Regular Expressions

Regular Expressions

- Aka **regex**, **regexp**, **rational expression**
  - Sequence of characters that define a search pattern
  - Usually used to find operations on strings or for input validation

- Use previously described regular operations to build up expressions describing languages
  - The output value of a regular expression is a language
    - $(0 \cup 1)0^* =$
      - language consisting of all strings starting with a 0 or a 1
      - followed by any number of zeros

Regular Expressions Formal Definition

- **Formal Definition** for Regular Expressions
- R is a regular expression over alphabet Σ if R is one of the following:
  1. a = any symbol in an alphabet Σ
  2. ε = any empty string
  3. $\varnothing$ = empty set i.e., empty language
  4. $(R_1 \cup R_2) =$ Union
     - R_1 and R_2 are <u>smaller</u> regular expressions
  5. $(R_1 \circ R_2) =$ Concatenation
  6. $(R_1^*) =$ Star Operation

- **Order of Precedence**
  - * (star) highest
  - Then ○ (concatenation)
  - U (union)

Languages from Regular Expressions

- Procedure for denoting a regular language from a given regular expression
  - Simplify expressions
  - Star operations provide all possible combinations of elements including the empty set
  - Identify any substrings that cannot be removed

- Example 1
  - Given Regular Expression: $((0 \cup 1)\varepsilon)^* \cup 0)$
  - Denotes language $\{0,1\}^* \cup \{0\} = \{0,1\}^* =$ All Strings

- Example 2
  - Given Regular Expression: $(0 \cup 1)^* 111(0 \cup 1)^*$
  - Denotes language $\{0,1\}^* \{111\}\{0,1\}^* =$ All strings with substring 111

Regular Expressions from Language

- Procedure for specifying a regular expression from a given regular language
  - Identify required substring
  - Place in between star strings
    - Star strings must not negate a constraint of the language
    - Special notation $R^+ = R \circ R^*$, $R^+ \cup \varepsilon = R^*$

- Example 1
  - Given language $L = strings\ over\ \{0,1\}\ with\ odd\ number\ of\ 1s$
  - Associated Regular Expression: $0^*10^*(0^*10^*10^*)^*$

- Example 2
  - Given language $L = strings\ with\ substring\ 01\ or\ 10$
  - Associated Regular Expression: $(0 \cup 1)^*01(0 \cup 1)^* \cup (0 \cup 1)^*10(0 \cup 1)^*$
  - Abbreviated Regular Expression: $\Sigma^*01\Sigma^* \cup \Sigma^*10\Sigma^*$

No Complements

- Previous Example
  - Given language $L = \mathit{strings\ with\ substring}\ 01\ \mathit{or}\ 10$
  - Associated Regular Expression: $(0 \cup 1)^{*}\ 01\ (0 \cup 1)^{*} \cup (0 \cup 1)^{*}\ 10\ (0 \cup 1)^{*}$
  - Abbreviated Regular Expression: $\Sigma^{*}\ 01\ \Sigma^{*} \cup \Sigma^{*}\ 10\ \Sigma^{*}$

- Example 1
  - Given language $L = \mathit{strings\ with\ neither\ substring}\ 01\ \mathit{or}\ 10$
  - Can't perform a simple complement operation, must write out expression
    - Strings that are all 0's or 1's
  - Associated Regular Expression: $0^{*} \cup 1^{*}$

- Example 2
  - Given language $L = \mathit{strings\ with\ no\ more\ than\ two\ consecutive}\ 0s\ \mathit{or}\ 1s$
  - Would be easy if we could write a complement but can't
    - Must write out expression: Alternate one or two of 0's or 1's
  - Associated Regular Expression: $(\varepsilon \cup 1 \cup 11)(\ (0 \cup 00)(1 \cup 11))^{*}\ (\varepsilon \cup 0 \cup 00)$

Uses for Regular Expressions

- Regular expressions commonly used to specify syntax
  - For (portions of) programming languages
  - Editors
  - Command languages like UNIX shell

- Example: Decimal Numbers

$$DD^*.D^* \cup D^*.DD^*$$

    - Where D is the alphabet {0,1, … , 9}
  - Need a digit either before or after the decimal point

Languages Denoted by Regular Expressions

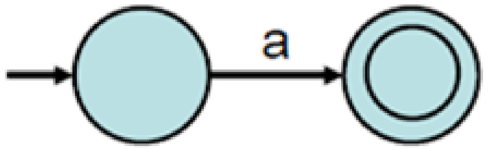
- If a language can be expressed by a regular expression, it is a regular (FA-recognizable) language.

