⑤ <u>Checking whether two given DFA's accept the same language</u>

Given B_1, B_2 DFA's, test whether $L(B_1) = L(B_2)$.
We rewrite this problem as the language

$$EQ_{DFA} = \{\langle B_1, B_2\rangle \mid B_1 \text{ and } B_2 \text{ are DFA's and } L(B_1) = L(B_2)\}.$$

<u>Theorem</u> EQ_{DFA} is a Turing-decidable language.

<u>Proof</u> Given two sets Γ and Σ, $\Gamma \neq \Sigma$ if $\exists\, x \in \Gamma$ such that $x \notin \Sigma$ (i.e. $\Gamma \setminus \Sigma \neq \emptyset$) or $\exists\, x \in \Sigma$ such that $x \notin \Gamma$ (i.e. $\Sigma \setminus \Gamma \neq \emptyset$). Recall from our unit on set theory that $\Gamma \setminus \Sigma = \Gamma \cap \overline{\Sigma}$, Γ intersect the complement of Σ. Similarly, $\Sigma \setminus \Gamma = \Sigma \cap \overline{\Gamma}$. Therefore, $\Gamma \neq \Sigma \iff (\Gamma \cap \overline{\Sigma}) \cup (\Sigma \cap \overline{\Gamma}) \neq \emptyset$. This expression is called the <u>symmetric difference</u> of sets Γ and Σ in set theory. Now, returning to our problem, note that B_1 and B_2 are DFA's $\Rightarrow L(B_1)$ and $L(B_2)$ are regular languages. Furthermore, we showed the set of regular languages is closed under union, intersection, and the taking of complements $\Rightarrow \left(L(B_1) \cap \overline{L(B_2)}\right) \cup \left(L(B_2) \cap \overline{L(B_1)}\right)$ is a regular language $\Rightarrow \exists\, C$ a DFA that recognizes the symmetric difference of $L(B_1)$ and $L(B_2)$ $\left(L(B_1) \cap \overline{L(B_2)}\right) \cup \left(L(B_2) \cap \overline{L(B_1)}\right)$. $L(B_1) = L(B_2)$ if this symmetric difference is empty $\Rightarrow \forall \langle B_1, B_2\rangle \in EQ_{DFA}$ $\exists \langle C\rangle \in E_{DFA}$, the language corresponding to the emptiness testing problem. Since E_{DFA} is Turing-decidable, EQ_{DFA} is Turing-decidable.

q.e.d.)

Next, we look at context-free grammars (CFG's) that we studied last term.

⑥ $L_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \text{ is a string}\}$

<u>Theorem</u> L_{CFG} is a Turing-decidable language.

<u>Sketch of proof</u> We could try to go through all possible applications of production rules allowable under G to see whether we can generate w, but infinitely many derivations may need to be tried. Therefore,

if G does not generate w, our algorithm would not halt. We would thus have a Turing machine that is a recognizer but not a decider. To get a decider, we have to put G into a special form called a Chomsky normal form that takes $2n-1$ steps to generate a string w of length n. We do not need to know what a Chomsky normal form is, just that one exists in order to write down our decider M:

M = on input $\langle G, w \rangle$, where G is a context-free grammar and w is a string.

1. Convert G to an equivalent grammar in Chomsky normal form.
2. List all derivations with $2n-1$ steps, where n is the length of w if $n > 0$. If $n = 0$, list all derivations with one step.
3. If any of these derivations generates w, then accept; otherwise, reject.

(q.e.d.)

⑦ <u>Emptiness testing for context-free grammars</u>

Given a context-free grammar G, figure out whether the language it generates $L(G)$ is empty or not.

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

<u>Theorem</u> E_{CFG} is a Turing-decidable language.

<u>Proof</u> We use a similar marking argument as we did to show E_{DFA} was Turing-decidable. We outline the Turing machine as

M = on input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G
2. Repeat until no new variables get marked:
3. Mark any nonterminal $\langle T \rangle$ if G contains a production rule $\langle T \rangle \to u_1 \cdots u_k$, and each symbol (terminal or non-terminal) $u_1, \ldots, u_k$ has already been marked.

4. If the start symbol $\langle S \rangle$ is not marked, accept; otherwise, reject.

As we can see from step 4, if $\langle S \rangle$ is marked, then the context-free grammar will end up generating at least one string as all terminals have already been marked in step 1. Therefore, $L(G) \neq \emptyset$, and we reject G.

(q.e.d.)

⑧ Equivalence problem for context-free grammars

Given two context-free grammars, G_1 and G_2, determine whether they generate the same language, i.e. $L(G_1) = L(G_2)$.

Rewrite this problem as a language:

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

To solve the equivalence problem for DFA's, we used the symmetric difference and the fact that the emptiness problem for DFA's is Turing-decidable. In this case, the emptiness problem for CFG's is Turing-decidable as we just proved, but the symmetric difference argument does <u>NOT</u> work as the set of languages produced by context-free grammars is <u>NOT</u> closed under complements or intersection so the following result is true instead:

<u>Proposition</u> EQ_{CFG} is not a Turing-decidable language.

This proposition is proven using a technique called reducibility.