Theory of Computer Science

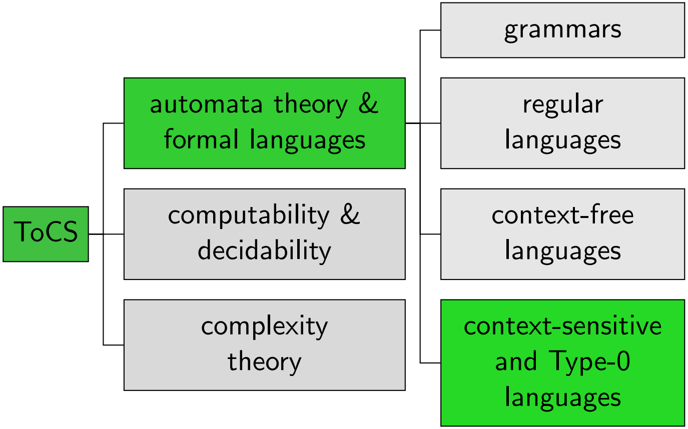
B13. Type-1 and Type-0 Languages: Closure & Decidability

Gabriele Röger

University of Basel

April 10, 2024

Content of the Course


ToCS
automata theory & formal languages
computability & decidability
complexity theory
grammars
regular languages
context-free languages
context-sensitive and Type-0 languages

Turing Machines vs. Grammars

Turing Machines

We have seen several variants of Turing machines:

- Deterministic TM with head movements left or right
- Deterministic TM with head movements left, right or neutral
- Multitape Turing machines
- Nondeterministic Turing machines

All variants recognize the same languages.

Turing Machines

We have seen several variants of Turing machines:

- Deterministic TM with head movements left or right
- Deterministic TM with head movements left, right or neutral
- Multitape Turing machines
- Nondeterministic Turing machines

All variants recognize the same languages.

We mentioned earlier that we can relate Turing machines to the Type-1 and Type-0 languages.

Reminder: Context-sensitive Grammar

Type-1 languages are also called context-sensitive languages.

Definition (Context-sensitive Grammar)

A context-sensitive grammar is a 4-tuple $\langle V, \Sigma, R, S \rangle$ with

- V finite set of variables (nonterminal symbols)
- Σ finite alphabet of terminal symbols with $V \cap \Sigma = \emptyset$
- $R \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ finite set of rules, where all rules are of the form $\alpha B \gamma \rightarrow \alpha \beta \gamma$ with $B \in V$ and $\alpha, \gamma \in (V \cup \Sigma)^*$ and $\beta \in (V \cup \Sigma)^+$. Exception: $S \rightarrow \varepsilon$ is allowed if S never occurs on the right-hand side of a rule.
- $S \in V$ start variable.

One Automata Model for Two Grammar Types?

Don’t we need different automata models for context-sensitive and Type-0 languages?

Picture courtesy of stockimages / FreeDigitalPhotos.net

Linear Bounded Automata: Idea

- Linear bounded automata are NTMs that may only use the part of the tape occupied by the input word.
- one way of formalizing this: NTMs where blank symbol may never be replaced by a different symbol

Linear Bounded Turing Machines: Definition

Definition (Linear Bounded Automata)

An NTM $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$
is called a linear bounded automaton (LBA)
if for all $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and all transition rules
$\langle q', c, y \rangle \in \delta(q, \Box)$ we have $c = \Box$.

LBAs Recognize Type-1 Languages

Theorem

The languages that can be recognized by linear bounded automata are exactly the context-sensitive (type-1) languages.

Without proof.

LBAs Recognize Type-1 Languages

Theorem

The languages that can be recognized by linear bounded automata are exactly the context-sensitive (type-1) languages.

Without proof.

proof sketch for grammar $\Rightarrow$ NTM direction:

- computation of the NTM follows the production of the word in the grammar in opposite order
- accept when only the start symbol (and blanks) are left on the tape
- because the language is context-sensitive, we never need additional space on the tape (empty word needs special treatment)

NTMs Recognize Type-0 Languages

Theorem

The languages that can be recognized by nondeterministic Turing machines are exactly the type-0 languages.

Without proof.

NTMs Recognize Type-0 Languages

Theorem

The languages that can be recognized by nondeterministic Turing machines are exactly the type-0 languages.

