Nonregular Languages

CS 81: Computability and Logic

October 27, 2010

How Can We Prove that a Language **is** Regular?

How Can We Prove that a Language **isn't** Regular?

Check the Intrinsic States

Theorem [Myhill-Nerode]:

A language is regular iff $\{ L_w \mid w \in \Sigma^* \}$ is finite.

$L = \{ 0^n 1^n \mid n \geq 0 \}$

- Consider L_ε, L_0, L_{00}, L_{000}, ...
  - $\varepsilon \in L_\varepsilon$ (and not the others)
  - $1 \in L_0$ (and not the others)
  - $11 \in L_{00}$ (and not the others)
  - $111 \in L_{000}$ (and not the others)
  - ...
- Therefore, L cannot be regular.

Mazes

- Suppose you enter a maze of twisty passages, all alike.
  (but with doors that open from only one side)

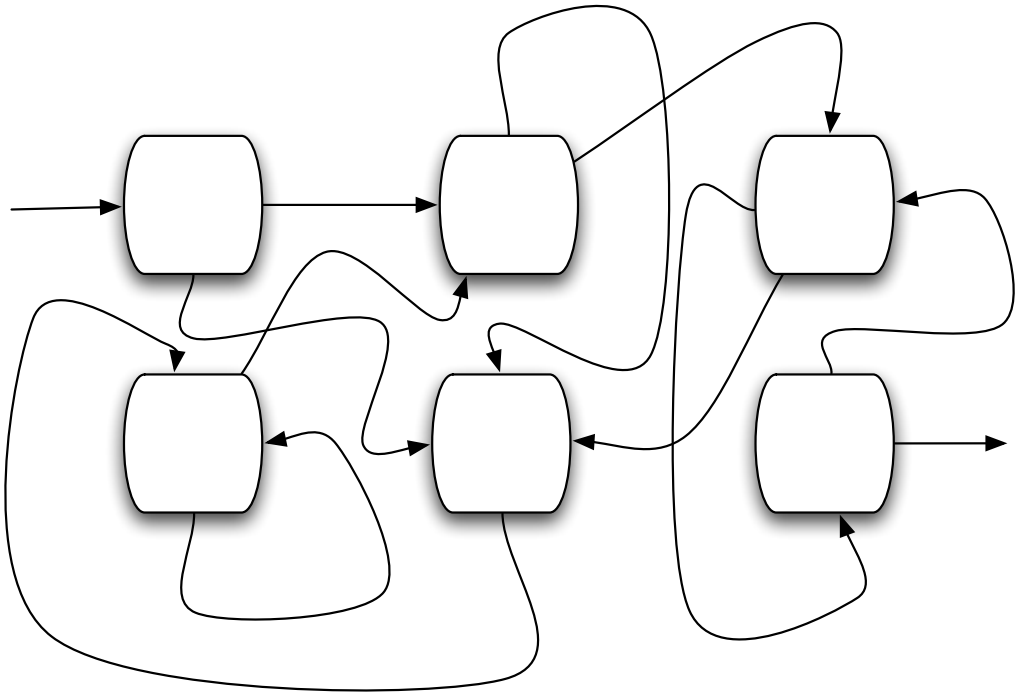
- You happen to know that this maze has exactly 19 rooms.
  You start wandering and pass through 27 rooms.
  What can you conclude?
- This wandering has brought you to an exit.
  What can you conclude about other solutions to the maze?
- Was there anything special about the numbers 19 and 27?

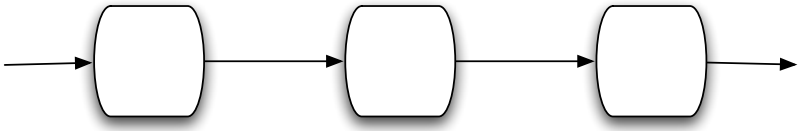
A "Theorem" About Finite Mazes

- For every finite maze there is a number p, such that
  - For every path through the maze s with $|s| \geq p$:
    - The path s contains at least one loop, which starts and ends within the first p steps.
    - There are infinitely many paths through the maze (at least one shorter, and arbitrarily many longer) whose lengths differ by a multiple of some constant.

