Theory of Computation Notes: Turing-Decidable Languages

Arjun Chandrasekhar

Here we will see that it is possible to determine many properties about DFAs. Given an arbitrary DFA, we can determine whether that DFA will reject every string, whether it will accept any string of a certain form, and even whether it is equivalent to another DFA.

1 DFA Acceptance

First we consider the following language

$$\mathrm{A}_{\mathrm{DFA}} = \{\langle D, w\rangle | \text{D is a DFA}, w \in L(D)\}$$

Here we receive as input two things: the description of some DFA D, as well as some string. We don't know what the DFA is ahead of time but want to figure out if D will accept w.

Theorem 1.1. $\mathrm{A}_{\mathrm{DFA}}$ is decidable.

Proof. We construct a machine M to decide $\mathrm{A}_{\mathrm{DFA}}$ as follows:

1. Construct the state graph for D
2. Run w through D
3. Accept if and only if w reaches an accept state

□

Corollary 1.1. The following language is decidable

$$\mathrm{A}_{\mathrm{NFA}} = \{\langle N, w\rangle | \text{N is an NFA}, w \in L(N)\}$$

Proof. We construct a machine M to decide $\mathrm{A}_{\mathrm{NFA}}$ as follows:

1. Convert N to an equivalent DFA D
2. Accept if and only if if $\langle D, w\rangle \in \mathrm{A}_{\mathrm{DFA}}$

□

Corollary 1.2. The following language is decidable

$$\mathrm{A}_{\mathrm{REG}} = \{\langle R, w\rangle | \text{R is a Regex}, w \in L(N)\}$$

Proof. We construct a machine M to decide $\mathrm{A}_{\mathrm{REG}}$ as follows:

1. Convert R to an equivalent regex N
2. Accept if and only if $\langle N, w\rangle \in \mathrm{A}_{\mathrm{NFA}}$

□

2 DFA Emptiness

We start by considering following language

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D\rangle \mid \text{D is a DFA}, L(D) = \emptyset\}$$

Here $\langle D\rangle$ is a description of a DFA, i.e. a list of all the states and transitions.

Theorem 2.1. $\mathrm{E}_{\mathrm{DFA}}$ is decidable.

Proof. We construct a machine M that does the following:

1. Construct the state graph for D
2. Check if there is a path from the start state of D to any of the accept states
3. If there is no path to any accept state, then M accepts
4. Otherwise, M rejects

□

Corollary 2.1. The following language is decidable.

$$L = \{\langle D\rangle | D \text{ is a DFA}, L(D) \neq \emptyset\}$$

Proof. First we prove it by construction. To decide L we construct a machine M that does the following:

1. Construct the state graph for D
2. Check if there is a path from the start state of D to any of the accept states
3. If there a path to *any* accept state, then M accepts
4. Otherwise, M rejects

□

Proof. The second proof makes use of the fact that decidable languages are closed under complement. We note that $L = {\mathrm{E}_{\mathrm{DFA}}}^c$. Since $\mathrm{E}_{\mathrm{DFA}}$ is decidable, so is L. □

$\mathrm{E}_{\mathrm{DFA}}$ will be the foundational piece for proving that other DFA properties are decidable.

3 Decidable DFA Properties

Lemma 3.1. The following language is decidable

$$L = \{\langle D\rangle | x111y \in L(D) \text{ for at least one } x, y \in \Sigma^*\}$$

Proof. Let $L_2 = \Sigma^*111\Sigma^*$. We first note the following:

- L_2 is a regular language that can be recognized by a DFA
- Regular languages are closed under intersection
- If $\langle D\rangle \in L$ then $L(D) \cap L_2 \neq \emptyset$. If D accepts some $x111y$, that same $x111y$ is also a member of L_2. Thus the languages intersect somewhere.

To decide L we construct a machine that does the following:

1. Create a DFA D_2 that recognizes $\Sigma^*111\Sigma^*$
2. Create a DFA D_3 that recognizes $L(D) \cap L(D_2)$
3. Accept if and only if $\langle D_3 \rangle \notin E_{DFA}$

□

Lemma 3.2. The following language is decidable

$$L = \{\langle D \rangle | x111y \in L(D) \text{ for all } x, y \in \Sigma^*\}$$

Proof. Here it's not good enough to check that $L(D)$ intersects with $\Sigma^*111\Sigma^*$ for at least one string. We need to check that D accepts *every* string that is part of $\Sigma^*111\Sigma^*$. Alternately, we want to check that $\Sigma^*111\Sigma^* \subseteq L(D)$.

