CS 154 – Lecture 6 Notes (4/17/00)

Decision Properties of Regular Languages

Given a (representation, e.g., RE, FA, of a) regular language L, what can we tell about L?

- Since there are algorithms to convert between any two representations, we can choose the rep that makes the test easiest.

Membership

Is string w in regular language L?

- Choose DFA representation for L.
- Simulate the DFA on input w.

Emptiness

Is $L = \emptyset$?

- Use DFA representation.
- Use a graph-reachability algorithm to test if at least one accepting state is reachable from the start state.

Finiteness

Is L a finite language?

- Note every finite language is regular (why?), but a regular language is not necessarily finite.

DFA method:

- Given a DFA for L, eliminate all states that are not reachable from the start state and all states that do not reach an accepting state.
- Test if there are any cycles in the remaining DFA; if so, L is infinite, if not, then L is finite.

RE method: Almost, we can look for a $*$ in the RE and say its language is infinite if there is one, finite if not. However, there are exceptions, e.g. $\mathbf{0}\epsilon^*\mathbf{1}$ or $\mathbf{0}^*\emptyset$. Thus:

1. Find subexpressions equivalent to $\emptyset$ by:
   - ❖ (Basis) $\emptyset$ is; ϵ and $\mathbf{a}$ are not.
   - ❖ (Induction) $E+F$ is iff both E and F are; EF is if either E or F are; E^* never is.
2. Eliminate subexpressions equivalent to $\emptyset$ by:
   - ❖ Replace $E+F$ or $F+E$ by F whenever E is and F isn't.
   - ❖ Replace E^* by ϵ whenever E is equivalent to $\emptyset$.

3. Now, find subexpressions that are equivalent to ϵ by:

    - ❖ (Basis) ϵ is; $\mathbf{a}$ isn't.

    - ❖ (Induction) $E+F$ is iff both E and F are; ditto EF; E^* is iff E is.

4. Now, we can tell if $L(R)$ is infinite by looking for a subexpression E^* such that E is not equivalent to ϵ.

Example

Consider $(\mathbf{0} + \mathbf{1}\emptyset)^* + \mathbf{1}\emptyset^*$.

- Step 1: $\emptyset$ (twice) and $\mathbf{1}\emptyset$ are subexpressions equivalent to $\emptyset$.

- Step 2: $\mathbf{0}^* + \mathbf{1}\epsilon$ remains.

- Step 3: only subexpression ϵ is equivalent to ϵ.

- Since $\mathbf{0}$ is starred, language is infinite.

Minimization of States

- Real goal is testing equivalence of (reps of) two regular languages.

- Interesting fact: DFA's have unique (up to state names) minimum-state equivalents.

    - ❖ But proof in course reader doesn't quite get to that point.

Distinguishable States

Key idea: find states p and q that are *distinguishable* because there is some input w that takes exactly one of p and q to an accepting state.

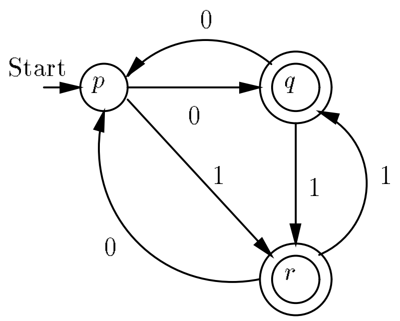
- Basis: any nonaccepting state is distinguishable from any accepting state ($w = \epsilon$).

- Induction: p and q are distinguishable if there is some input symbol a such that $\delta(p, a)$ is distinguishable from $\delta(q, a)$.

    - ❖ All other pairs of states are indistinguishable, and can be merged into one state.

Example (Very Simple)

Consider:



- p is distinguishable from q and r by basis.

Can we distinguish q from r?

- No string beginning with 0 works, because both states go to p, and therefore any string of the form $0x$ takes q and r to the same state.
- No string beginning with 1 works.
  - Technically, $\delta(q, 1) = r$ and $\delta(r, 1) = q$ are not distinguishable. Thus, induction does not tell us q and r are distinguishable.
  - What happens is that, starting in either q or r, as long as we have inputs 1, we are in one of the accepting states, and when a 0 is read, we go to the same state forever after.

