ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Prof. Dr. Andreas Podelski
Matthias Heizmann
Christian Schilling

July 8th-9th, 2014

6. Lecture
Computer Science Theory

Chapter IV – The notion of algorithm: What can be computed using machines? (pp. 71-96)

§1 Turing machines (pp. 61-88)

Variations of Turing machines There are several notions extensions to Turing machines (TMs) which are sometimes more useful. But it turns out that they do not add more power to the machines – TMs are already most powerful. This is the case because we abstract away from space and time, i.e., we are only interested in a result after a finite but unbounded amount of time.

(a) k-tape TM: We have a fixed (k) number of auxiliary tapes.

(b) TM with k heads

(c) two-dimensional tape (possibly with k heads)

(d) one-sided infinite tape

(e) TM with final states

Note that in this case we lose the ability to produce an output, apart from "true" and "false". However, that is usually enough, which can be seen in the following way:

Let $h : \Sigma^* \rightharpoonup \Gamma^*$ be Turing-computable. The *graph of h* is defined as

$$\{w\#h(w) \mid w \in \Sigma^* \text{ and } h(w) \text{ is defined}\}.$$

Instead of asking a TM for its output given an input w, we can also ask the yes/no question "is $w\#v$ an element of the graph?".

Connection to Turing decidability
A language accepted by a TM with final states is called *Turing-acceptable.*

Theorem 1.13 Let $L \subseteq \Sigma^*$ and $\overline{L} = \Sigma^* \setminus L$. Then L and $\overline{L}$ are Turing-acceptable iff L is Turing-decidable.

(f) nondeterministic TM

§2 Grammars (pp. 89-94)

We skip this part in the interest of time.

Summary: There are equivalent (Chomsky) grammar classes for

- Turing machines (CH0, in the sense of accepting/deciding)
- context-sensitive languages (CH1, not covered here)
- context-free languages (CH2, already covered)
- regular languages (CH3)

Summary of the chapter (pp. 95-96)

Definition 2.13 A function $f : \Sigma^* \rightharpoonup \Gamma^*$ is called *recursive* if f is Turing-computable.
A language $L \subseteq \Sigma^*$ is called *recursive* if $\chi_L : A^* \to \{0,1\}$ is recursive, i.e., if L is Turing-decidable.

Church-Turing thesis (1936) The functions which are intuitively computable using algorithms are exactly the recursive functions, i.e. "recursion" is the mathematic formalization of the notion "algorithm".

People have tried to find other concepts. But they all turned out to be equivalent to TMs. Some of them are:

- μ-recursive functions/μ-calculus
- λ-calculus
- two-counter machines/register machines with at least two registers
- satisfiability of first-order formulae

Chapter V – Non-computable functions – undecidable problems (pp. 97-122)

We prove that there are (total) functions $f : \mathbb{N} \to \mathbb{N}$ which are not Turing-computable. By the Church-Turing-thesis this means there are functions which computers cannot compute.
We do this in an abstract way first, then show a concrete example.

§1 Existence of non-computable functions (pp. 97-100)

We use the notion of *(un-)countably infinite.*

(a) The number of Turing-computable functions is at most the number of Turing machines. The latter is countably infinite.

(b) The number of functions $f : \mathbb{N} \to \mathbb{N}$ is uncountably infinite.

Thus there must be functions (even infinitely many) which are not Turing-computable.

§2 Concrete undecidable problem: halting for Turing machines (pp. 101-107)

We want to show that the elementship-problem is undecidable for CH0-languages. For this we consider a very simple formalism:
Let $B = \{0, 1\}$ be our alphabet. The *halting problem* for TMs is:

Given: an encoding of (τ, w) over B, where τ is a TM and w a word
Question: Does τ applied to w halt, i.e., is $h_\tau(w)$ defined?

We can encode a TM by first translating to an intermediate *standard encoding* and then to the binary alphabet (see lecture script).

Definition 2.2 (Special halting problem)

$$K = \{bw_\tau \mid \tau \text{ applied to } bw_\tau \text{ halts}\}$$

Theorem 2.3 K is undecidable. ✎(1)