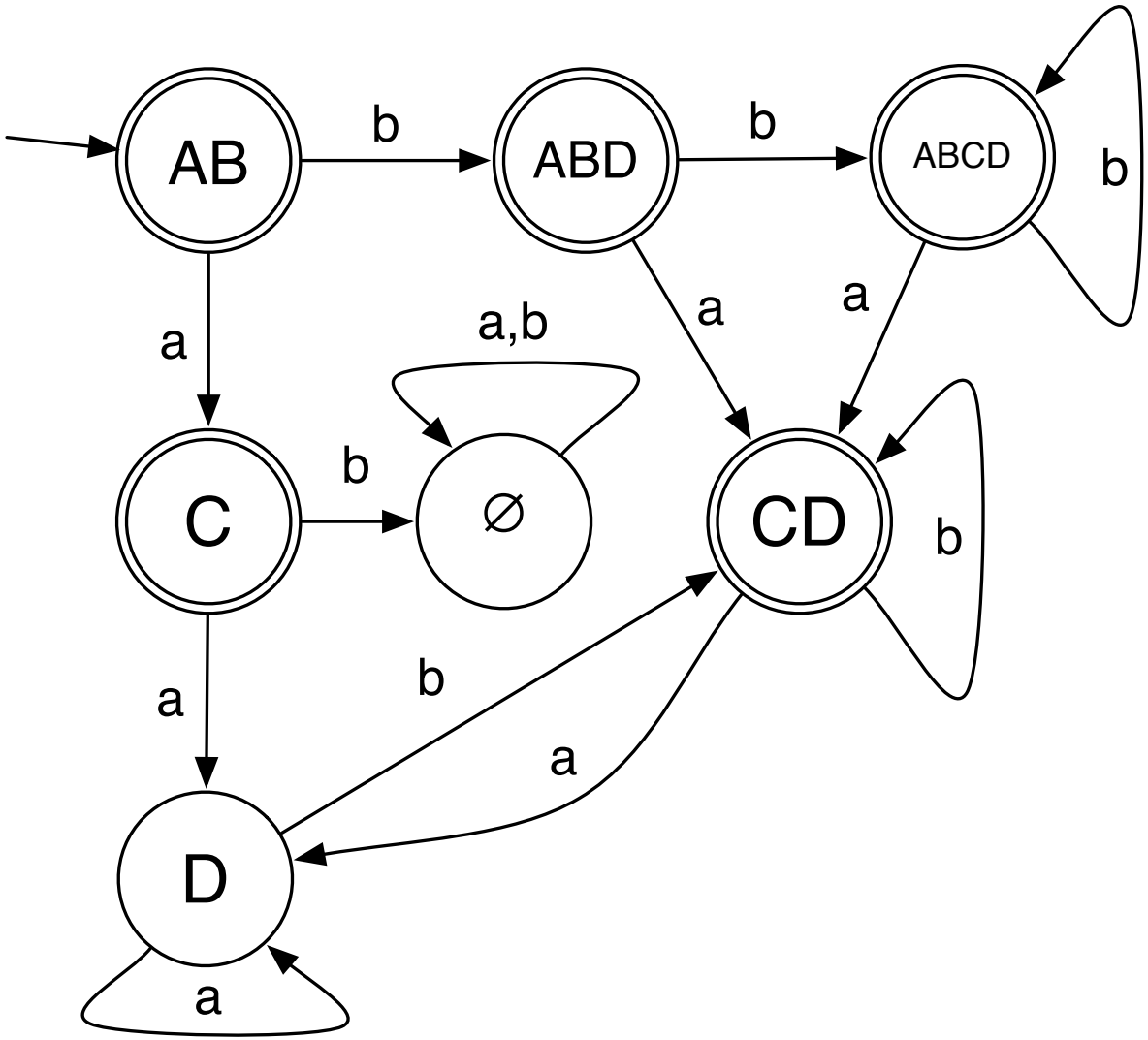
Examples

- For every finite maze there's a number p, such that
  - For every path through the maze s with $|s| \geq p$:
    - The path s contains at least one loop, which starts and ends within the first p steps.
    - There are infinitely many paths through the maze (at least one shorter, and arbitrarily many longer) whose lengths differ by a multiple of some constant.





Finite Automata as Mazes



A "Pumping" Lemma

If L is a regular language, then there exists a number p such that:

For every $s \in L$ with $|s| \geq p$:

There's at least one way to decompose s into xyz where
$|y| > 0$ and
$|xy| \leq p$ and
$xy^iz \in L$ for every $i \geq 0$.

Pumping Lemma Equivalents

- The Pumping Lemma tells us that

  If L is regular, then every string in L can be pumped.

- What logically follows?

  If every string in L can be pumped, then L is regular?

  If not every string in L can be pumped, then L is not regular!

  If there is at least one string in L that can't be pumped, L isn't regular!

Using the Pumping Lemma

- To prove a language L is not regular (¬i)
  - Suppose L were regular.
    - Carefully pick one long string $s \in L$.
    - Show that s cannot be pumped.
  - Contradiction. Therefore, L is not regular.

- You **cannot** use this Pumping Lemma to prove a language **is** regular!
  - For some non-regular languages, every string can be pumped!
    $\{ a^i b^j c^j \mid a \geq 1, j \geq 0 \} \cup \{ b^j c^k \mid j,k \geq 0 \}$ with p = 1

$L = \{ 0^n 1^n \mid n \geq 0 \}$

- Suppose L were regular.
  - Let p be the "pumping length" of L.
  - Pick $s := 0^p 1^p$. Since $|s| \geq p$, it should be pumpable.
  - Consider all possible decompositions
    $s = xyz$ with $y \neq \varepsilon$ and $|xy| \leq p$.
  - None of them work for pumping; contradiction.
- So L is not regular.

