Advanced topics in theoretical computer science
Summary of topics

July 18, 2021

1. Turing machines basic notions.

2. Register machines
   - Register Machines: generalities; definition; state of a register machine; initial state/input; output; the semantics of a register machine; computed function; computable function.
   - LOOP programs: syntax, semantics.
   - Every LOOP program terminates for every input. (Additional instructions will be provided in the exam if needed.)
   - WHILE programs: syntax, semantics.
   - LOOP ⊆ WHILE;
   - Non-termination; WHILE computable functions might not be total.
   - GOTO programs: syntax, semantics.
   - WHILE = GOTO, WHILE\textsuperscript{part} = GOTO\textsuperscript{part} (transformations!).
   - Every WHILE computable function can be computed by a WHILE-IF program with one while loop only (construction!).
   - GOTO ⊆ TM, GOTO\textsuperscript{part} ⊆ TM\textsuperscript{part}.
   - LOOP ≠ TM (idea of the proof)

3. Recursive functions
   - Primitive recursive functions: definition, notation. Examples of primitive recursive functions. The set of primitive recursive functions is closed under reordering, omitting, repeating arguments. Additional arguments.
     - Case distinction (primitive recursive)
     - Sums and products (primitive recursive)
     - The bounded μ operator (definitions using the bounded μ-operator - primitive recursive under certain conditions)
Other examples of primitive recursive functions: divides, prime, $n$-th prime number, ...

• G"odelisation is primitive recursive. Simultaneous recursion.
• $P = \text{LOOP}$ (constructions!)

• $\mu$-recursive functions: definition, notation

• $F_\mu \subseteq \text{WHILE}$, $F^\text{part}_\mu \subseteq \text{WHILE}^\text{part}$

• There exist $\mu$-recursive functions which are not primitive recursive (e.g. the Ackermann function, idea).

• G"odelisation of Turing machines/configurations of Turing machines: only idea

• Simulation lemma (only statement)

• Every TM-computable function is $\mu$-recursive (idea of proof)

• Corollary: Kleene normal form (only statement)

• $F_\mu = \text{TM} = \text{WHILE}$

4. The Church-Turing Thesis

5. Computability, (Un-)decidability

• TM, acceptance, decidability

• Undecidability of the halting problem

• Undecidability: proof via reduction

  • Examples: Undecidability of $H_0$

• The theorem of Rice (formal version, as given on pages 27-28 of the file “computability1.pdf” - Slides 22.06.2021).

• Undecidability of the set of theorem in first-order logic (only idea of proof)

• The Post Correspondence Problem (statement; STS, PNS)

  • Undecidability of $\text{Trans}_G$ for some STS $G$.

  • Post correspondence system associated $P_{G,w,w'}$ associated with a STS $G$ and pair of words $w, w'$. Construction of a solution from a computation $w' \Rightarrow^* G w''$.

  • The Post Correspondence Problem is undecidable (only idea of proof).

  • Consequences: it is undecidable whether:

    • a c.f. grammar is ambiguous;
    • the intersection of two DCFL/non-ambiguous c.f./c.f. languages is empty;
    • for a c.f. language $L$ over alphabet with $> 1$ letters $L = \Sigma^*$;
    • equality/inclusion of c.f. languages; equality of c.f. and regular language; inclusion of regular language in a c.f. language.
6. Complexity:

- DTIME and NTIME; DSPACE and NSPACE
- if \( f \) is computable then every language in DTIME(\( f(n) \)) is decidable
- if \( f \) is computable then every language in DSPACE(\( f(n) \)) is decidable
- DTIME(\( f(n) \)) \subseteq NTIME(\( f(n) \)); DSPACE(\( f(n) \)) \subseteq NSPACE(\( f(n) \));
- NTIME(\( f(n) \)) \subseteq \text{DTIME}(2^{O(n^h)}) \text{ where } h \in O(f); \text{ NSPACE}(f(n)) \subseteq \text{DSPACE}(f(n)^2)
- Constant factors are usually ignored (series of theorems on this)
- Definitions P, NP, PSPACE
  - NP \subseteq \bigcup_{i \geq 1} \text{DTIME}(2^{O(n^i)})
  - P \subseteq \text{NP} \subseteq \text{PSPACE}
- Polynomial time reducibility
  - Assume \( L_2 \preceq_{\text{pol}} L_1 \). If \( L_1 \in \text{NP} \) then \( L_2 \in \text{NP} \); if \( L_1 \in \text{P} \) then \( L_2 \in \text{P} \).
  - Composition of two polynomial reductions is a polynomial reduction.
- NP-hard, NP-complete languages
- PSPACE-hard, PSPACE-complete languages
- Open Problem: \( P = \text{NP} \)
- P, PSPACE closed under complement
- co-NP, definition
- Open Problem: \( \text{NP} = \text{co-NP} \)
- Open Problems: \( P = \text{PSPACE}; \text{NP} = \text{PSPACE} \)

- How to prove that \( L \) is NP-complete.
- Examples of NP-complete problems (SAT (idea of proof); 3-CNF-SAT (reduction from SAT); Clique and Rucksack (reduction from 3-CNF-SAT)
- Example of a PSPACE complete problem: Quantified Boolean formulae (QBF).