CS 208: Computing Theory

Prof. Dr. Brahim Hnich
Faculty of Computer Sciences
Izmir University of Economics
Automata Theory

Regular Expressions
Regular expressions

- Recall: a language is a set of strings
- Regular expressions: A notation for building up languages
  - Example: \((0 \cup 1) \ 0^*\)
  - 0 and 1 are shorthand for \{0\} and \{1\}
  - So \((0 \cup 1) = \{0\} \cup \{1\} = \{0,1\}\)
  - \(0^*\) is \(\{0\}^* = \{\varepsilon, 0, 00, 000, 0000, 00000, 000000, \ldots\}\)
  - Concatenation (\(\circ\)), like multiplication is implicit
  - \((0 \cup 1) \ 0^*\) is the language of all strings starting with 0 or 1 followed by any number of 0’s
- Usually used in text editors or shell script
  - Lexical analyzer in any compiler
More examples

- Let $\Sigma$ be an alphabet
  - The regular expression $\Sigma$ is a language of one symbol strings
  - $\Sigma^*$ is all strings
  - $\Sigma^*1$ is all strings ending in 1
  - $0\Sigma^*U\Sigma^*1$ is all strings starting in 0 or ending in 1
Four operations
- Star (highest precedence)
- Concatenation
- Union (least precedence)
- Parenthesis used to change usual precedence
Formal definition: regular expression

- **Inductive definition**
  - R is a regular expression if R is
    - a for some a in \( \Sigma \)
    - \( \varepsilon \)
    - \( \emptyset \)
    - \( (R_1 \cup R_2) \) and R1 and R2 are regular expressions
    - \( (R_1 \circ R_2) \) and R1 and R2 are regular expressions
    - \( (R_1^*) \) and R1 is a regular expression
Let $L(R)$ denote the language defined by regular expression $R$

- $L(R)$ is defined as shown in Table

<table>
<thead>
<tr>
<th>R</th>
<th>L(R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>${\epsilon}$</td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td>$L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>$(R_1 \circ R_2)$</td>
<td>$L(R_1) \circ L(R_2)$</td>
</tr>
<tr>
<td>$(R_1)^*$</td>
<td>$L(R_1)^*$</td>
</tr>
</tbody>
</table>
Remarkable fact

- **Theorem**: A language is regular if and only if some regular expression describes it.

- This theorem states two things

  - If a language is described by a regular expression, then it is regular ($\leftarrow$)

  - If a language is regular, then it can be described by a regular expression ($\rightarrow$)
Proof (\(\leftarrow\))

- If a language is described by a regular expression, then it is regular (\(\leftarrow\))
  - Given a regular expression describing some language \(A\), we show how to convert \(R\) into a NFA recognizing \(A\)
  - By previous result: If an NFA recognizes \(A\), then \(A\) is regular
NFA accepting regular expression

- $R = a \in \Sigma$

Diagram:

- Start state $q_1$
- Transition on $a$ from $q_1$ to $q_2$
- $q_2$ is the accept state
NFA accepting regular expression

\[ R = \varepsilon \]
NFA accepting regular expression

\[ R = \emptyset \]
NFA accepting regular expression

\[ R = (R_1 \cup R_2) \]
NFA accepting regular expression

- $R = (R_1 \cup R_2)$
NFA accepting regular expression

- $R = (R_1 \circ R_2)$
NFA accepting regular expression

Let $R = (R_1 \circ R_2)$.
NFA accepting regular expression

\[ R = (R1)^* \]
NFA accepting regular expression

\[ R = (R1)^* \]
Example

\[ R = a \]
Example

\[ R = b \]
Example

$R = ab$
Example

\[ R = ab \cup a \]
Example

\[ R = (ab \cup a)^* \]
Nonregular languages

- We have made a lot of progress understanding what finite automata can do

- But, what can’t they do?
Limitations of finite automata

- \( B = \{0^n1^n \mid 0 \leq n\} \)
  - Examples: 01, 0011, 000111,....
  - Machines must “remember” how many 0s have been seen so far as it reads the input!
  - Impossible with finite automata
    - Because the number of 0s isn’t limited, the machine will have to keep track of an unlimited number of possibilities (states!)
Limitations of finite automata

\[ C = \{ w \mid w \text{ has an equal number of 0s and 1s} \} \]

