Today's learning goals

• Introduction to Turing Machines
• Configurations and computations
• Deciders and Recognizers

Sipser Ch 1.4, 2.1
Beyond Push Down Automata

• Can you define a more powerful automaton that is able to recognize \( L_3 = \{ a^i b^j a^k \mid i=j=k \} \)?

• Can you recognize \( L_3 \)
  – with a 2PDA (Push Down Automaton with 2 stacks)?

• Can you recognize \( L_4 = \{ a^i b^j a^k b^h \mid i=j=k=h \} \)
  – with a 3PDA (Push Down Automaton with 3 stacks)?
  – with a 2PDA
Context-free languages

Regular languages
Turing machines

- Unlimited input
- Unlimited (read/write) memory
- Unlimited time
Turing machine computation

- Read/write head starts at leftmost position on tape
- Input string written on leftmost squares of tape, rest is blank
- Computation proceeds according to transition function:
  - Given current state of machine, and current symbol being read
    - 1) transitions to new state
    - 2) writes a symbol to its current position (overwriting existing symbol)
    - 3) moves the tape head L or R
- Computation ends if and when it enters either the accept or the reject state.
Formal definition of TM

A Turing machine is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\) where \(Q, \Sigma, \Gamma\) are all finite sets and

1. \(Q\) is the set of states
2. \(\Sigma\) is the input alphabet (not containing blank symbol)
3. \(\Gamma\) is the tape alphabet (including blank symbol as well as all symbols in \(\Sigma\))
4. \(\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\) is the transition function
5. \(q_0 \in Q\) is the start state
6. \(q_{\text{accept}} \in Q\) is the accept state
7. \(q_{\text{reject}} \in Q\) is the reject state

\(q_{\text{reject}} \neq q_{\text{accept}}\)
Formal definition:

A Turing machine is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})\) of finite sets and

1. \(Q\) is the set of states
2. \(\Sigma\) is the input alphabet (not containing \(\Gamma\))
3. \(\Gamma\) is the tape alphabet (including \(\Sigma\))
4. \(\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\) is the transition function
5. \(q_0 \in Q\) is the start state
6. \(q_{accept} \in Q\) is the accept state
7. \(q_{reject} \in Q\) is the reject state

Is this a deterministic model?

A) Yes
B) No
C) It depends on the state diagram
D) It depends on the input
E) I don’t know
Turing Machine example

- \( L = \{w#w \mid w \in \{a,b\}^*\} \) (No CFG/PDA!)

- Informal description: \( \Gamma = \{a,b,#,c,\square\} \)
  
  1) If input is 
     1) Move right skipping all c's
     2) If input is \( \square \), then accept, else reject
  
  2) Read a or b, store in internal state, and change it to c
  
  3) Move right until read 
  
  4) Move right skipping all c's
  
  5) Read a or b, matching the symbol read in step 1.
     1) If mismatch \( \rightarrow \) reject
     2) If match, overwrite with c
  
  6) Move left until read 
  
  7) Move left until read c
  
  8) Go back to 1).
Turing Machine (formal)

- \( L = \{w#w \mid w \text{ in } \{a,b\}^*\} \)  (No CFG/PDA!)
- Draw state diagram
Turing Machine (formal)

- $L = \{ w#w \mid w \text{ in } \{a,b\}^* \}$ (No CFG/PDA!)
- Draw state diagram in JFLAP
Turing Machine Computations

- Configuration: all the information store by the system at any point during computation
  - Internal state ($Q$)
  - Tape content ($\Gamma^*$)
  - Position of tape head

- Formally: ($\alpha, q, \beta$) in $\Gamma^* \times Q \times \Gamma^*$
  - $q$: internal state
  - $\alpha$: tape content to the left of tape head
  - $\beta$: tape content from tape head to the right
Computation: Example

• Assume
  - TM is in configuration (0110,q,111)
  - $\delta(q,0) = (p,1,L)$ and $\delta(q,1) = (p,1,R)$

• The next configuration is

A) (0110,p,111)
B) (011,p,0111)
C) (01100,p,11)
D) (01101,p,11)
E) I don’t know
**TM Computations**

- Fix TM $M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

- Computation of $M$ on input $w$:
  - Sequence of configurations
  - Start from initial configuration $C_0 = (\epsilon, q_s, w)$
  - Move from one configuration $C_i$ to the next $C_{i+1}$ according to $\delta$
  - Until reaching halting configuration $(\ldots, q_a, \ldots)$ or $(\ldots, q_r, \ldots)$

- Input is accepted if final state is $q_a$
- Input is rejected if final state is $q_r$
Fix TM $M=\langle Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r \rangle$.