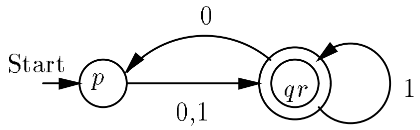
Constructing the Minimum-State DFA

- For each group of indistinguishable states, pick a "representative."
  - Note a group can be large, e.g., $q_1, q_2, \ldots, q_k$, if all pairs are indistinguishable.
  - Indistinguishability is transitive (why?) so indistinguishability partitions states.
- If p is a representative, and $\delta(p, a) = q$, in minimum-state DFA the transition from p on a is to the representative of q's group (to q itself if q is either alone in a group or a representative).
- State state is representative of the original start state.
- Accepting states are representatives of groups of accepting states.
  - Notice we could not have a "mixed" (accepting + nonaccepting) group (why?).

- Delete any state that is not reachable from the start state.

Example

For the DFA above, p is in a group by itself; $\{q, r\}$ is the other group.



Why Above Minimization Can't be Beaten

Suppose we have a DFA A, and we minimize it to construct a DFA M. Yet there is another DFA N that accepts the same language as A and M, yet has fewer states than M. Proof contradiction that this can't happen:

- Run the state-distinguishability process on the states of M and N together.
- Start states of M and N are indistinguishable because $L(M) = L(N)$.
- If $\{p, q\}$ are indistinguishable, then their successors on any one input symbol are also indistinguishable.
- Thus, since neither M not N could have an inaccessible state, every state of M is indistinguishable from at least one state of N.
- Since N has fewer states than M, there are two states of M that are indistinguishable from the same state of N, and therefore indistinguishable from each other.
- But M was designed so that all its states *are* distinguishable from each other.
- We have a contradiction, so the assumption that N exists is wrong, and M in fact has as few states as any equivalent DFA for A.
- In fact (stronger), there must be a 1-1 correspondence between the states of any other minimum-state N and the DFA M, showing that the minimum-state DFA for A is unique up to renaming of the states.

Context-Free Grammars

Notation for recursive description of languages. Example:

$$
\begin{array}{l}
Roll \rightarrow <ROLL> \ Class \ Studs \ </ROLL> \\
Class \rightarrow <CLASS> \ Text \ </CLASS> \\
Text \rightarrow Char \ Text \\
Text \rightarrow Char \\
Char \rightarrow a \cdots \text{(other chars)} \\
Studs \rightarrow Stud \ Studs \\
Studs \rightarrow \epsilon \\
Stud \rightarrow <STUD> \ Text \ </STUD>
\end{array}
$$

- Generates "documents" such as:

```
<ROLL><CLASS>cs154</CLASS>
    <STUD>Sally</STUD>
    <STUD>Fred</STUD>
</ROLL>
```

- *Variables* (e.g., *Studs*) represent sets of strings (i.e., languages).
  - In sensible grammars, these strings share some common characteristic or role.
- *Terminals* (e.g., a or $< ROLL >$) = symbols of which strings are composed.
  - "Tags" like $< \ ROLL \ >$ could be considered either a single terminal or the concatenation of 6 terminals.
- *Productions* = rules $Head \rightarrow Body$.
  - $Head$ is a variable.
  - *Body* is a string of zero or more variables and/or terminals.
- *Start Symbol* = variable that represents "the language."
- Notation: $G = (V, \Sigma, P, S)$ = (variables, terminals, productions, start symbol).

Example

Generates 0-1 strings such that each block of 0's is followed by at least as many 1's.

$$
\begin{array}{l}
S \rightarrow AS \mid \epsilon \\
A \rightarrow 0A1 \mid A1 \mid 01
\end{array}
$$

- Vertical bar separates different bodies for one head.

Derivations

- $\alpha A\beta \Rightarrow \alpha\gamma\beta$ whenever there is a production $A \rightarrow \gamma$.
  - Subscript with name of grammar, e.g., $\underset{G}{\Rightarrow}$, if necessary.

- Example: $011AS \Rightarrow 0110A1S$.

- $\alpha \stackrel{*}{\Rightarrow} \beta$ means string α can become β in zero or more derivation steps.

  - Examples: $011AS \stackrel{*}{\Rightarrow} 011AS$ (zero steps); $011AS \stackrel{*}{\Rightarrow} 0110A1S$ (one step); $011AS \stackrel{*}{\Rightarrow} 0110011$ (three steps).

Language of a CFG

$L(G) =$ set of terminal strings w such that $S \underset{G}{\stackrel{*}{\Rightarrow}} w$, where S is the start symbol.

Aside: Notation

- $a, b, \ldots$ terminals; $\ldots, y, z$ strings of terminals.
- Greek letters are strings of variables and/or terminals, often called *sentential forms*.
- $A, B, \ldots$ are variables.
- $\ldots, Y, Z$ are variables or terminals.
- S is typically the start symbol.

