon}$*,* $\mathbf{x}^1 = \mathbf{x}$.
The concatenation of two languages $\mathcal{L}_{12} = \mathcal{L}_1\mathcal{L}_2$ *is the set of all strings that result from concatenating all strings from* $\mathcal{L}_1$ *with all strings from* $\mathcal{L}_2$.
As a shorthand notation, similar to the cartesian product, the i-times repeated concatenation of a language $\mathcal{L}$ *with itself is denoted as* $\mathcal{L}^i$.
For convenience: $\mathcal{L}^0 = \{\boldsymbol{\varepsilon}\}$*,* $\mathcal{L}^1 = \mathcal{L}$*,* $(\forall i \geq 1)\varnothing^i = \varnothing$.

2.2 The Chomsky Hierarchy

Definition 6 (Formal grammar)
A formal grammar is a 4-tuple $(\mathrm{N}, \Sigma, \mathrm{P}, \mathrm{s})$ *consisting of an alphabet of non-terminal symbols* N*, an alphabet of terminal symbols* Σ *s.t.* $\mathrm{N} \cap \Sigma = \varnothing$*, a set of production rules* P *and a start symbol* $\mathrm{s} \in \mathrm{N}$*.*

Definition 7 (Regular language)
A regular language is a language produced by a regular grammar.

2.3 Pumping lemma

Theorem 1 (Pumping lemma for regular languages)
$\mathcal{L}$ *regular* $\Rightarrow \exists p \in \mathbb{N}$ *s.t.* $(\forall \mathrm{z} \in \mathcal{L}, |\mathrm{z}| \geq p)\ \exists \mathrm{uvw} = \mathrm{z}$ *s.t.*

1. $|\mathrm{uv}| \leq p$
2. $|\mathrm{v}| \geq 1)$
3. $(\forall i \in \mathbb{N})\ \mathrm{uv^iw} \in \mathcal{L}$

Theorem 2 (Pumping lemma for context-free languages)
$\mathcal{L}$ *context-free* $\Rightarrow \exists p \in \mathbb{N}$ *s.t.* $(\forall \mathrm{z} \in \mathcal{L}, |\mathrm{z}| \geq p)\ \exists \mathrm{luvwr} = \mathrm{z}$ *s.t.*

1. $|\mathrm{uvw}| \leq \mathrm{p}$
2. $|\mathrm{uw}| \geq 1$
3. $(\forall i \in \mathbb{N})\ \mathrm{lu^ivw^ir} \in \mathcal{L}$

3 Automata

Example automaton

$$\begin{tikzcd}
 & & q_1 \arrow[loop above, "0"] \arrow[dr, "1"] & \\
\text{start} \arrow[r] & q_0 \arrow[ur, "0"] \arrow[dr, "1"'] & & q_3 \\
 & & q_2 \arrow[loop below, "1"] \arrow[ur, "0"'] &
\end{tikzcd}$$

4 Computability

This section lists the definitions from the chapters on computability.

There are three very similar main sets that we look at throughout the course: The set of positive integers including 0 ($\mathbb{N}$), the set of all words on some alphabet Σ including the empty word ε (Σ^*) as mentioned in the first chapter and the set of all words of infinite length on some alphabet Σ (Σ^ω). For the reason of this diversity, many of the following definitions and theorems are provided differently for the respective sets.

4.1 Numberings, notations, and representations

Definition 8 (Numbering)
A numbering is a surjective function $\nu :\subseteq \mathbb{N} \to M$.

Definition 9 ($(\nu_1, ..., \nu_k, \nu) - \textit{computable}$ for numberings)
A function $f :\subseteq M_1 \times \cdots \times M_k \to M$ is called $(\nu_1, ..., \nu_k, \nu) - computable$, iff $\exists$ computable $g : \mathbb{N}^k \to \mathbb{N}$ s.t. $f(\nu_1(i_1), ..., \nu_k(i_k)) = \nu \circ g(i_1, ..., i_k)$.

$$\begin{tikzcd}
\mathbb{N}^k \arrow[r, "g"] \arrow[d, "\nu_1, ..., \nu_k"'] & \mathbb{N} \arrow[d, "\nu"] \\
M_1 \times \cdots \times M_k \arrow[r, "f"'] & M
\end{tikzcd}$$

Definition 10 (Notation)
A notation is a surjective function $\nu :\subseteq \Sigma^* \to M$.

Definition 11 ($(\nu_1, ..., \nu_k, \nu) - computable$ for notations)
A function $f :\subseteq M_1 \times \cdots \times M_k \to M$ *is called* $(\nu_1, ..., \nu_k, \nu) - computable$*, iff* $\exists$ *computable* $g : \Sigma^{*k} \to \Sigma^*$ *s.t.* $f(\nu_1(i_1), ..., \nu_k(i_k)) = \nu \circ g(i_1, ..., i_k)$.

$$\begin{tikzcd}
\Sigma^{*k} \arrow[r, "g"] \arrow[d, "\nu_1, ..., \nu_k"'] & \Sigma^* \arrow[d, "\nu"] \\
M_1 \times \cdots \times M_k \arrow[r, "f"'] & M
\end{tikzcd}$$

Definition 12 (Representation)
A representation is a surjective function $\delta :\subseteq \Sigma^\omega \to M$.

