Outline

- Deterministic PDAs
- Turing Machines and variants
  - multitape TMs
  - nondeterministic TMs
- Church-Turing Thesis
- decidable, RE, co-RE languages

Deterministic PDA

Proof:
- convert machine into “normal form”
  - always reads to end of input
  - always enters either an accept state or single distinguished “reject” state
- step 1: keep track of when we have read to end of input
- step 2: eliminate infinite loops

Deterministic PDA

step 1: keep track of when we have read to end of input

Deterministic PDA

determine step 2: eliminate infinite loops

for accept state q': replace outgoing “ε, ? → ?” transition with self-loop with same label
Deterministic PDA

step 2: eliminate infinite loops
- on input x, if infinite loop, then:

\[ \begin{align*}
  &\text{stack} \\
  &\text{height} \\
  &i_0 \quad i_1 \quad i_2 \quad i_3 \\
  &\text{infinite} \\
  &\text{sequence} i_0 < i_1 < i_2 < \cdots \\
  &\text{such that for all } k, \text{ stack height never decreases below } h(t(i_k)) \text{ after time } i_k
\end{align*} \]

Deterministic PDA

step 2: eliminate infinite loops
- infinite seq. \( i_0 < i_1 < \cdots \) such that for all \( k \), stack height never decreases below \( h(t(i_k)) \) after time \( i_k \)
- infinite subsequence \( j_0 < j_1 < j_2 < \cdots \) such that same transition is applied at each time \( j_k \)

\[ \begin{align*}
  &\text{safe to replace:} \\
  &p, t \rightarrow s \\
  &r', a, t \rightarrow t \text{ (for all } a, t) \\
  &\epsilon, t \rightarrow t \text{ (for all } t) \\
  &\epsilon, t \rightarrow s \\
  &p', \epsilon, t \rightarrow s \\
  &r', \epsilon, t \rightarrow s \\
  &\boxed{\text{never see any stack symbol below } t \text{ from } j_k \text{ on}} \\
  &\boxed{\text{we are in a periodic, deterministic sequence of stack operations}} \\
  &\boxed{\text{independent of the input}} \\
\end{align*} \]

Deterministic PDA

- finishing up…
- have a machine \( M \) with no infinite loops
- therefore it always reads to end of input
- either enters an accept state \( q' \), or enters “reject” state \( r' \)
- now, can swap: make \( r' \) unique accept state to get a machine recognizing complement of \( L \)

Summary

- Nondeterministic Pushdown Automata (NPDA)
- Context-Free Grammars (CFGs) describe Context-Free Languages (CFLs)
  - terminals, non-terminals
  - productions
  - yields, derivations
  - parse trees

- NDPAs and CFGs are equivalent
- CFL Pumping Lemma is used to show certain languages are not CFLs
Summary

- deterministic PDAs recognize DCFLs
- DCFLs are closed under complement

there is an efficient algorithm (based on dynamic programming) to determine if a string $x$ is generated by a given grammar $G$

So far…

- several models of computation
  - finite automata
  - pushdown automata
- fail to capture our intuitive notion of what is computable
  - regular languages
  - context free languages
  - all languages

A more powerful machine

- limitation of NPDA related to fact that their memory is stack-based (last in, first out)
- What is the simplest alteration that adds general-purpose “memory” to our machine?
  - Should be able to recognize, e.g., $\{a^n b^n c^n : n \geq 0\}$

Turing Machines

- New capabilities:
  - infinite tape
  - can read OR write to tape
  - read/write head can move left and right

Turing Machine

- Informal description:
  - input written on left-most squares of tape
  - rest of squares are blank
  - at each point, take a step determined by
    - current symbol being read
    - current state of finite control
  - a step consists of
    - writing new symbol
    - moving read/write head left or right
    - changing state
Example Turing Machine

language \( L = \{ w\#w : w \in \{0,1\}^* \} \)

Example TM diagram

TM formal definition

• A TM is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\) where:
  – \(Q\) is a finite set called the states
  – \(\Sigma\) is a finite set called the input alphabet
  – \(\Gamma\) is a finite set called the tape alphabet
  – \(\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\) is a function called the transition function
  – \(q_0\) is an element of \(Q\) called the start state
  – \(q_{\text{accept}}, q_{\text{reject}}\) are the accept and reject states

Example TM configurations

• At every step in a computation, a configuration determined:
  – the contents of the tape
  – the state
  – the location of the read/write head
• next step completely determined by current configuration
• shorthand: string \(uqv\) with \(u,v \in \Gamma^*\), \(q \in Q\)
TM configurations

- configuration $C_1$ yields configuration $C_2$ if TM can legally* move from $C_1$ to $C_2$ in 1 step
  - notation: $C_1 \Rightarrow C_2$
  - also: "yields in 1 step" notation: $C_1 \Rightarrow^1 C_2$
  - "yields in k steps" notation: $C_1 \Rightarrow^k C_2$
    if there exists configurations $D_1, D_2, \ldots, D_{k-1}$ for which $C_1 \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots \Rightarrow D_{k-1} \Rightarrow C_2$
  - also: "yields in some # of steps" ($C_1 \Rightarrow^* C_2$)

*Convention: TM halts upon entering $q_{\text{accept}}$, $q_{\text{reject}}$

TM acceptance

- start configuration: $q_0 w$ (w is input)
- accepting config.: any config. with state $q_{\text{accept}}$
- rejecting config.: any config. with state $q_{\text{reject}}$

TM $M$ accepts input $w$ if there exist configurations $C_1, C_2, \ldots, C_k$
  - $C_1$ is start configuration of $M$ on input $w$
  - $C_i \Rightarrow C_{i+1}$ for $i = 1, 2, 3, \ldots, k-1$
  - $C_k$ is an accepting configuration

Deciding and Recognizing

- TM $M$:
  - $L(M)$ is the language it recognizes
  - if $M$ rejects every $x \notin L(M)$ it decides $L$
  - set of languages recognized by some TM is called Turing-recognizable or recursively enumerable (RE)
  - set of languages decided by some TM is called Turing-decidable or decidable or recursive

* Formal definition of "yields":
  \[ uaq_i bv \Rightarrow uq_j acv \]
  if $\delta(q_i, b) = (q_j, c, L)$, and
  \[ uaq_i bv \Rightarrow uaq_j v \]
  if $\delta(q_i, b) = (q_j, c, R)$

- two special cases:
  - left end: $qaqv \Rightarrow qcv$ if $\delta(q_i, b) = (q_j, c, L)$
  - right end: $uaq_i$ same as $uaq_{i-1}$

- Formal definition of "yields" notation: $C_1 \Rightarrow C_2$
- "yields in 1 step" notation: $C_1 \Rightarrow^1 C_2$
- "yields in k steps" notation: $C_1 \Rightarrow^k C_2$
- "yields in some # of steps" ($C_1 \Rightarrow^* C_2$)

- Convention: TM halts upon entering $q_{\text{accept}}$, $q_{\text{reject}}$