Without proof.

proof sketch for grammar $\Rightarrow$ NTM direction:

- analogous to previous proof
- for grammar rules $w_1 \to w_2$ with $|w_1| > |w_2|$,
  we must "insert" symbols into the existing tape content;
  this is a bit tedious, but not very difficult

What about the Deterministic Variants?

We know that DTMs and NTMs recognize the same languages. Hence:

Corollary

The Turing-recognizable languages are exactly the Type-0 languages.

What about the Deterministic Variants?

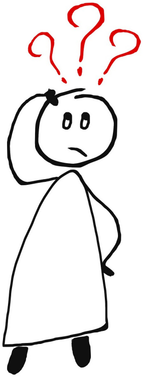
We know that DTMs and NTMs recognize the same languages. Hence:

Corollary

The Turing-recognizable languages are exactly the Type-0 languages.

Note: It is an open problem whether deterministic LBAs can recognize exactly the type-1 languages.

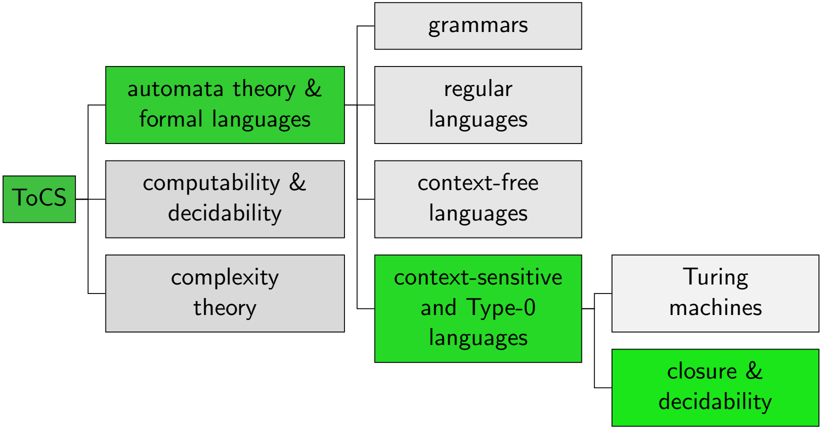
Questions



Questions?

Closure Properties and Decidability

Content of the Course



Closure Properties

	Intersection	Union	Complement	Concatenation	Star
Type 3	Yes	Yes	Yes	Yes	Yes
Type 2	No	Yes	No	Yes	Yes
Type 1	Yes$^{(2)}$	Yes$^{(1)}$	Yes$^{(2)}$	Yes$^{(1)}$	Yes$^{(1)}$
Type 0	Yes$^{(2)}$	Yes$^{(1)}$	No$^{(3)}$	Yes$^{(1)}$	Yes$^{(1)}$

Proofs?
(1) proof via grammars, similar to context-free cases
(2) without proof
(3) proof in later chapters (part C)

Decidability

	Word problem	Emptiness problem	Equivalence problem	Intersection problem
Type 3	Yes	Yes	Yes	Yes
Type 2	Yes	Yes	No	No
Type 1	Yes(1)	No(3)	No(2)	No(2)
Type 0	No(4)	No(4)	No(2)	No(2)

Proofs?

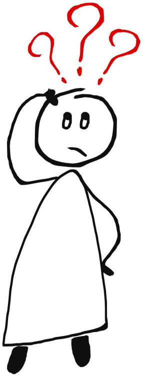
(1) same argument we used for context-free languages

(2) because already undecidable for context-free languages

(3) without proof

(4) proofs in later chapters (part C)

Questions



Questions?

Summary

Summary

- Turing machines recognize exactly the type-0 languages.
- Linear bounded automata recognize exactly the context-sensitive languages.
- The context-sensitive and type-0 languages are closed under almost all usual operations.
  - exception: type-0 not closed under complement
- For context-sensitive and type-0 languages almost no problem is decidable.
  - exception: word problem for context-sensitive lang. decidable

What's Next?

contents of this course:

A. background ✓
  - ▷ mathematical foundations and proof techniques

B. automata theory and formal languages ✓
  - ▷ What is a computation?

C. Turing computability
  - ▷ What can be computed at all?

D. complexity theory
  - ▷ What can be computed efficiently?

E. more computability theory
  - ▷ Other models of computability