Definition 13 ($(\delta_1, ..., \delta_k, \delta) - computable$ for representations)
A function $f :\subseteq M_1 \times \cdots \times M_k \to M$ *is called* $(\delta_1, ..., \delta_k, \delta) - computable$*, iff* $\exists$ *computable* $g : \Sigma^{\omega k} \to \Sigma^\omega$ *s.t.* $f(\delta_1(i_1), ..., \delta_k(i_k)) = \delta \circ g(i_1, ..., i_k)$.

$$\begin{tikzcd}
\Sigma^{\omega k} \arrow[r, "g"] \arrow[d, "\delta_1, ..., \delta_k"'] & \Sigma^\omega \arrow[d, "\delta"] \\
M_1 \times \cdots \times M_k \arrow[r, "f"'] & M
\end{tikzcd}$$

4.2 Recursive and recursively enumerable

There are two functions which classify automata into the two categories of deciders and recognizers (acceptors), which are defined as follows:

Definition 14 (characteristic function)

$$cf_A(x) := \begin{cases} 1, & \textit{if } x \in A \\ 0, & \textit{otherwise} \end{cases}$$

Definition 15

$$df_A(x) := \begin{cases} 1, & \textit{if } x \in A \\ div, & \textit{otherwise} \end{cases}$$

Note that the authors of the course use "div" for diverging in the sense of non-terminating. Others may call the value "undefined".

A set $A \subseteq \mathbb{N}^k$ is called *recursive* (decidable), iff its characteristic function cf_A is computable, i.e., there exists a decider for A.

Definition 16 $A \subseteq \mathbb{N}^k$ *recursive*
$\Leftrightarrow cf_A : \mathbb{N}^k \to \mathbb{N}$ *computable*

Theorem 3 $A \subseteq \mathbb{N}^k$ *recursive*
$\Leftrightarrow \exists$ *computable* $f \in R^{(k)}$ *s.t.* $A = f^{-1}\{0\}$
I.e., all elements of A are mapped to a fixed value (not necessarily 0) by a recursive function f.

Definition 17 $A \subseteq (\Sigma^*)^k$ *recursive*
$\Leftrightarrow A = f^{-1}\{\epsilon\}$ *for some total computable* $f : (\Sigma^*)^k \to \Sigma^*$

A set $A \subseteq \mathbb{N}^k$ is called *recursively enumerable* (semi-decidable), in short *r.e.*, iff A is the domain of some computable function, i.e. there exists an acceptor for A.

Definition 18 $A \subseteq \mathbb{N}^k$ *recursively enumerable*
$\Leftrightarrow \exists$ *computable* $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ *s.t.* $A = Dom(f)$

Theorem 4 $A \subseteq \mathbb{N}^k$ *r.e.*
$\Leftrightarrow df_A : \mathbb{N}^k \to \mathbb{N}$ *computable*

Theorem 5 $A \subseteq \mathbb{N}^k$ *r.e.*
$\Leftrightarrow \pi^{(k)}[A]$ *r.e.*

Theorem 6 $A \subseteq \mathbb{N}^k$ *recursive*
$\Leftrightarrow A$ *r.e.* $\wedge\ \mathbb{N}^k \backslash A$ *r.e.*

Theorem 7 (Projection Theorem) $A \subseteq \mathbb{N}^k$ *r.e.*
$\Leftrightarrow \exists$ *recursive* $B \subseteq \mathbb{N}^{k+1}$ *s.t.* $A = \{x \in \mathbb{N}^k \mid (\exists t)\ (x, t) \in B\}$

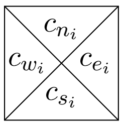
Definition 19 $A \subseteq (\Sigma^*)^k$ *r.e.*
$\Leftrightarrow A = Dom(f)$ *for some computable* $f :\subseteq (\Sigma^*)^k \to \Sigma^*$

4.3 Hierarchies and separation

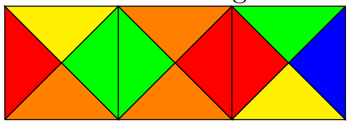
Theorem 8 (Time Hierarchy Theorem)
$\mathbf{DTIME}\left(o\left(\frac{f(n)}{\log f(n)}\right)\right) \subsetneq \mathbf{DTIME}(f(n))$

Theorem 9 (Cook-Levin[1])
$\text{3SAT} \in \mathbf{NP} \wedge (\forall\, \text{L} \in \mathbf{NP})\ \text{L} \leq_p \text{3SAT}$

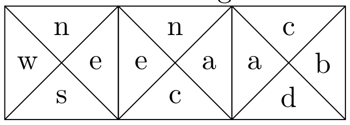
A Wang tile



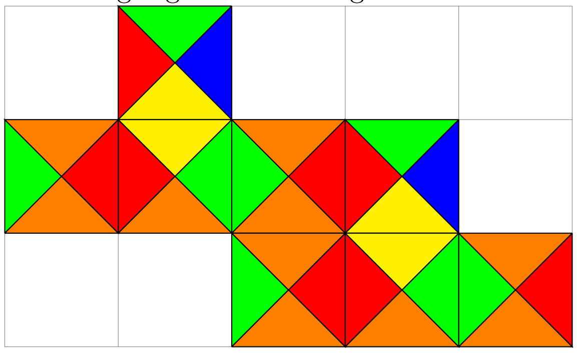
Coloured Wang tiles



Labeled Wang tiles



Filling a grid with Wang tiles



References

[1] Stephen A. Cook, *The complexity of theorem-proving procedures.* Proceedings of the 3rd Annual ACM Symposuium on the Theory of Computing (STOC'71). ACM, New York 1971, pp 151–158.