- Examples: 01, 011100, 010101, ... 
- Machines must “remember” how many 0s and 1s have been seen so far as it reads the input!
- Impossible with finite automata
  - Because the number of 0s and 1s isn’t limited, the machine will have to keep track of an unlimited number of possibilities (states!)
Limitations of finite automata

- \( D = \{ w \mid w \) has an equal number of occurrences of 01 and 10 as substrings\} 
  - Examples: 00, 011100, 01010101010,....
  - Machines must “remember” how many 01s and 10s have been seen so far as it reads the input!
  - Impossible with finite automata
    - Because the number of 01s and 10s isn’t limited, the machine will have to keep track of an unlimited number of possibilities (states!)
Limitations of finite automata

- \( D = \{ w \mid w \text{ has an equal number of occurrences of 01 and 10 as substrings} \} \)
  - Examples: 00, 011100, 01010101010,....
  - Machines must “remember” how many 01s and 10s have been seen so far as it reads the input!
  - Impossible with finite automata
    - Because the number of 01s and 10s isn’t limited, the machine will have to keep track of an unlimited number of possibilities (states!)
Limitation of finite automata

- Our intuition may fail us (remember language D)
- So, how to prove that a certain language is not regular?
  - Pumping Lemma
Pumping lemma

- We will show that all regular languages have a special property
  - If a string is longer than a certain critical length \( l \), then it can be “pumped” to a larger length by repeating an internal substring

- This is a powerful technique for showing that a language is not regular!
Theorem: If $A$ is a regular language, then there is a number $p$ (the pumping length) where, if $s$ is any string of length at least $p$, then $s$ may be divided into three pieces, $s = xyz$, such that

1. For each $0 \leq i$, $xy^iz \in A$
2. $|y| > 0$
3. $|xy| \leq p$
Pumping lemma

- **Theorem**: If $A$ is a regular language, then there is a number $p$ (the pumping length) where, if $s$ is any string of length at least $p$, then $s$ may be divided into three pieces, $s = xyz$, such that

1. For each $0 \leq i$, $xyz^i \in A$ : note that $y^0$ is $\varepsilon$

2. $|y| > 0$: without this the theorem is trivially true!

3. $|xy| \leq p$ : $x$ and $y$ together should not have a length bigger than $p$
How to use the pumping lemma

- To show that language A is not regular

- Approach: Proof by contradiction
  - Assume A is regular in order to obtain a contradiction
  - Use the pumping lemma to guarantee the existence of pumping length $p$
  - Find a string $s$ of length $p$ or more that cannot be pumped
    - Demonstrate that $s$ cannot be pumped by considering all possible ways of dividing $s$ into $x$, $y$, and $z$, and for each division find an $i$ where $xyz$ does not belong to A
Prove that $B = \{0^n1^n \mid 0 \leq n\}$ is not regular

- Assume $B$ is regular
- Because of pumping lemma, we have the pumping length $p$
- Now consider string $s$ to be $0^p1^p$
- Theorem says $s = xyz$, where $xy^i z$ belongs to $B$
  - $y$ is all 0s in $s$, then we have too many 0’s in $xy^i z$
  - $y$ is all 1s in $s$, then we have too many 1’s in $xy^i z$
  - $y$ is mixed of 0s followed by 1s in $s$, then $xy^i z$ will have 0s and 1s out of order
Another Application

- Prove that $C = \{w \mid w \text{ has an equal number of 0s and 1s} \}$ is not regular
  - Assume $C$ is regular
  - Because of pumping lemma, we have the pumping length $p$
  - Now consider string $s$ to be $0^p 1^p$
  - Theorem says $s = xyz$, where $xy^i z$ belongs to $C$
    - $y$ is all 0s in $s$, then we have too many 0’s in $xy^i z$
    - $y$ is all 1s in $s$, then we have too many 1’s in $xy^i z$
    - $y$ is mixed of 0s and 1s in $s$, then by condition 3, we have $|xy|$ is less than or equal to $p$, so $y$ will have only 0s
Conclusions

Regular expression
  closure properties
  union
  concatenation
  star

Non-regular languages
  pumping lemma

Next: Context-free grammars