Decidability
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2015
Problems about automata

Does $q_0 \xrightarrow{ab} q_1$ accept input $abb$?

We can formulate this question as a language

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

Is $A_{DFA}$ decidable?

One possible way to encode a DFA $D = (Q, \Sigma, \delta, q_0, F)$ and input $w$

$$((q_0, q_1)(a, b)(q_0, a, q_0)(q_0, b, q_1)(q_1, a, q_0)(q_1, b, q_1)) (q_0)(q_1)(abb)$$
Problems about automata

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

**Pseudocode:**
On input \( \langle D, w \rangle \), where
\[ D = (Q, \Sigma, \delta, q_0, F) \]

Set \( q \leftarrow q_0 \)
For \( i \leftarrow 1 \) to length\((w)\)
\[ q \leftarrow \delta(q, w_i) \]
If \( q \in F \) accept, else reject

**TM description:**
On input \( \langle D, w \rangle \), where \( D \) is a DFA, \( w \) is a string

Simulate \( D \) on input \( w \)
If simulation ends in an accept state, accept; else reject
Problems about automata

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

Turing machine details:

Check input is in correct format
  (Transition function is complete, no duplicate transitions)

Perform simulation:

\[
((q_0, q_1)(a, b)((q_0, a, q_0)(q_0, b, q_1)(q_1, a, q_0)(q_1, b, q_1))(q_0)(q_1))(\dot{\text{abb}}) \\
((q_0, q_1)(a, b)((q_0, a, q_0)(q_0, b, q_1)(q_1, a, q_0)(q_1, b, q_1))(q_0)(q_1))(\dot{\text{bb}}) \\
((q_0, q_1)(a, b)((q_0, a, q_0)(q_0, b, q_1)(q_1, a, q_0)(q_1, b, q_1))(q_0)(q_1))(\text{abb}) \\
((q_0, q_1)(a, b)((q_0, a, q_0)(q_0, b, q_1)(q_1, a, q_0)(q_1, b, q_1))(q_0)(q_1))(\text{abb})
\]
Problems about automata

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

Turing machine details:

Check input is in correct format  
(Transition function is complete, no duplicate transitions)

Perform simulation: (very high-level)

Put markers on start state of \( D \) and first symbol of \( w \)
Until marker for \( w \) reaches last symbol:
  Update both markers
If state marker is on accepting state, accept; else reject

**Conclusion**: \( A_{DFA} \) is decidable
Acceptance problems about automata

\[ A_{\text{DFA}} = \{ \langle D, w \rangle \mid D \text{ is a DFA that accepts input } w \} \]
\[ A_{\text{NFA}} = \{ \langle N, w \rangle \mid N \text{ is an NFA that accepts input } w \} \]
\[ A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression that generates } w \} \]

Which of these is decidable?
Acceptance problems about automata

\[ A_{\text{NFA}} = \{ \langle N, w \rangle \mid N \text{ is an NFA that accepts input } w \} \]

The following TM decides \( A_{\text{NFA}} \):
On input \( \langle N, w \rangle \) where \( N \) is an NFA and \( w \) is a string

Convert \( N \) to a DFA \( D \) using the conversion procedure from Lecture 3
Run TM \( M \) for \( A_{\text{DFA}} \) on input \( \langle D, w \rangle \)
If \( M \) accepts, accept; else reject

Conclusion: \( A_{\text{NFA}} \) is decidable
Acceptance problems about automata

\[ A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression that generates } w \} \]

The following TM decides \( A_{\text{REX}} \)

On input \( \langle R, w \rangle \), where \( R \) is a regular expression and \( w \) is a string

Convert \( R \) to an NFA \( N \) using the conversion procedure from Lecture 4
Run the TM for \( A_{\text{NFA}} \) on input \( \langle N, w \rangle \)
If \( N \) accepts, accept; else reject

\textbf{Conclusion:} \( A_{\text{REX}} \) is decidable  ✔️
Other problems about automata

\[ \text{MIN}_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a minimal DFA} \} \]

\[ \text{EQ}_{\text{DFA}} = \{ \langle D_1, D_2 \rangle \mid D_1 \text{ and } D_2 \text{ are DFAs and } L(D_1) = L(D_2) \} \]

\[ E_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA and } L(D) \text{ is empty} \} \]

Which of the above is decidable?
Other problems about automata

\[
\text{MIN}_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a minimal DFA} \}
\]

The following TM decides MIN\textsubscript{DFA}
On input \langle D \rangle, where \( R \) is a DFA

Run the DFA minimization algorithm from Lecture 7
If every pair of states is distinguishable, accept; else reject