- Regular expressions will have an equivalent finite automata.
  - **Kleene's Theorem**

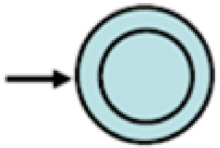
Proof Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language recognized by a finite automata
  - Theorem allows us to convert R to a finite automata

- Proof
  - For each R, define an NFA M with L(M)=L(R)
  - Proceed by induction on the structure of R (formal definition):
    - Show for the three base cases (a, ε, ∅)
    - Show how to construct NFAs for more complex expressions from NFAs for their subexpressions
  - Case 1: R = a
    - L(R) = {a}, accepts only a

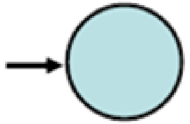


  - Case 2: R = ε
    - L(R) = {ε}, accepts only ε

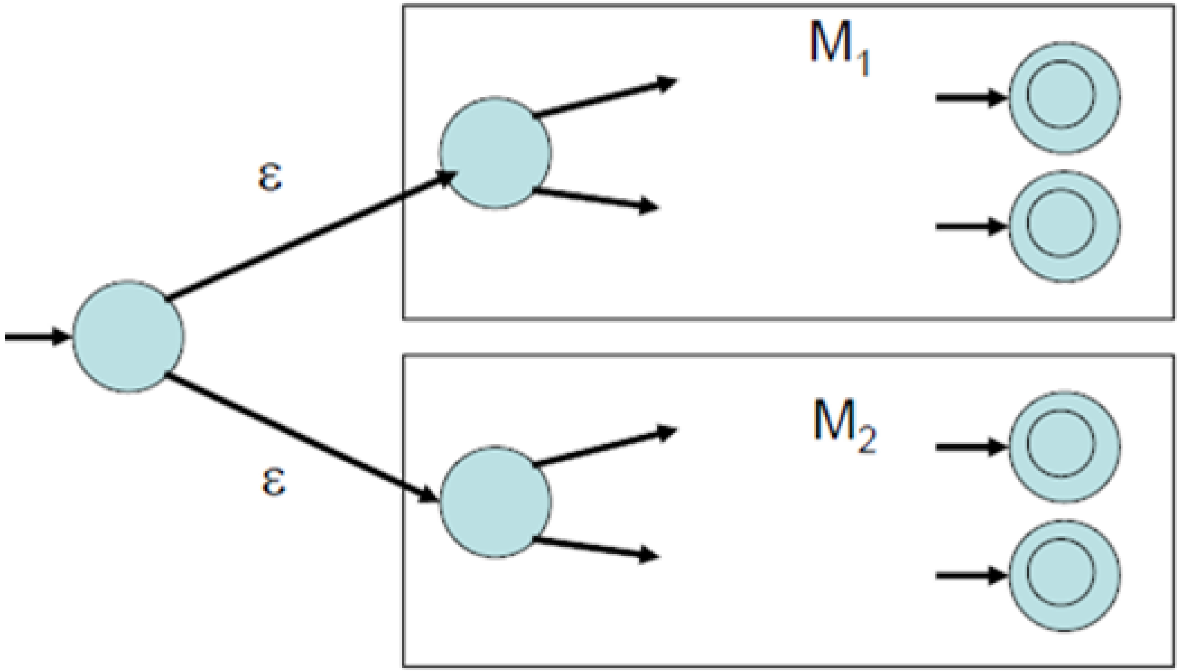


Proof Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language recognized by a finite automata
- Proof
  - Case 3: $R = \emptyset$
    - $L(R) = \emptyset$, accepts nothing



  - Case 4: $R = R_1 \cup R_2$
    - M_1 recognizes $L(R_1)$
    - M_2 recognizes $L(R_2)$

