Theoretical Computerscience – Summary

WS 24/25

Contents

0 Words

A word w (also called String) has length l and consists of symbols $\sigma \in$ ˚.

The empty word ε has length 0.

The concatenation of two words w and x is denoted as wx.

w^i denotes the i-fold concatenation of w with itself. Therefore $w^3 = www$.

$\varepsilon w = w\varepsilon = w$.

0.1 Subwords

A word w is a subword of a word x if there exist words u and v such that $x = uwv$.

The empty word is a subword of every word.

The word w is a subword of itself.

0.2 Prefixes and Suffixes

A word w is a prefix of a word x if there exists a word v such that $x = wv$.

A word w is a suffix of a word x if there exists a word u such that $x = uw$.

The empty word is a prefix and suffix of every word.

The word w is a prefix and suffix of itself.

1 Finite Automatons

TODO: Add chapter contents

2 Regular Languages

TODO: Add chapter contents

3 Regular Expressions

A regular expression always describes a regular language. If we can build a regular expression E, then L(E) ∈ REG.

3.1 Languages of regular expressions

If $E = \emptyset$ then $L(E) = \emptyset$
If $E = \varepsilon$ then $L(E) = \{\varepsilon\}$
If $E = \sigma$ then $L(E) = \{\sigma\}$
If $E = (E_1 + E_2)$ then $L(E) = L(E_1) \cup L(E_2)$
If $E = (E_1 E_2)$ then $L(E) = L(E_1)L(E_2)$
If $E = (E_1)^*$ then $L(E) = L(E_1)^*$

3.2 Identities

1. $E + F = F + E$
2. $(E + F) + G = E + (F + G)$
3. $(EF)G = E(FG)$
4. $\emptyset + E = E + \emptyset = E$
5. $\varepsilon E = E\varepsilon = E$
6. $\emptyset E = E\emptyset = \emptyset$
7. $E + E = E$
8. $(E + F)G = (EG) + (FG)$
9. $E(F + G) = (EF) + (EG)$
10. $(E^*)^* = E^*$
11. $\emptyset^* = \varepsilon$
12. $\varepsilon^* = \varepsilon$

3.3 Regular expressions and finite automata

If E is a regular expression, then $L(E) \in$ REG.
For every deterministic finite automaton $M = (Q, \sum, \delta, q_0, Q_{acc})$ there is a regular expression E with $L(M) = L(E)$.

4 The Pumping Lemma

TODO: Add chapter contents

5 Equivalences and Myhill Nerode

5.1 Equivalence Relations

TODO: Add chapter contents

5.2 Myhill Nerode

TODO: Add chapter contents

6 Limits of computability

TODO: Add chapter contents

7 FOR and WHILE programs

7.1 WHILE programming language

Variables: $x_0, x_1, \ldots, x_n$
Constants: 0, 1, 2, ...
Keywords: while, do, od
Other symbols: :=,≠, ;, +, −, [,]
W_0: set of all simple statements

7.2 FOR programming language

Variables: $x_0, x_1, \ldots, x_n$
Constants: 0, 1, 2, ...
Keywords: for, do, od
Other symbols: :=,≠, ;, +, −, [,]

7.3 Semantics

Input is stored in $x_0, \ldots, x_{s-1}$ Output is the content of x_0 after execution of P.
The set $X = \{x_0, x_1, x_2, \ldots\}$ if akk variables is finite.

7.4 FOR/WHILE computable functions

1. A partial function $f : \mathbb{N}^s \rightarrow \mathbb{N}$ is WHILE computable, if there is a WHILE program P such that $f = \phi_P$.
2. f is FOR computable if $f = \phi_P$ for some FOR program P.
3. The set of all WHILE computable functions is denoted by R.
4. The set of all FOR computable functions is denoted by PR.

7.5 Decidable languages

The characteristic function of L is the function $\chi_L : \mathbb{N} \rightarrow \mathbb{N}$ defined by

$$\chi_L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}$$

A language L is decidable if χ_L is computable ($\chi_L \in R$).
The set of all decidable languages is denoted by REC.

8 Syntactic Sugar

8.1 Syntactic Sugar

Program 1: Assignments $x_i := x_j$

```
x_k := 0;
x_i := x_j + x_k;
```

8.2 Arithmetic Operations

Program 2: Multiplication $x_i := x_j \cdot x_k$

```
x_i := 0;
for x_j do
    x_i := x_i + x_k
od
```

We assume that $i \neq j,k$

8.3 Pairing Functions

$\langle x_1, x_2 \rangle = \frac{1}{2} \cdot (x_1 + x_2) \cdot (x_1 + x_2 + 1) + x_2$

$\pi_i(\langle x_1, x_2 \rangle) = x_i$ for $i = 1,2$ and $x_1, x_2 \in \mathbb{N}$