Again, we will refer to $L_2 = \Sigma^*111\Sigma^*$. To decide L we note the following things

- L_2 is regular and it can be recognized by a DFA
- Regular languages are closed under complement and intersection
- If $L_2 \subseteq L(D)$, then $L_2 \cap L(D)^c = \emptyset$. If D accepts *every* string in L_2, it rejects nothing from L_2, so L_2 has nothing in common with $L(D)^c$

To decide L we construct a machine M that does the following:

1. Construct a DFA D_2 for $\Sigma^*111\Sigma^*$
2. Construct at DFA D_3 that recognizes $L(D)^c$
3. Construcat a DFA D_4 that recognizes $L(D_2) \cap L(D_3)$
4. Accept if and only if $\langle D_4 \rangle \in \mathrm{E_{DFA}}$

□

4 DFA Equality

Can we check if two DFAs are equivalent? The answer is in fact yes!

Theorem 4.1. The following language is decidable

$$\mathrm{EQ_{DFA}} = \{\langle D_1, D_2 \rangle | L(D_1) = L(D_2)\}$$

Proof. Suppose $L(D_1) = L(D_2)$. Then the following statements are true:

- $L(D_1) \cap L(D_2)^c = \emptyset$, i.e. $L(D_1)$ does not include any strings that are not part of $L(D_2)$
- $L(D_1)^c \cap L(D_2) = \emptyset$ (same logic)
- Because of the first two statements, we see that $(L(D_1) \cap L(D_2)^c) \cup (L(D_1)^c \cap L(D_2))$
- Regular languages are closed under union, intersection, and complement

To decide $\mathrm{EQ_{DFA}}$ We construct a machine M that does the following:

1. Construct a DFA D_3 for $L(D_1) \cap L(D_2)^c$
2. Construct a DFA D_4 for $L(D_1)^c \cap L(D_2)$

3. Construct a DFA D_5 for $L(D_4) \cup L(D_5)$
4. Accept if and only if $\langle D_5 \rangle \in \mathrm{E}_{\mathrm{DFA}}$

$\square$

Lemma 4.1. The following language is decidable

$$L = \{\langle D \rangle | L(D) = \Sigma^* 111 \Sigma^*\}$$

Proof. We decide L as follows:

1. Construct a DFA D_2 that recognizes $\Sigma^* 111 \Sigma^*$
2. Accept if and only if $\langle D, D_2 \rangle \in \mathrm{EQ}_{\mathrm{DFA}}$

$\square$

5 Decidable CFL Properties

Finally, we show that two languages related to properties about context-free grammars are decidable.

Theorem 5.1. The following language is decidable

$$\mathrm{A}_{\mathrm{CFG}} = \{\langle G, w \rangle | \text{G is a CFG}, w \in L(G)\}$$

Proof. To prove that $\mathrm{A}_{\mathrm{CFG}}$ is decidable, we note the following two facts:

- Every CFG has an equivalent Chomsky Normal Form grammar
- Suppose G is in CNF, and G generates a string w of length n. Then G generates w in exactly $2n - 1$ steps.

We decide $\mathrm{A}_{\mathrm{CFG}}$ as follows:

1. Convert G to an equivalent grammar G' that is in CNF
2. Let $n = |w|$
3. Try all possible derivations in G' that use $2n - 1$ steps
4. If any derivation generates w, accept G
5. If every derivation fails, reject G

$\square$

Theorem 5.2. The following language is decidable

$$\mathrm{A}_{\mathrm{PDA}} = \{\langle P, w \rangle | \text{P is a PDA}, w \in L(P)\}$$

Proof. We decide $\mathrm{A}_{\mathrm{PDA}}$ as follows:

1. Convert P to an equivalent CFG G
2. Accept if and only if $\langle P, w \rangle \in \mathrm{A}_{\mathrm{CFG}}$

$\square$

Theorem 5.3. The following language is decidable

$$\mathrm{E}_{\mathrm{CFG}} = \{\langle G\rangle | \text{G is a CFG}, L(G) = \emptyset\}$$

Proof. We decide $\mathrm{E}_{\mathrm{CFG}}$ as follows:

1. 'Mark' every terimal symbol
2. Repeat the following until no new rule is marked:
   (a) Go through each rule $A \to U_1U_2 \dots U_n$
   (b) If $U_1, U_2, \dots, U_n$ are all marked, then mark A
3. Accept if and only the start variable S is <u>not</u> marked

□