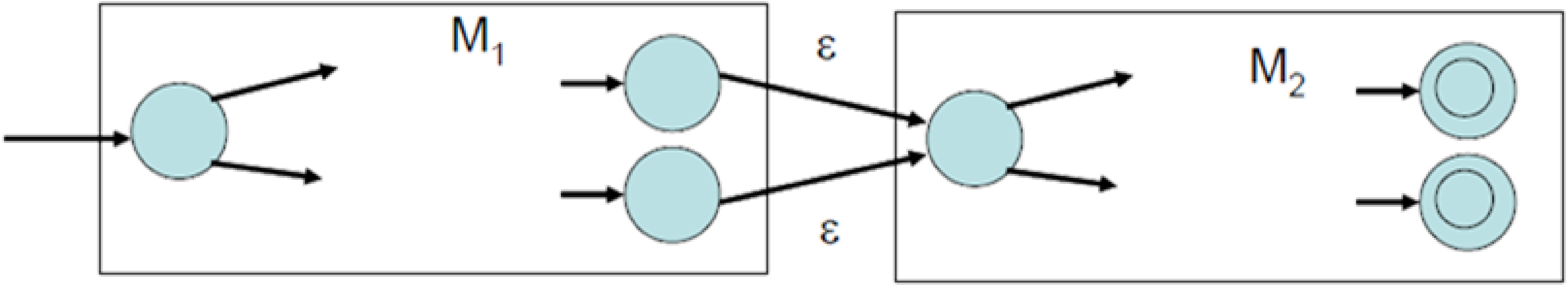


    - Same construction we used to show regular languages are closed under union

Proof Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language recognized by a finite automata

- Proof
  - Case 5: $R = R_1 \circ R_2$
    - M_1 recognizes $L(R_1)$
    - M_2 recognizes $L(R_2)$

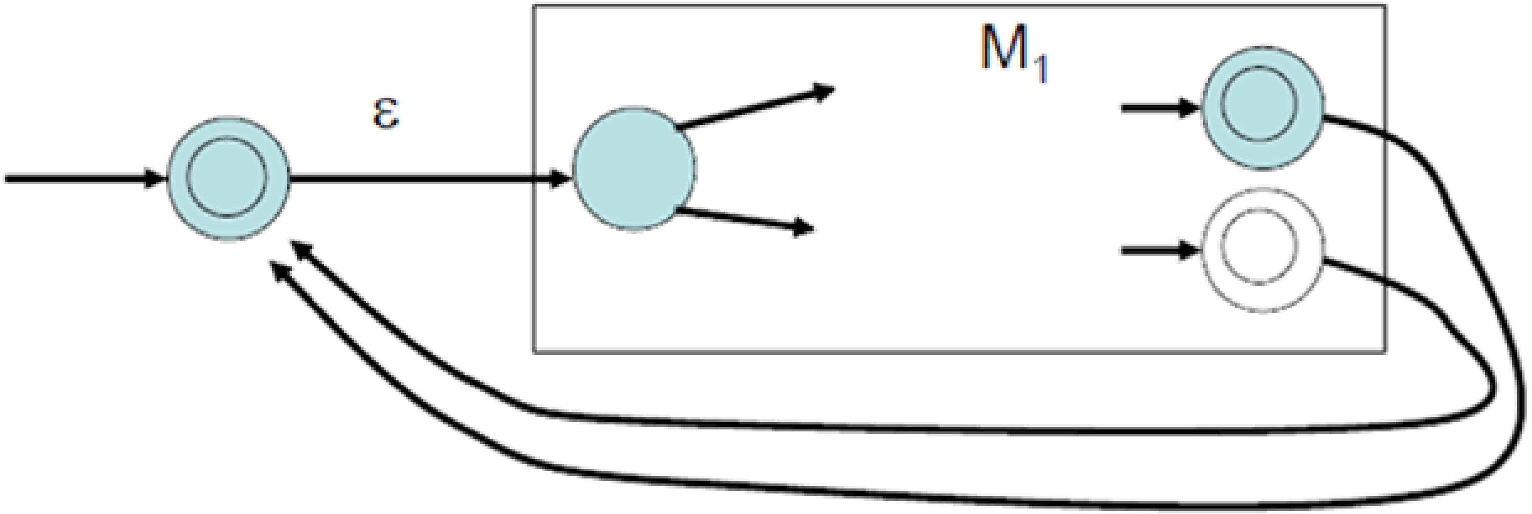


    - Same construction we used to show regular languages are closed under star

Proof Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language recognized by a finite automata

- Proof
  - Case 6: $R = (R_1)^*$
    - M_1 recognizes $L(R_1)$



    - Same construction we used to show regular languages are closed under star