Theory of Computation

CS3330
2020

Chapter 3: Turing Machine

Definitions

1. **Turing machine:** A Turing machine is a 7-tuple $M = (Q, \sum, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.
   (a) Deterministic TM (**DTM**): $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
   (b) Nondeterministic TM (**NTM**): $\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$, any any point in a computation, the machine proceed according to several probabilities.
2. **Halting Configurations:** Accepting and rejecting configurations are also called halting configurations.
3. **Turing-recognizable language:** A language L is Turing-recognizable if there is a Turing machine M that recognizes it. [It is difficult to distinguish a TM machine that fail to accept (reject) from one that merely takes long-time (looping) to halt.]
4. **Decider:** A TM that halts on all inputs is called a decider.
5. **A Language is called Turing-decidable if some TM decides it.**
   (a) Any **regular language** is Turing-decidable.
   (b) Any **context-free language** is Turing-decidable.
   (c) Every decidable language is Turing-recognizable, and Certain Turing-recognizable languages are not decidable.
6. **Algorithm as a Turing Machine:** An algorithm is a decider TM in the standard representation. [The input to a Turing machine is always a string: A string representation of an object O is $\langle O \rangle$.]

Theorems, Corollaries

1. Every NTM has an equivalent DTM vice versa.
2. Every multiape Turing machine has an equivalent single tap Turing machine.
3. A language is Turing-recognizable iff some NTM recognizes it.
4. A language is decidable iff some NTM decides it.

Chapter 4: Decidablility

Definition

1. **Decidable Language:** A language is decidable if there is an *algorithm* (i.e. a Turing decider) to recognize it.
2. $A_{DFA} = \{\langle B, w\rangle \mid B$ is a DFA that acceptes $w\}$
3. $A_{NFA} = \{\langle B, w\rangle \mid B$ is a NFA that acceptes $w\}$
4. $E_{DFA} = \{\langle A\rangle \mid A$ is a DFA and $L(A) = \emptyset\}$
5. $EQ_{DFA} = \{\langle A, B\rangle \mid A, B$ are DFAs, and $L(A) = L(B)\}$.
6. $ALL_{DFA} = \{\langle A\rangle \mid A$ is a DFA that recognize$\sum^*\}$
7. $A_{CFG} = \{\langle G, w\rangle \mid G$ is a CFG that generates a string $w\}$ (Chomsky normal form).
8. $E_{CFG} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \emptyset$
9. **Every context-free language is decidable.**
   (a) Class of CF languages is not closed under intersection.
   (b) Class of CF languages is not closed under complementation.
10. $EQ_{CFG} = \{\langle G \wedge H\rangle \mid G, H$ are CFG, and $L(G) = L(H)\}$
11. Four examples are porovied in $p30 - p38$ of the slides *Decidability*.
12. A set is countable if either it is finite or it has the same size as $\mathcal{N}$.
    (a) Countable set: $\mathcal{N}$, Rational Number $\mathcal{Q}$, $\sum^*$, The set of Turing Machines.
    (b) Uncountable set: $\mathcal{R}$, The set of all formal languages $\mathcal{L} = \{L \mid L \subseteq \sum^*\}$.

13. $A_{TM} = \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$. [If $A_T M$ is decidable, then output *accpet* when M accept w and output *reject* when M reject or looping on w. In practice, we can not determine whether a turning machine is looping, so $A_T M$ is undecidable.]

14. **Co-Turning recognizable languages:** The complement of a language A, $\mathcal{C}(A)$ is recognizable. [Complement of a language A, $\mathcal{C}(A)$ is the language consisting of all strings that doesn't belong to A.]

Theorems, Corollaries

1. A_{DFA}, A_{NFA}, E_{DFA},EQ_{DFA}, ALL_{DFA}, A_{CFG} are decidable languages.
2. EQ_{CFG} is not decidable language.
3. The proof of EQ_{DFA}, ALL_{DFA} is according to the complementary relationship with E_{DFA}.
4. A_{TM} is undecidable. (Proof by contradiction $p64 - p69$ of the slides *Decidability*.)
5. A language A is decidable iff both A and $\mathcal{C}(A)$ are Turing-recognizable.

Chapter 5: Reducibility

Definitions

1. **Reduction:**
   (a) Reduction is a terminating process. [An brief explanation about reduction: If A can reduce to B, it implies that problem B is at least as harder as A. As results, if $B$is solvable, then A must be solvable (A is easier than B). If A is unsolvable, then B also unsolvable (B is harder than A)].
   (b) When a problem A is reduced to problem B, solving A cannot be harder than the sum of reduction and solving B, because a solution to B gives a solution to A.
   (c) **If A is reduced to B and B is decidable, then A is decidable (because a solution to B solves A in finite number of steps)**.
   (d) **If A is undecidable and reducible to B then B is undecidable (because if B would be decidable then A would, which is a contradiction.)**
2. $HALT_{TM} = \{\langle M, w\rangle \mid M$ is a TM $\wedge M$ halts on $w\}$.[Halting Problem: Determine whether a Turing machine M halts on an input w. If M can be decided, then M is accepted when it halts; M is rejected when it has loop]
3. $E_{TM} = \{\langle M\rangle \mid$ M is a TM $\wedge L(M) = \emptyset\}$.

4. $Regular_{TM} = \{\langle M\rangle \mid$ M is a TM $\wedge L(M)$ is regular $\}$. [*i.e.*, whether a TM recognizes a regular language.]

