Recall Rice's theorem for a Rec. enum. property

1) If \( L \) is in \( \mathcal{S} \) and if \( L' \) is any rec. enum. language such that \( L \subseteq L' \), then \( L' \) has property \( \mathcal{S} \).

2) If \( L \) is an infinite language in \( \mathcal{S} \), then there exists at least one finite subset of \( L \) that is in \( \mathcal{S} \).

3) The set of all finite languages in \( \mathcal{S} \) is enumerable, i.e. a Turing machine can list all the finite languages in \( \mathcal{S} \).

Listing finite languages not containing \( \varepsilon \)

- When a Turing machine is listing all the finite languages with a property, the output on a tape is something like:
  - \( 00$010$1$0001#00$111$0101$11100#0111# \ldots \) representing
    - \( \{00, 010, 1, 0001\} \)
    - \( \{00, 111, 0101, 11100\} \)
    - \( \{0111\} \)
    - etc
Example

- For example, we can have a TM list all finite languages not containing \(\varepsilon\) (also see the appendix to these notes to include \(\varepsilon\)):
- On Tape 1 generate all the strings in canonical order:
- B, then 0, replaced by 1, replaced by 00, replaced by 01, replaced by 10, replaced by 11, replaced by 000, etc

Example - II

- On a Tape 2, put #\& on the tape to represent the empty set, and a necessary marker “&”
- For every particular string on Tape 1, do the following:

Making the new sets

- For every set already on Tape 2 up to the “&” marker, copy that set to the end of the tape and add the single string on Tape 1

Move the marker

- When the symbol “&” is reached, move all the symbols left one cell to cover the “&”, writing the maker “&” at the right hand end of the languages generated so far
example tapes

- Tape 1: B
- Tape 2: #&
- Tape 1: 0
- Tape 2: #&0#
  then #0#&
- Tape 1: 1
- Tape 2: #0#&1#0$1#
  then #0#1#0$1#&

example tapes - II

- Tape 1: 00
- Tape 2: #0#1#0$1#&00#0$00#1$00#0$1$00#
  then #0#1#0$1#00#0$00#1$00#0$1$00#&
- etc

Push-down Automata

PDA’s

A pushdown automaton is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where
- $Q$ is a finite set of states
- $\Sigma$ is the input alphabet
- $\Gamma$ is the stack alphabet
rest of the tuple

- \( q_0 \) is the initial state
- \( Z_0 \) is the (stack) start symbol
- \( F \) is the subset of final states in \( Q \)
- \( \delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \text{finite subsets of } Q \times \Gamma^* \)

Instantaneous descriptions

- The instantaneous description (i.d.) for is \((q, w, \gamma)\), where
- \( q \) is the current state
- \( w \in \Sigma^* \) is the REMAINING input
- \( \gamma \in \Gamma^* \) is the content of the stack with the TOP OF STACK as the FIRST symbol

transitions and moves

- If \( \delta(q, a, Z) \) includes \((p, \beta)\), then if we have an i.d. \((q, aw, Z\alpha)\), we can (possibly) move to \((p, w, \beta\alpha)\)
- We write \((q, aw, Z\alpha) \|-- (p, w, \beta\alpha)\)
- The transitions CAN be sets: \( \delta(q, a, Z) = \{(p_1, \beta_1), (p_2, \beta_2), ..., (p_n, \beta_n)\} \)
- so sometimes there can be several moves from \((q, aw, Z\alpha)\)

there are also \(\varepsilon\)-transitions

- We can also have \(\delta(q, \varepsilon, Z)\) defined and it may include \((p, \beta)\)
- In this case, an i.d. \((q, aw, Z\alpha)\) can move to \((p, aw, \beta\alpha)\)
- We write \((q, aw, Z\alpha) \|-- (p, aw, \beta\alpha)\)
- no input is consumed
For every move, you must POP the stack.
Hence if the stack is empty, there can be no move, the PDA halts.
\( \delta(q, \varepsilon, Z) \) is not really an \( \varepsilon \)-transition since we are consuming the top of the stack.

Determinism means “no choice of moves”.
For a PDA this is subtle.
First \( \delta(q, a, Z) \) and \( \delta(q, \varepsilon, Z) \) must all have 0 or 1 elements, otherwise, for sure, we have a choice but ...

Thus the PDA is deterministic if
(1) \( \delta(q, a, Z) \) and \( \delta(q, \varepsilon, Z) \) have 0 or 1 elements for all \( q, a \) and \( Z \) AND
(2) IF \( \delta(q, \varepsilon, Z) \neq \emptyset \) for some \( q \) and \( Z \), THEN \( \delta(q, a, Z) = \emptyset \) for all \( a \) (for that particular \( q \) and \( Z \)).
Non-determinism is necessary

- Some CFL's require a non-deterministic PDA
- A CFG for \{ww^R : w \in (0 + 1)^*\} is
  \[ S \to 0S0 \mid 1S1 \mid \epsilon \]
- There is no deterministic automaton

Example: first 4 transitions

- Example 5.2:
  \[\{q_1, q_2\}, \{0, 1\}, \{R, B, G\}, \delta, q_1, R, \emptyset\]
  \[\delta(q_1, 0, R) = \{(q_1, BR)\}\]
  \[\delta(q_1, 1, R) = \{(q_1, GR)\}\]
  \[\delta(q_1, 0, B) = \{(q_1, BB),(q_2, \epsilon)\}\]
  \[\delta(q_1, 0, G) = \{(q_1, BG)\}\]