8.4 Stacks

$S = \langle n, Y\rangle$

$n \in \mathbb{N}$ is the number of elements in the stack and $Y \in \mathbb{N}^n$ is the stack itself

The empty stack is $\langle 0, 0\rangle$

If $a_1, \ldots, a_n$ is stored in S, then $Y = \langle a_n, \langle a_{n-1}, \ldots \langle a_1, 0\rangle \ldots\rangle\rangle$

Program 3: *push* (S, x)

```
Y := ⟨x, π₂(S)⟩;
S := ⟨π₁(S) + 1, Y⟩;
```

Program 4: *pop* (S)

```
n := π₁(S);
if n ≠ 0 then
    S := ⟨n − 1, π₂(π₂(S))⟩;
fi
```

Program 5: $x = top\ (S)$

```
x := π₁(π₂(S));
```

Program 6: $b = isempty\ (S)$

```
n := π₁(S);
if n ≥ 0 then
    b := 0;
else
    b := 1;
fi
```

8.5 Arrays

TODO: Add array implementation

9 Gödel numberings

Bijection $gd : W \rightarrow \mathbb{N}$
WHILE program U, that outputs $\phi_P(x)$ in input i,$x \in \mathbb{N}$ where $P = \text{göd}^{-1}(i)$.

9.1 Encoding

Simple Statements:

1. $x_i := x_j + x_k$ is encoded by $\langle 0, \langle i, \langle j, k\rangle\rangle\rangle_5$
2. $x_i := x_j - x_k$ is encoded by $\langle 1, \langle i, \langle j, k\rangle\rangle\rangle_5$
3. $x_i := c$ is encoded by $\langle 2, \langle i, c\rangle\rangle_5$

While loop and concatenation:

1. If $P =$ while $x_i \neq 0$ do P_1 od is encoded by $\langle 3, \langle i, \text{göd}(P_1)\rangle\rangle_5$
2. If $P = [P_1; P_2]$ then $\text{göd}(P) = \langle 4, \langle \text{göd}(P_1), \text{göd}(P_2)\rangle\rangle_5$

10 Diagonalisation

TODO: Add chapter contents

11 Universal WHILE Program

TODO: Add chapter contents

12 Halting Problem

TODO: Add chapter contents

13 Reductions

13.1 Known problems

13.1.1 Verification Problem V

The verification problem has two gödel number inputs namely i, j and checks if the two programs corresponding to these gödel numbers output the same for every possible input.

13.1.2 Special Verification Problem V_0

The special verification problem has a gödel number i as input and checks if the program corresponding to that gödel number will output 0 for every possible input.

13.1.3 Program termination T

The program termination problem has a gödel number i as input and checks if the program corresponding to that gödel number will terminate for every possible input.
TODO: Add formulas for V, V0 and T

13.2 Many-one Reduction

A WHILE computable total function $f : \mathbb{N} \to \mathbb{N}$ is called many-one reduction from L to L' if

- $L, L' \subseteq \mathbb{N}$
- $\forall x \in \mathbb{N} : x \in L \iff f(x) \in L'$

If such an f exists, then L is many-one reducible to L', therefore we can write $L \leq L'$

13.3 Known Reductions

$H_0 \leq V_0$
$V_0 \leq V$
$V_0 \leq T$
$\overline{H_0} \leq V_0$

14 More on Reductions

TODO: Add chapter contents

15 Rice's Theorem

TODO: Add chapter contents

16 Turing Machines

TODO: Add chapter contents

17 Examples, tricks and syntactic sugar for Turing Machines

TODO: Add chapter contents

18 Church-Turing Thesis

TODO: Add chapter contents

19 Common Proof Techniques

19.1 Pumping Lemma

The pumping lemma can only be used to show that a language is not REG. We take a word $w \in L$ and split it into multiple sub words u,v,w where $|v| \geq n$ with $n \geq 0$. Now there are words x,y,z with $v = xyz$ and $|y| \geq 0$ such that $uxy^izw \in L$ for all $i \in \mathbb{N}$. Afterwards we try to find an i such that the pumped word is no longer $\in L$ and thus we proved that the language is not regular.

19.1.1 Example

Exercise:
Show that the following language is not regular.
Let $A = \{1^{(3n)} \mid n \in \mathbb{N}\}$

Solution:
Let $n \geq 0$ be given.
We choose $u = 1^n$, $v = 1^n$, $w = 1^n$ such that $uvw = 1^{3n}$ and $|v| = n$.
Let x,y,z be given as $x = 1^r$, $y = 1^s$, $z = 1^t$ with $xyz = v$ and $s \geq 0$ since $y \neq \varepsilon$ and $r + s + t = n$.
$uxy^izw = 1^n 1^r 1^{s \cdot i} 1^t 1^n = 1^n 1^{r+s+t} 1^{s \cdot (i-1)} 1^n$
We choose $i = 0$, therefore $1^n 1^{r+s+t} 1^{s \cdot (i-1)} 1^n = 1^n 1^{n-s} 1^n \notin A$ since it is not of the form 1^{3m} anymore for any $m \in \mathbb{N}$.
Therefore we cannot pump language A and thus it is not regular.