Computation of $M$ on input $w$:
- Sequence of configurations
- Start from initial configuration $C_0=(\varepsilon, q_s, w)$
- Move from one configuration $C_i$ to the next $C_{i+1}$ according to $\delta$
- Until reaching halting configuration $(\ldots, q_a, \ldots)$ or $(\ldots, q_r, \ldots)$

Input is accepted if final state is $q_a$
Input is rejected if final state is $q_r$

What is the maximum length of a computation?
A) Same as input length
B) At most length of input
C) May be longer than input, but it is always finite
D) May be finite or infinite
E) I don’t know
TM Computations

- Fix TM $M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

- Computation of $M$ on input $w$:
  - Sequence of configurations
  - Start from initial configuration $C_0 = (\varepsilon, q_s, w)$
  - Move from one configuration $C_i$ to the next $C_{i+1}$ according to $\delta$
  - Until reaching halting configuration $(\ldots, q_a, \ldots)$ or $(\ldots, q_r, \ldots)$

- Input is accepted if final state is $q_a$

- Input is rejected if final state is $q_r$

Does $M$ need to read the whole input $w$ before halting?

A) Yes, computation terminates only after reading $w$
B) Only before accepting $w$
C) No. $M$ may accept or reject without reading $w$ entirely
D) I don’t know
TM Computations

- Fix TM $M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

- Computation of $M$ on input $w$:
  - Sequence of configurations
  - Start from initial configuration $C_0 = (\varepsilon, q_s, w)$
  - Move from one configuration $C_i$ to the next $C_{i+1}$ according to $\delta$
  - Until reaching halting configuration $(\varepsilon, q_a, \varepsilon)$ or $(\varepsilon, q_r, \varepsilon)$

- Input is accepted if final state is $q_a$
- Input is rejected if final state is $q_r$

Which of the following is true?

A) Computation may visit both $q_a$ and $q_r$

B) Computation may visit $q_a$ more than once

C) Computation may visit $q_b$ more than once

D) Computation may visit $q_s$ more than once

E) None of the above
Language of a TM

- $L(M) = \{w \mid M \text{ accepts } w\}$
- M may reject or loop on strings not in $L(M)$
- A language X is **recognizable** if $X = L(M)$ for some TM M
- A TM M is a decider if $M(w)$ halts on every input w
- A language X is **decidable** is $X = L(M)$ for some decider M
Language of a TM

- \( L(M) = \{ w \mid M \text{ accepts } w \} \)
- M may reject or loop on strings not in \( L(M) \)
- A language X is **recognizable** if X = \( L(M) \) for some TM M
- A TM M is a decider if M(w) halts on every input w
- A language X is **decidable** if X = \( L(M) \) for some decider M

Which of the following is true?

A) If X is decidable, then X is recognizable

B) If X is recognizable, then X is recognizable

C) If X is decidable, then X is decidable

D) I don’t know
Decidable vs Recognizable

- If A is decidable then A is recognizable
- If A is decidable then $A$ is recognizable
  - Equivalently, we may say that A is co-recognizable
- Summary: If A is decidable, then A is both recognizable and co-recognizable
- Question: If A is both recognizable and co-recognizable, can we conclude that A is decidable?
Decidable vs Recognizable

- If A is decidable then A is recognizable
- If A is decidable then $A$ is recognizable
  - Equivalently, we may say that A is co-recognizable
- Summary: If A is decidable, then A is recognizable and co-recognizable
- Question: If A is both recognizable and co-recognizable, can be conclude that A is decidable?

A) Yes
B) No
C) It depends on A
D) I don’t know
TM vs CFG/PDA vs RegEx/DFA

- There are Context Free languages that are not regular
  - DFA < PDA

- There are (Turing) decidable languages that are not context free
  - PDA < TM

- What about 2PDA?
  - 2PDA ≤ TM ?
  - TM ≤ 2PDA ?

- What about 3PDA? or 2TM? etc....
TM models

- TM with doubly infinite tape
- TM with 2-dimensional tape
- 2-Tape TM
- K-Tape TM
- Non-deterministic TM
- Theorem: All above models are equivalent
Equivalence between models

- Consider two models, e.g., TM and TM

- What does it mean for TM and 2TM to be equivalent?
  - Any TM $M$ can be transformed into a 2TM $M'$ such that $L(M) = L(M')$
  - Any 2TM $M'$ can be transformed into a TM $M$ such that $L(M) = L(M')$

- Strengthen: $M$ terminates iff $M'$ terminates
Church-Turing Thesis

• Theorem: TM, 2TM, k-TM, NTM, etc. are all equivalent
• Theorm: TM, λ-calculus, java, etc. are all equivalent
• Church-Turing thesis: any “reasonable” model of computation is equivalent to the TM
Reading

• Sipser 2
  - review, we will keep using DFA, CFG, etc.
• Sipser 3
  - Turing machines
  - equivalence between different models
• Read ahead: Sipser 4.1
  - decidable problems concerning DFA, CFG, etc.