Running the PDA

\[\begin{array}{cccccc}
1 & 1 & 0 & 0 & 1 & 1 \\
\end{array}\]

remaining 6 transitions

- \[\delta(q_1, 1, B) = \{(q_1, GB)\}\]
- \[\delta(q_1, 1, G) = \{(q_1, GG),(q_2, \epsilon)\}\]
- \[\delta(q_2, 0, B) = \{(q_2, \epsilon)\}\]
- \[\delta(q_2, 1, G) = \{(q_2, \epsilon)\}\]
- \[\delta(q_1, \epsilon, R) = \{(q_2, \epsilon)\}\]
- \[\delta(q_2, \epsilon, R) = \{(q_2, \epsilon)\}\]
consume 1 or make the $\epsilon$-transition

consume 0 or make the $\epsilon$-transition

consume 1

consume 0
Finally

- Only one of the three can continue (the left-most one)
- The PDA accepts by empty stack

The language of a PDA

- Given a PDA, there are moves
  \((q, aw, Z\alpha) \rightarrow (p_1, w, \beta_1\alpha)\)
  \(\text{OR}\)
  \((q, aw, Z\alpha) \rightarrow (p_2, aw, \beta_2\alpha)\)
- Define \(\rightarrow^*\) as follows:
reflexive, transitive closure of |--

- $(q, w, \alpha) \vdash^* (q, w, \alpha)$
- If $(q_1, w_1, \alpha_1) |-- (q_2, w_2, \alpha_2) |--
  (q_3, w_3, \alpha_3) |-- \ldots |-- (q_n, w_n, \alpha_n)$
  then we write
- $(q_1, w_1, \alpha_1) \vdash^* (q_n, w_n, \alpha_n)$

Acceptance by final state

- A PDA accepts a string $w$ by final state if moves can be chosen so that
- $(q_0, w, Z_0) \vdash^* (q_n, \varepsilon, \varepsilon)$
  where $q_n$ is a final state; the stack contents $\alpha_n$ is not important but all
  the input must be consumed

Acceptance by empty stack

- A PDA accepts a string $w$ by empty stack if moves can be chosen so that
- $(q_0, w, Z_0) \vdash^* (q_n, \varepsilon, \varepsilon)$
  where $q_n$ is any state; the stack is empty and all the input must be
  consumed
- In this case we can take $F = \emptyset$

Theorems

- A language is accepted by empty stack using one PDA if and only if it
  is accepted by final state using some other PDA
- [on rare occasions the same PDA does both]
- PDA's accept exactly the CFL's
Outline proof

- The equivalence of the two types of acceptance is proved in Theorems 5.1, 5.2
- In both proofs a special extra stack symbol is put at the bottom of the stack

final state from empty stack

- If you are simulating a PDA that accepts by empty stack, then you move to a final state if you pop the stack down to that special symbol [when you reach the end of the input]
- (since hitting that bottom symbol means that the PDA being simulated emptied its stack)

empty stack from final state

- The extra stack symbol prevents the stack being emptied (hence accepting) before reaching the final state
- Once you reach a final state, transitions are added to pop the stack down to nothing

Greibach Normal Form (GNF)

- Sheila Greibach’s normal form (GNF) for a CFG is one where EVERY production has the form: $A \rightarrow a\alpha$, where $a \in T$ and $\alpha \in V^*$
- We saw this construction previously
CFG to PDA-I

- Given a CFG $G = (V, T, S, P)$, in GNF, a PDA for $L(G)$ is $\{\{q\}, T, V, q, S, \emptyset\}$ where
- $\delta(q, a, A) = \{(q, \gamma) : (A \rightarrow a\gamma) \in P\}$
- (see Theorem 5.3)

Example

- Previously we listed all finite languages not containing $\varepsilon$
- We can have a TM list all finite languages:
- On Tape 1 generate all the strings in canonical order with a marker $\$:$
- $\$B$, then $\$0$, replaced by $\$1$, replaced by $\$00$, replaced by $\$01$, replaced by $\$10$, replaced by $\$11$, replaced by $\$000$, etc

Appendix

Enumerating all finite languages

Example - II

- On a Tape 2, put $\$B#$& (where $\$ marks the start of the tape) on the tape to represent the empty set, the set containing $\varepsilon$ (using B) and a necessary marker “&”
- For every particular string on Tape 1, do the following:
Making the new sets

- For every set already on Tape 2 up to the “&” marker, copy that set to the end of the tape and add the single string on Tape 1 as you go along.

Move the marker

- When the symbol “&” is reached, move all the symbols left one cell to cover the “&”, writing the marker “&” at the right hand end of the languages generated so far.

Example tapes

- Tape 1: ¢B
- Tape 2: ¢#B#&
- Tape 1: ¢0
- Tape 2: ¢#B#0#B$0#B$0#& then ¢#B#0#B$0#B$0#B$0#&
- Tape 1: ¢1
- Tape 2: ¢#B#0#B$0#&1#B$1#0$1#B$0$1#B$0$1# then ¢#B#0#B$0#1#B$1#0$1#B$0$1#B$0$1#&

Example tapes - II

- Tape 1: ¢00
- Tape 2: c#B#0#B$0#1#B$1#0$1#B$0$1#&00#B$00#B$0$00#1$00#B$1$00#0$1$00#B$0$1$00#B$0$1$00#B$0$1$00# then c#B#0#B$0#1#B$1#0$1#B$0$1# then c#B#0#B$0#1#B$1#0$1#B$0$1#B$0$1#&00#B$0$00#1$00#B$1$00#0$1$00#B$0$1$00# then c#B#0#B$0#1#B$1#0$1#B$0$1#00#B$00#0$1$00#B$0$1$00#B$0$1$00# &
- etc