Now we have algorithms
We want to check problems solvable or not by computers
Need a TM to decide it
i.e., accept/reject in finite \# steps
We will show some examples
Acceptance Problems for DFA I

\[ A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts } w \} \]

- \( \langle B, w \rangle \) is the input
  - Note that a DFA can be represented as a string \((Q, \Sigma, \ldots)\)
- Is \(A_{DFA}\) decidable?
- Idea: input \(\langle B, w \rangle\)
  1. simulate \(B\) on \(w\)
  2. ends in an accept state \(\Rightarrow\) accept
     otherwise \(\Rightarrow\) reject
Proof of $A_{DFA}$ I

- Put

$$B = \langle Q, \Sigma, \delta, q_0, F \rangle$$

- into a tape

- Check if $w \in \Sigma^*$ and $B$ a valid DFA

- Simulate $w$ according to $\delta$ after the last element of $w$

- check if in a final state
$A_{NFA}$ is:

$$A_{NFA} = \{ \langle B, w \rangle \mid B \text{ is an NFA that accepts } w \}$$

- We can convert $B$ to a DFA and use the procedure for $A_{DFA}$
- It’s like to use the procedure for $A_{DFA}$ as a subroutine
\( A_{REX} = \{ \langle R, w \rangle \mid R: \text{regular expression generates } w \} \)

- It’s similar
- We convert \( R \) to a DFA first
- The key is that the conversion is a finite procedure
$E_{DFA} = \{ \langle A \rangle \mid A : DFA, L(A) = \emptyset \}$

i.e. $A$ accepts nothing

Idea:

DFA accepts something
$\iff$ reaching a final state from $q_0$ after several links

procedure

1. mark $q_0$
repeat until no new state marked
mark all

\[ a \rightarrow b, \]

where \( a \) has been marked

if no \( q \in F \) marked, accept. otherwise, reject

Example: a state diagram with 3 nodes and the following connections

\[ 1 \rightarrow 2, 3 \]
Marked states in running the procedure

1
12
12

- Each iteration: at least one new state marked
- At most $n$ iterations: $n$: $\# \text{ states}$
$EQ_{DFA} = \{ \langle A, B \rangle \mid A, B : DFAs, L(A) = L(B) \}$

- $EQ_{DFA}$ is decidable
- Idea for the proof:
  DFA C: exclusive or of A and B
  If $L(A) = L(B)$
  then $L(C) = \emptyset$
(latex source from https://texample.net/tikz/examples/set-operations-illustrated-with-venn-diagrams/)
Formally

\[ L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)) \]

- \( B \) DFA \( \Rightarrow \) so is \( \overline{B} \)
- \( A, B \) DFA \( \Rightarrow \) so is \( A \cup B, A \cap B \)
- Use \( E_{DFA} \)
Decidability and CFL I

- CFG

\[ A_{CFG} = \{ \langle G, w \rangle \mid G : CFG, \text{ generates } w \} \]

- We prove that \( A_{CFG} \) is decidable
- But an issue is the \( \infty \) possible derivations of a CFG
- For example,

\[ A \rightarrow B, B \rightarrow A \]

- Chomsky normal form

\[ A \rightarrow BC \\
A \rightarrow a \]
Decidability and CFL II

- Any $w, |w| = n$, derivation in exactly $2n - 1$ steps
- If $q$ rules, check $q^{2n-1}$ branches

Proof

1. Convert $G$ to Chomsky
2. Check all $q^{2n-1}$ branches

Results apply to PDA as well
\[ E_{CFG} = \{ \langle G \rangle \mid G : CFG, L(G) = \emptyset \} \]

- idea: bottom up setting to see if any string can be generated from the start variable. From

\[ A \rightarrow a \]

We search if \( \exists \)

\[ B \rightarrow A \]

- Proof:
Mark all terminals

Repeat until no new variables marked
if

\[ A \to U_1 \cdots U_k \]

and

all \( U_1, \ldots, U_k \) marked

⇒ mark \( A \)

If start state marked, accept
Otherwise, reject
$EQ_{CFG}$

$$EQ_{CFG} = \{ \langle G, H \rangle \mid G, H : CFG, L(G) = L(H) \}$$

- Remember that $EQ_{DFA}$ is decidable
- However, we cannot apply the same proof as CFL is not closed for $\cap$ and complementation
- It’s proved in Chapter 5 that this language is not decidable
- We do not discuss details
CFL decidable 1

- This question different from $A_{CFG}$ decidable or not
- How about converting PDA to a TM?
- For nondeterministic PDA we can do NTM
- But nondeterministic PDA may have $\infty$-long branches
  TM runs forever
- So converting PDA to TM does not really work
- A proof that works:
  Find grammar $G$ for this CFL
  Run TM for $\langle G, w \rangle$ by using $A_{CFG}$