Leftmost/Rightmost Derivations

- Note choice of variable to replace at each step.
  - Derivations may appear different only because we make the *same* replacements in a different order.
  - To avoid such differences, we may restrict the choice.
- A *leftmost* derivation always replaces the leftmost variable in a sentential form.
  - Yields *left-sentential forms*.
- *Rightmost* defined analogously.
- $\underset{lm}{\Rightarrow}$, $\underset{rm}{\Rightarrow}$, etc., used to indicate derivations are leftmost or rightmost.

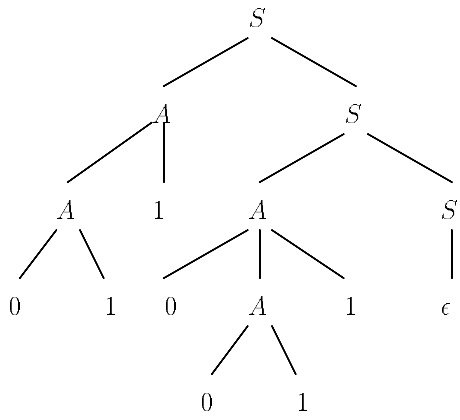
Example

- $S \underset{lm}{\Rightarrow} AS \underset{lm}{\Rightarrow} A1S \underset{lm}{\Rightarrow} 011S \underset{lm}{\Rightarrow} 011AS \underset{lm}{\Rightarrow} 0110A1S \underset{lm}{\Rightarrow} 0110011S \underset{lm}{\Rightarrow} 0110011$
- $S \underset{rm}{\Rightarrow} AS \underset{rm}{\Rightarrow} AAS \underset{rm}{\Rightarrow} AA \underset{rm}{\Rightarrow} A0A1 \underset{rm}{\Rightarrow} A0011 \underset{rm}{\Rightarrow} A10011 \underset{rm}{\Rightarrow} 0110011$

Derivation Trees

- Nodes = variables, terminals, or ϵ.
  - Variables at interior nodes; terminals and ϵ at leaves.
  - A leaf can be ϵ only if it is the only child of its parent.
- A node and its children from the left must form the head and body of a production.

Example



Equivalence of Parse Trees, Leftmost, and Rightmost Derivations

The following about a grammar $G = (V, \Sigma, P, S)$ and a terminal string w are all equivalent:

1. $S \overset{*}{\Rightarrow} w$ (i.e., w is in $L(G)$).
2. $S \underset{lm}{\overset{*}{\Rightarrow}} w$
3. $S \underset{rm}{\overset{*}{\Rightarrow}} w$
4. There is a parse tree for G with root S and *yield* (labels of leaves, from the left) w.

- Obviously (2) and (3) each imply (1).

Parse Tree Implies LM/RM Derivations

- Generalize all statements to talk about an arbitrary variable A in place of S.
  - Except now (1) no longer means w is in $L(G)$.
- Induction on the height of the parse tree.

Basis: Height 1: Tree is root A and leaves $w = a_1, a_2, \ldots, a_k$.

- $A \rightarrow w$ must be a production, so $A \underset{lm}{\Rightarrow} w$ and $A \underset{rm}{\Rightarrow} w$.

Induction: Height $>$ 1. Tree is root A with children $X_1, X_2, \ldots, X_k$.

- Those X_i's that are variables are roots of shorter trees.
    - ❖ Thus, the IH says that they have LM derivations of their yields.
- Construct a LM derivation of w from A by starting with $A \underset{lm}{\Rightarrow} X_1X_2\cdots X_k$, then using LM derivations from each X_i that is a variable, in order from the left.
- RM derivation analogous.

Derivations to Parse Trees

Induction on length of the derivation.

Basis: One step. There is an obvious parse tree.

Induction: More than one step.

- Let the first step be $A \Rightarrow X_1X_2\cdots X_k$.
- Subsequent changes can be reordered so that all changes to X_1 and the sentential forms that replace it are done first, then those for X_2, and so on (i.e., we can rewrite the deriviation as a LM derivation).
- The derivations from those X_i's that are variables are all shorter than the given deriviation, so the IH applies.
- By the IH, there are parse trees for each of these derivations.
- Make the roots of these trees be children of a new root labeled A.

Example

Consider derivation $S \Rightarrow AS \Rightarrow AAS \Rightarrow AA \Rightarrow A1A \Rightarrow A10A1 \Rightarrow 0110A1 \Rightarrow 0110011$

- Subderivation from A is: $A \Rightarrow A1 \Rightarrow 011$
- Subderivation from S is: $S \Rightarrow AS \Rightarrow A \Rightarrow 0A1 \Rightarrow 0011$
- Each has a parse tree; put them together with new root S.