5. Summary of Linear Bounded Automata is ignored here, because I don't review it...

Theorems, Corollaries

1. $HALT_{TM}$ is undecidable: Using the properties of reducing A_{TM} to $HALT_{TM}$. Proof can be found in $p216$ of spiser's textbook.

2. E_{TM} is undecidable: Using the properties of reducing A_{TM} to E_{TM}. Proof can be found in $p217$ of spiser's textbook.

3. $Regular_{TM}$ is undecidable. Proof can be found in $p218$ of spiser's textbook.

0.7 Undecidable Problems: Methodology

To prove that a problem P is undecidable by reduction method, it proceeds as follows:

1. Find a problem Q (A_TM,E_TM, etc) is known to be undecidable.

2. Assume that P is decidable by a TM M_p.

3. Use the TM M_p to construct a decider M_Q that solve Q.

4. Since it is already know that Q is undecidable M_Q cannot exist. Hence M_p cannot exist either, thus P is undecidable by contradiction.

Samples can be found in the proof of Theorems.

Chapter 7: Time Complexity

Definitions

1. **Running time of a TM A:** Number of steps the TM has moved, which is represented as a function of the length of input n, *i.e.* $f(n)$.

2. **Worst-case-analysis:** the longest running time of all inputs of a particular length n.

3. **Average-case-analysis:** the average of all running times of inputs of a particular length.

4. **Big-O:** Let f and g be functions, $f, g : \mathcal{N} \rightarrow \mathcal{R}$. $f(n) = O(g(n))$ if positive integers c and n_0 **exist** such that for $\forall n \geq n_0, f(n) \leq c(g(n))$.

5. **Small-o:** Let f and g be functions, $f, g : \mathcal{N} \rightarrow \mathcal{R}$. $f(n) = o(g(n))$ if $\lim_{n\rightarrow\infty} \frac{f(n)}{g(n)} = 0$. *i.e.*, $f(n) = o(g(n))$ means that for **any** real $c > 0$ there exists n_0 such that $f(n) < cg(n)$ for all $n > n_0$.

6. Polynomial bounds: n^c for $c > 0$.

7. Exponential Bounds: $a^{cn^{\delta}}$, for $a > 1$, $c, \delta > 0$.

8. **TIME($t(n)$):**, Let $t : \mathcal{N} \rightarrow \mathcal{N}$ be a function. The time complexity class TIME($t(n)$) is the collection of languages that are decidable by an $O(t(n))$ time TM.

9. **NTIME($t(n)$):**, Let $t : \mathcal{N} \rightarrow \mathcal{N}$ be a function. The time complexity class NTIME($t(n)$) is the collection of languages that are decidable by an $O(t(n))$ time NTM.

10. The running time of a **NTM** $f(n)$ is the maximum number of steps that the NTM makes on any branch of its computation, on any input of length n.

11. **Class P:** P is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine, *i.e.*, $P = \cup_k TIME(n^k)$

12. $\langle\cdot\rangle$ to indicate a reasonable encoding (polynomial time) of one or more objects into a string, such as familiar encoding methods for graphs, automata, etc.

13. PATH $= \{(G, s, t)\ G$ is a direct graph that has a direct path from s to $t\}$.

14. RELPRIME $=\{\langle x, y\rangle \mid x$ and y are relative prime $\}$.

15. HAMPATH$=\{\langle G, s, t\rangle | G\}$ is a directed graph with a Hamiltonian path from s to t. (Each vertex visited once).

16. **Verifier:** A verifier for a language A is an algorithm V where: $A = \{w | V\ accpets \langle w, c\rangle$ for some string $c\}$.

17. **Class NP:** NP is a class of languages that have polynomial time verifiers. (NP is short cut for Nondeterministic Polynomial time).

18. A langugage $A \in NPC$ if: 1) $A \in NP$, 2) For every $L \in NP$, it reduces to NP in polynomial.

Theorems, Corollaries

1. Any language that can be decided in $o(n \log n)$ on a single-tape TM is regular.

2. Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time multitape Turing machine has an equivalent $O(t^2(n))$ time single-tape Turing machine.

3. (Theorem 7.8) Let $t(n)$ and $s(n)$ be functions, where $t(n), s(n) \geq n$. If a multitape Turing machine takes $t(n)$ time and $s(n)$ space for any input of length n, then there is an equivalent $O(t(n)s(n))$ time single-tape Turing machie.

4. (Theorem 7.11) Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time non-deterministic single-tape Turing machine has an equivalent $2^{O(t(n))}$ time deterministic single-tape Turing machine.

5. TIME($t(n)$)$\subset$ NTIME($t(n)$) $\subset$ TIME($2^{ct(n)}$).

6. There is at most a polynomial difference between the time complexity of problems measured on deterministic single-tape and multitape TM.

7. There is at most an exponential difference between the time complexity of problems measured on Deterministic and Nondeterministic TM.

8. (Theorem 7.14) PATH $\in P$

9. (Theorem 7.15) RELPRIME $\in P$

10. HAMPATH $\in NP$

11. (Theorem 7.20) A language is in NP **iff** it is decided by some nondeterministic polynomial time Turing machine.

12. **NP** $\in \cup_k$NTIME(n^k).

13. If L reduce to L', and $L' \in P$, then $L \in P$.

14. If L reduce to L', and $L \notin P$, then $L' \notin P$.

15. Every $L \in NP$ reduce to SAT. [If $SAT \in P$, then $P = NP$.]

16. SAT is NP-Complete.