CSE 431 Theory of Computation **Spring 2014**

Lecture 4: April 10

Lecturer: Paul Beame *Scribe: Shao-Chieh Cheng, Siwakorn Srisakaokul*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

4.1 Decidability

In this lecture, we continue to describe Turing machines at a high-level.

We can encode arbitrary objects such as polynomials, graphs, and automata as strings. This allows us to define languauges in terms of these objects. One can then argue about the decidability and recognizability of these languages. We give some examples below.

Theorem 4.1 *For an input string w, The language*

$$A_{DFA} = \{\langle B, w\rangle \ : \ B \textit{ is a DFA that accepts } w\}$$

is decidable.

Proof: On input $\langle B, w\rangle$,

1. Check if the input $\langle B, w\rangle$ is well-formed.
2. Simulate the execution of B on w.
3. If the simulation ends in an accept state, *accept*; Otherwise, $reject$.

■

Theorem 4.2 *For an input string w, The language*

$$A_{NFA} = \{\langle B, w\rangle \ : \ B \textit{ is a NFA that accepts } w\}$$

is decidable.

Proof: On input $\langle B, w\rangle$,

1. Check if the input $\langle B, w\rangle$ is well-formed.
2. Apply the NFA $\rightarrow$ DFA convertor to build $\langle B', w\rangle$ where B' is an equivalent DFA to B.
3. Run TM M from Theorem 4.1 on the input $\langle B', w\rangle$.
4. If M accepts, *accept*; Otherwise, $reject$.

■

Theorem 4.3 *For an input string w, The language*

$$A_{REX} = \{\langle R, w\rangle \ : \ R \text{ is a regular expression that generates } w\}$$

is decidable.

Proof: On input $\langle R, w\rangle$,

1. Check if the input $\langle R, w\rangle$ is well-formed.
2. Convert the regular expression R to an equivalent DFA B.
3. Run TM M from Theorem 4.1 on the input $\langle B, w\rangle$.
4. If M accepts, *accept*; Otherwise, *reject*.

■

Theorem 4.4 *For a DFA B, let $L(B)$ denote the language of B. The language*

$$E_{DFA} = \{\langle B\rangle \ : \ B \text{ is a DFA and } L(B) = \emptyset\}$$

is decidable.

Proof: The Turing machine E described below decides E_{DFA}.

1. Check if input $\langle B\rangle$ is of the right form.
2. Graph-search (DFS or BFS) on the graph of B to see if any of the final states of B is reachable from its start state. If yes, reject $\langle B\rangle$. If not, accept $\langle B\rangle$.

■

Theorem 4.5 $EQ_{DFA} = \{\langle M_1, M_2\rangle \ : \ M_1, M_2 \text{ are DFAs and } L(M_1) = L(M_2)\}$ *is decidable.*

Proof: The Turing machine described below decides EQ_{DFA}.

1. Check if input $\langle M_1, M_2\rangle$ is of the right form.
2. Using the notation for DFAs given in Sipser, let $M_1 = (Q_1, \Sigma, \delta_1, p_0, F_1)$, and $M_2 = (Q_2, \Sigma, \delta_2, q_0, F_2)$. Create a new DFA $M = (Q, \Sigma, \delta, (p_0, q_0), F)$ such that:
   - $Q = Q_1 \times Q_2$ (i.e. the states of M is the Cartesian product of M_1 and M_2).
   - The transition function $\delta' : Q \times \Sigma \to Q$ of M is defined by $\delta'((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$
   - The final states $F = \{(p, q) \ : \ (p \in F_1 \wedge q \notin F_2) \text{ or } (p \notin F_1 \wedge q \in F_2)\}$

   It can be verified that $L(M) = L(M_1) \Delta L(M_2)$, where Δ denotes the set symmetric difference operator. Note that $L(M_1) = L(M_2)$ iff $L(M) = \emptyset$.
3. Feed $\langle M\rangle$ into the TM E for Theorem 4.4 . Accept $\langle M_1, M_2\rangle$ iff E accepts $\langle M\rangle$.

■

Theorem 4.6 *$A_{TM} = \{\langle M, w\rangle : M$ is a TM that accepts $w\}$ is recognizable but not decidable. A_{TM} is called the **Halting problem**.*

Proof: The Turing machine U described below recognizes A_{TM}.

1. Check if input $\langle M, w\rangle$ is of the right form.
2. Simulate M on input w step by step. Accept $\langle M, w\rangle$ iff M accepts w.

U is called a **universal Turing machine**.

We'll prove that A_{TM} is not decidable in the next couple of lectures. ■

As a final remark, note that there are only countably many Turing machines, but uncountably many languages over a (nonempty) alphabet Σ. Therefore, there exist languages that are not even Turing-recognizable.