CS340: Theory of Computation — Sem I 2017-18

# Lecture Notes 16: Decidable Properties of Regular and CFLs

*Raghunath Tewari* — IIT Kanpur

## 1 Decidability Properties of Regular Languages

We will look at some properties of regular languages that are decidable, such as whether a language has a certain string, or whether it is empty or not, and so on. The language is presented either via an automaton that accepts the language or a regular expression.

### 1.1 DFA Acceptance Problem

Consider the following language,

$$A_{DFA} = \{\langle D, w\rangle \mid D \text{ is a DFA that accepts the string } w\}.$$

Is $A_{DFA}$ decidable?
Here $\langle D, w\rangle$ denotes an encoding of the DFA $D$ and a string $w$. It can be any encoding scheme as long as it is consistent.
The answer to the above question is yes. It is easy to construct a TM that accepts the language $A_{DFA}$.

**Description of a halting TM that accepts $A_{DFA}$**

**Input:** $\langle D, w\rangle$ where $D$ is a DFA and $w$ is a string.

1. Simulate $D$ on $w$.
2. If it ends at an accept state then *accept* else *reject*.

**Note 1.** Observe that instead of a DFA, if a regular language is presented as an NFA or an RE, the corresponding language is still decidable. This is because we have seen algorithms to convert an NFA or an RE to a DFA (RE to NFA to DFA).

### 1.2 DFA Emptiness Problem

Consider the language,

$$E_{DFA} = \{\langle D\rangle \mid D \text{ is a DFA and } L(D) = \emptyset\}.$$

$E_{DFA}$ is decidable as well. This is because, $L(D) = \emptyset$ if and only if there is no path in the DFA from the start state to any accept state. We can use a graph traversal algorithm such as DFS or BFS to check this and answer accordingly.

### 1.3 DFA Equality Problem

Consider the language,

$$EQ_{DFA} = \{\langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are 2 DFAs and } L(D_1) = L(D_2)\}.$$

Of course we cannot simulate every possible string on both $D_1$ and $D_2$ and verify whether they accept the same language, since the number of strings is infinite.
The trick is to use some closure properties of regular languages.
Firstly, observe that

$$L(D_1) = L(D_2) \iff \left(L(D_1) \cap \overline{L(D_2)}\right) \cup \left(\overline{L(D_1)} \cap L(D_2)\right) = \emptyset.$$

Using closure properties, we can construct a DFA $D$ such that

$$L(D) = \left(L(D_1) \cap \overline{L(D_2)}\right) \cup \left(\overline{L(D_1)} \cap L(D_2)\right).$$

Now using the algorithm for $E_{DFA}$ we can check whether $L(D) = \emptyset$ or not, and hence whether $L(D_1) = L(D_2)$.

## 2 Decidability Properties of CFLs

We now discuss the similar decidability properties for CFLs. Since CFLs are defined by CFGs and PDAs, we will assume that the CFL is presented to us as a CFG or a PDA. Once again, it does not matter whether the CFL is presented using a CFG or a PDA, since we can convert one to the other.

### 2.1 CFG Acceptance Problem

Consider the language,

$$A_{CFG} = \{\langle G, w \rangle \mid G \text{ is a CFG that accepts the string } w\}.$$

Recall that $G$ accepts $w$ if $S \overset{*}{\Rightarrow} w$, where $S$ is the start variable of $G$. So one possibility is to cycle through every possible derivation from $S$ and see if it yields $w$. However for an arbitrary grammar we do not know a priori how many steps are there is the derivation and hence we would not know when to stop. Fortunately Chomsky Normal Form comes to our rescue in this case. We know that every CFG can be converted to a CFG in Chomsky Normal Form. Secondly, for a grammar $G$ in Chomsky Normal Form, the number of steps in the derivation of a non-empty string $w \in L(G)$ is $2|w| - 1$ and is 1 for $\epsilon$. I leave this as an exercise.

**Exercise 1.** Let $G$ be a grammar in Chomsky Normal Form and $w \in L(G)$. Show that if $w$ is not the empty string then the length of any derivation of $w$ in $G$ is $2|w| - 1$ and if $w = \epsilon$ then the length of any derivation is 1.

#### Description of a halting TM that accepts $A_{CFG}$

**Input:** $\langle G, w \rangle$ where $G$ is a CFG that accepts the string $w$.

1. Convert $G$ to an equivalent CNF grammar, say $G'$.

2. List all derivations of length $2|w|-1$ if $w \neq \epsilon$ and all derivations of length 1 if $w = \epsilon$.
3. If one of the derivations generate $w$ then *accept* else *reject*.

**Note 2.** There is a polynomial time solution for this problem by Cocke, Younger and Kasami using bottom-up parsing and dynamic programming. This is popularly known as the CYK algorithm. It too assumes that the grammar is given in Chomsky Normal Form and gives an $O(n^3 \cdot |G|)$ time algorithm, where $|G|$ is the size of the grammar and $n$ is the length of the input string.

## 2.2 CFG Emptiness Problem

Consider the language,

$$E_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}.$$

**Description of a halting TM that accepts $E_{CFG}$**

**Input:** $\langle G \rangle$ where $G$ is a CFG and $L(G) = \emptyset$.

1. Mark all terminals in $G$.
2. For every rule $A \to X_1 X_2 \ldots X_k$, mark $A$ if all symbols $X_1, X_2, \ldots, X_k$ are marked. Repeat this step until no new symbols get marked.
3. If the start symbol is not marked then *accept* else *reject*.

## 2.3 CFG Equality Problem

Consider the language,

$$EQ_{CFG} = \{\langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are two CFGs and } L(G_1) = L(G_2)\}.$$

One might be tempted to claim that $EQ_{CFG}$ is also decidable but as it turns out it is *not decidable*. We will see a technique to prove this later in the course. For now think about the following questions.

**Exercise 2.** Is $EQ_{CFG}$ Turing recognizable? Is $\overline{EQ_{CFG}}$ Turing recognizable?