$L = \{ ww \mid w \in \{0,1\}^* \}$

- Suppose L were regular.
  - Let p be the "pumping length" of L.
  - Pick $s := 0^p 1 0^p 1$. Since $|s| \geq p$, it should be pumpable.
  - Consider all possible decompositions
    $s = xyz$ with $y \neq \varepsilon$ and $|xy| \leq p$.
  - None of them work for pumping; contradiction.
- So L is not regular.

$L = \{ w \in \Sigma^* \mid w \text{ a palindrome} \}$

- Suppose L were regular.
  - Let p be the "pumping length" of L.
  - Pick s := . Since $|s| \geq p$, it should be pumpable.
  - Consider all possible decompositions
    $s = xyz$ with $y \neq \varepsilon$ and $|xy| \leq p$.
  - None of them work for pumping; contradiction.
- So L is not regular.

$L = \{ 0^n 1^m \mid n \leq m \}$

- Suppose L were regular.
  - Let p be the "pumping length" of L.
  - Pick s := . Since $|s| \geq p$, it should be pumpable.
  - Consider all possible decompositions
    $s = xyz$ with $y \neq \varepsilon$ and $|xy| \leq p$.
  - None of them work for pumping; contradiction.
- So L is not regular.

$L = \{ 0^n 1^m \mid n \geq m \}$

- Suppose L were regular.
  - Let p be the "pumping length" of L.
  - Pick s := . Since $|s| \geq p$, it should be pumpable.
  - Consider all possible decompositions
    $s = xyz$ with $y \neq \varepsilon$ and $|xy| \leq p$.
  - None of them work for pumping; contradiction.
- So L is not regular.

Phrase-Structure Grammars

Language = Countable Set of Strings

- So far: recognition of strings in L
- New viewpoint: production of strings in L

Unrestricted Grammars

- An unrestricted grammar consists of

  1. A set V of **variables** (a.k.a. **nonterminals**)
  2. A disjoint set Σ (of **terminals**)
  3. A set of rules of the form LEFT $\rightarrow$ RIGHT where
     $\text{LEFT} \in (V \cup \Sigma)^+$ and $\text{RIGHT} \in (V \cup \Sigma)^*$
  4. One designated $S \in V$, called the **start variable** (a.k.a. **start symbol**)

Rewriting Strings

- If we have a rule LEFT $\rightarrow$ RIGHT, we can replace LEFT by RIGHT inside any string:

$$\alpha \text{LEFT} \beta \Rightarrow \alpha \text{RIGHT} \beta$$

- The language of the grammar is the set

$$\{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

Example

$$S \rightarrow aBSc$$

$$S \rightarrow \varepsilon$$

$$Ba \rightarrow aB$$

$$Bc \rightarrow bc$$

$$Bb \rightarrow bb$$

- What is the start symbol? What are the terminals and nonterminals?

Expressive Power

- If a program can generate a sequence of strings, the same set can be generated by an unrestricted grammar.

Restricted Grammars

- Chomsky Hierarchy
  - Type 0: Unrestricted Grammars
  - Type 1: Context-Sensitive Grammars
  - Type 2: Context-Free Grammars
  - Type 3: Regular Grammars

Type 0: Unrestricted

$$S \rightarrow aBSc$$

$$S \rightarrow \varepsilon$$

$$Ba \rightarrow aB$$

$$Bc \rightarrow bc$$

$$Bb \rightarrow bb$$

Type 0: Unrestricted

$S \rightarrow HaC$

$C \rightarrow aC$

$C \rightarrow aT$

$T \rightarrow ZU$

$AZ \rightarrow ZA$

$aZ \rightarrow ZAa$

$HZ \rightarrow HY$

$YA \rightarrow AY$

$Ya \rightarrow aY$

$YU \rightarrow T$

$T \rightarrow X$

$aX \rightarrow Xaa$

$AX \rightarrow Xa$

$HX \rightarrow \varepsilon$

Type 1: Context-Sensitive

$S \rightarrow abc$

$S \rightarrow aSQ$

$bQc \rightarrow bbcc$

$cQ \rightarrow Qc$

$S \rightarrow A_1BCS_1$

$S \rightarrow A_1BC$

$S_1 \rightarrow ABCS_1$

$S_1 \rightarrow ABC$

$BA \rightarrow AB$

$CA \rightarrow AC$

$CB \rightarrow BC$

$A_1 \rightarrow a$

$aA \rightarrow aa$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

Type 2: Context Free

$S \rightarrow aSb$

$S \rightarrow \varepsilon$

$S \rightarrow \varepsilon$

$S \rightarrow 0B$

$S \rightarrow 1A$

$A \rightarrow 0S$

$A \rightarrow 1AA$

$B \rightarrow 1S$

$B \rightarrow 0BB$

Type 3: Regular

$$S \rightarrow 1B$$

$$B \rightarrow 1B$$

$$B \rightarrow 0C$$

$$C \rightarrow 0S$$

$$S \rightarrow 0S$$

$$C \rightarrow 1B$$

$$C \rightarrow \varepsilon$$