**Conclusion:** MIN\textsubscript{DFA} is decidable
Other problems about automata

\[ \text{EQ}_{\text{DFA}} = \{ \langle D_1, D_2 \rangle \mid D_1 \text{ and } D_2 \text{ are DFAs and } L(D_1) = L(D_2) \} \]

The following TM decides \( \text{EQ}_{\text{DFA}} \)
On input \( \langle D_1, D_2 \rangle \), where \( D_1 \) and \( D_2 \) are DFAs

Run the DFA minimization algorithm from Lecture 7 on \( D_1 \) to obtain a minimal DFA \( D'_1 \)
Run the DFA minimization algorithm from Lecture 7 on \( D_2 \) to obtain a minimal DFA \( D'_2 \)
If \( D'_1 = D'_2 \), accept; else reject

**Conclusion:** \( \text{EQ}_{\text{DFA}} \) is decidable  

✓
Other problems about automata

\[ E_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA and } L(D) \text{ is empty} \} \]

The following TM \( T \) decides \( E_{\text{DFA}} \)
On input \( \langle D \rangle \), where \( D \) is a DFA

Run the TM \( S \) for \( \text{EQ}_{\text{DFA}} \) on input \( \langle D, \rightharpoonup \circ \rangle \)
If \( S \) accepts, \( T \) accepts; else \( T \) rejects

**Conclusion:** \( E_{\text{DFA}} \) is decidable  

\[ \checkmark \]
Problems about context-free grammars

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

\[ L \text{ where } L \text{ is a context-free language} \]

\[ \text{EQ}_{\text{CFG}} = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) = L(G_2) \} \]

Which of the above is decidable?
Problems about context-free grammars

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

The following TM \( V \) decides \( A_{\text{CFG}} \)
On input \( \langle G, w \rangle \), where \( G \) is a CFG and \( w \) is a string

Eliminate the \( \varepsilon \)- and unit productions from \( G \)
Convert \( G \) into Chomsky Normal Form \( G' \)
Run Cocke–Younger–Kasami algorithm on \( \langle G', w \rangle \)
If the CYK algorithm finds a parse tree, \( V \) accepts; else \( V \) rejects

Conclusion: \( A_{\text{CFG}} \) is decidable \( \checkmark \)
Problems about context-free grammars

Where $L$ is a context-free language

Let $L$ be a context-free language
There is a CFG $G$ for $L$

The following TM decides $L$
On input $w$

Run TM $V$ from the previous slide on input $\langle G, w \rangle$
If $V$ accepts, accept; else reject

Conclusion: every context-free language $L$ is decidable
Problems about context-free grammars

\[ \text{EQ}_{\text{CFG}} = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) = L(G_2) \} \]

is not decidable \( \times \)

What’s the difference between \( \text{EQ}_{\text{DFA}} \) and \( \text{EQ}_{\text{CFG}} \)?

To decide \( \text{EQ}_{\text{DFA}} \) we minimize both DFAs

But there is no method that, given a CFG or PDA, produces a unique equivalent minimal CFG or PDA
Universal Turing Machine and Undecidability
Turing Machines versus computers

A computer is a machine that manipulates data according to a list of instructions.

How does a Turing machine take a program as part of its input?
The universal Turing machine

The universal TM $U$ takes as inputs a program $M$ and a string $x$, and simulates $M$ on $w$.

The program $M$ itself is specified as a TM.
Turing machines as strings

A Turing machine is
\((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})\)

This Turing machine can be described by the string

\[
\langle M \rangle = (q, qa,qr)(\theta,1)(\theta,1,\square)
\]

\[
((q,q,\square/\square R)(q,qa,\theta/0R)(q,qr,1/1R))
\]

\[(q)(qa)(qr)\]
The universal Turing machine

$U$ on input $\langle M, w \rangle$:

**Simulate** $M$ on input $w$

If $M$ enters accept state, $U$ accepts

If $M$ enters reject state, $U$ rejects
Acceptance of Turing machines

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

U on input \( \langle M, w \rangle \) simulates \( M \) on input \( w \)

\[
\begin{align*}
M \text{ accepts } w & \quad \Downarrow & \quad M \text{ rejects } w & \quad \Downarrow & \quad M \text{ loops on } w & \quad \Downarrow \\
U \text{ accepts } \langle M, w \rangle & & U \text{ rejects } \langle M, w \rangle & & U \text{ loops on } \langle M, w \rangle & \\
\end{align*}
\]

TM \( U \) recognizes \( A_{\text{TM}} \) but does not decide \( A_{\text{TM}} \)
Recognizing versus deciding

Accept

Reject

Infinite loop

The language **recognized** by a TM $M$ is the set of all inputs that $M$ accepts

A TM **decides** language $L$ if it recognizes $L$ and halts on every input

A language $L$ is **decidable** if some TM decides $L$