19.2 Myhill Nerode

19.2.1 Example

TODO: Add example for Myhill Nerode

19.3 Reductions

19.3.1 Example

Exercise:
Consider the following language $L = \{i \in \mathbb{N} \mid \text{göd}^{-1}(i) \text{ outputs 42 on input 1337}\}$
Prove $L \notin \text{co}-\text{RE}$

General Ideas:
$\overline{L}$ is the set of all programs that either diverge or output something else than 42 on input 1337.
L is the set of all programs that output 42 on input 1337 which is equal to running a sub program that halts on every input and outputting 42 afterwards. Thus we could try to reduce the special halting problem H_0 to L.

Solution:
$\forall L : L \in \text{REC} \iff L \in \text{RE} \land \overline{L} \in \text{RE}$
$L \notin \text{co}-\text{RE} \iff \overline{L} \in \text{RE}$

$H_0 \notin \text{REC}$ but $H_0 \in \text{RE}$, therefore $\overline{H_0} \notin \text{RE}$ thus reducing $\overline{H_0} \leq \overline{L}$ suffices to show that $\overline{L} \notin \text{RE}$ but this is the same as $H_0 \leq L$ by equivalence.

The reduction is given by $f(i) := \text{göd}(P)$
We create the WHILE program P, that has input m. However we ignore the input m and simulate i on the input i, then return 42. Thus f is obviously WHILE computable.

We consider the following two cases:

- Let $i \in H_0$, then i halts on input i by definition. Program P halts on arbitrary inputs m and outputs 42. Therefore P also outputs 42 on our input 1337.
  Thus $\text{göd}(P) \in L$ holds.
- Let $i \notin H_0$, then i does not halt on input i by definition. Therefore Program P diverges on all inputs, which is also the case for input 1337.
  Therefore P diverges on our input 1337.
  Thus $\text{göd}(P) \notin L$ holds.

Therefore we have successfully shown that $H_0 \leq L$ is a valid reduction.

20 Useful Proofs

20.1 Regular Languages

20.1.1 Finite Set

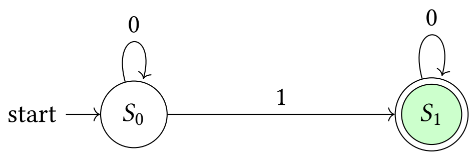
Exercise:
Show that the following language is regular over the alphabet $\{0,1\}$.
$L = \{x \mid x \text{ is prime and } x < 1'000'000'000\}$

Solution:
Since there are only finitely many prime numbers between 0 and 1′000′000′000, the set of the words that are accepted by L is finite and thus the language is regular.

20.1.2 Finite Automaton

Exercise:
Show that the following language is regular over the alphabet $\{0,1\}$.
$L = \{0^n 1 0^m \mid n,m \in \mathbb{N}\}$

Solution:



Since we can describe the language L by the finite automaton given above, the language is regular.

20.1.3 Regular Expression

Exercise:
Show that the following language is regular over the alphabet $\{0,1\}$.
$L = \{0^n 1^m \mid n,m \in \mathbb{N}\}$

Solution:
Let $E = 0^* 1^*$ be the regular expression describing the language L.
Since we can describe the language L by the regular expression given above, the language is regular.

20.1.4 Closure Properties

20.2 Non-Regular Languages

20.2.1 Pumping Lemma

Exercise:
Show that the following language is not regular over the alphabet {0,1}.
$L = \{0^n 1^n \mid n \in \mathbb{N}\}$

Solution:
Let $n \geq 0$ be given.
We choose $u = 1^n$, $v = 1^n$, $w = 1^n$ such that $uvw = 1^{3n}$ and $|v| = n$.
Let x,y,z be given as $x = 1^r$, $y = 1^s$, $z = 1^t$ with $xyz = v$ and $s \geq 0$ since $y \neq \varepsilon$ and $r + s + t = n$.
$uxy^i zw = 1^n 1^r 1^{s \cdot i} 1^t 1^n = 1^n 1^{r+s+t} 1^{s \cdot (i-1)} 1^n$
We choose $i = 0$, therefore $1^n 1^{r+s+t} 1^{s \cdot (i-1)} 1^n = 1^n 1^{n-s} 1^n \notin A$ since it is not of the form 1^{3m} anymore for any $m \in \mathbb{N}$.
Therefore we cannot pump language A and thus it is not regular.

20.2.2 Myhill Nerode

Exercise:
Show that the following language is not regular over the alphabet {0,1}.
$L = \{0^n 1^n \mid n \in \mathbb{N}\}$

Solution:
TODO: Add solution for Myhill Nerode