Example 1.41: Convert to DFA

Transition table of NFA

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

DFA table (no ε)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>R(-)=-</td>
<td>R(2)=2</td>
</tr>
<tr>
<td>2</td>
<td>R(2)=23</td>
<td>R(3)=3</td>
</tr>
<tr>
<td>3</td>
<td>R(1)=13</td>
<td>R(-)=-</td>
</tr>
<tr>
<td>12</td>
<td>R(2)=23</td>
<td>R(23)=23</td>
</tr>
<tr>
<td>13</td>
<td>R(1)=13</td>
<td>R(2)=2</td>
</tr>
<tr>
<td>23</td>
<td>R(123)=123</td>
<td>R(3)=3</td>
</tr>
<tr>
<td>123</td>
<td>R(123)=123</td>
<td>R(23)=23</td>
</tr>
<tr>
<td>-</td>
<td>R(-)=</td>
<td>R(-)=</td>
</tr>
</tbody>
</table>

Epsilon Table

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

DFA Table

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R(-)=-</td>
<td>R(2)=2</td>
</tr>
<tr>
<td>2</td>
<td>R(2)=23</td>
<td>R(3)=3</td>
</tr>
<tr>
<td>3</td>
<td>R(1)=13</td>
<td>R(-)=-</td>
</tr>
<tr>
<td>12</td>
<td>R(2)=23</td>
<td>R(23)=23</td>
</tr>
<tr>
<td>13</td>
<td>R(1)=13</td>
<td>R(2)=2</td>
</tr>
<tr>
<td>23</td>
<td>R(123)=123</td>
<td>R(3)=3</td>
</tr>
<tr>
<td>123</td>
<td>R(123)=123</td>
<td>R(23)=23</td>
</tr>
<tr>
<td>-</td>
<td>R(-)=</td>
<td>R(-)=</td>
</tr>
</tbody>
</table>
Theorem 1.45: Regular languages are closed under union

- Prove this with simpler construction than we did with DFAs
- Proof Idea: We can use NFAs
  - Let $A_1$ and $A_2$ be regular languages
  - So there exists NFAs $N_1$ and $N_2$ such that $L(N_1) = A_1$ and $L(N_2) = A_2$
  - Construct $N$ so that it includes the states and transitions of $N_1$ and $N_2$
    + Add new start state and epsilon transition to start states of $N_1$ and $N_2$
    + Accept states is the union of accept states of $N_1$ and $N_2$

- Note that NFA is not useful for proving intersection
More Closure Properties

**Theorem 1.47:** The class of regular languages is closed under the concatenation operation.
- $A \cdot B = \{xy | x \in A \text{ and } y \in B\}$
- Proof Idea:

**Theorem 1.49:** The class of regular languages is closed under the star operation.
- $A^* = \{x_1x_2...x_k | k \geq 0 \text{ and each } x_i \in A\}$
- Proof Idea:

Overview

- Nondeterminism (Continued)
  $\Rightarrow$ Regular Expressions
Regular Expressions

- In arithmetic, we can use + and \( \times \) to build up expressions
  - Example: \((5 + 3) \times 4\)
  - This has a value of 32

- Similarly, can use operations to build up expressions describing languages
  - Example: \((0 \cup 1)0^*\)
  - What is the value of this?
  - It is a language (set of strings)
    - Either a 0 or a 1 followed by any number of 0's
    - How?

Meaning of Regular Expressions (informally)

- What is language of \((0 \cup 1)0^*\)?
  - Symbols 0 and 1 are shorthand for the sets \{0\} and \{1\}
    - So, \((0 \cup 1)\) means \{\{0\} \cup \{1\}\}, which is the language \{0, 1\}
  - The part \(0^*\) means \{0\}^*
    - This is the star operation to the set containing just 0
    - \{0\}^* is set of all strings containing any number of 0s
  - Between \((0 \cup 1)\) and \(0^*\) there is an implicit concatenation operation \(\circ\)
    - Similar to how \(\times\) in arithmetic can be implicit (i.e. \(3y\))
  - Concatenation takes all combinations of a string from first language with a string from second language
Another Example

• \((0 \cup 1)^*\)
  - Starts with the language \((0 \cup 1)\) and applies the \(^*\) operation
  - Any string of 0’s and 1’s
  - If \(\Sigma = \{0, 1\}\), we can write \(\Sigma\) as shorthand for \((0 \cup 1)\)

Formal Definition

**Def. 1.52:** We say that \(R\) is a **regular expression** if \(R\) is
1. \(a\) for some \(a\) in the alphabet \(\Sigma\)
2. \(\epsilon\)
3. \(\emptyset\)
4. \((R_1 \cup R_2)\), where \(R_1\) and \(R_2\) are regular expressions
5. \((R_1 \circ R_2)\), where \(R_1\) and \(R_2\) are regular expressions
6. \((R_1^*)\), where \(R_1\) is a regular expression.

• ‘Computation’
  - Items 1 and 2 represent the languages \(\{a\}\) and \(\{\epsilon\}\), respectively
  - Item 3 is the empty language
  - Item 4, 5, 6 represent the languages obtained by taking the union or
    concatenation of \(R_1\) and \(R_2\) or the star of \(R_1\), respectively
Notes

• Difference between $\epsilon$ and $\emptyset$?
• Circular definition?
  - Inductive definition
• Parentheses can be omitted
  - Just like in arithmetic. Precedence order is star, concatenation, union
• $R^+$ shortform for $RR^*$
• Use $L(R)$ to refer to language of a regular expression

Examples

• $0^*10^*$
• $\Sigma^*1\Sigma^*$
• $\Sigma^*001\Sigma^*$
• $1^*(01^*)^*$
• $(\Sigma\Sigma)^*$
• $01 \cup 10$
• $1^*\emptyset$
• $\emptyset^*$
Identities

- \( R \cup \emptyset \)
- \( R \circ \epsilon \)
- What about?
  - \( R \circ \emptyset \)
  - \( R \cup \epsilon \)

Equivalence with Finite Automata

**Theorem 1.54:** A language is regular iff some regular expression describes it.

- In other words, a language can be recognized by a NFA iff it is the language of a regular expression
### Regular Expression → Regular Language

- **Proof Idea:** Build a NFA to recognize the language.
- **Make a NFA for each of the 6 cases from the formal definition of regular expression**:
  - \( R = a \)
  - \( R = \varepsilon \)
  - \( R = R_1 \cup R_2 \)
  - \( R_1 \circ R_2 \)
  - \( R^* \)

---

### Example

- \( a \)
- \( b \)
- \( ab \)
- \( ab \cup a \)
- \( (ab \cup a)^* \)
Regular Language → Regular Expression

• Proof idea: convert DFA into a regular expression
• Generalized nondeterministic finite automaton (GNFA)
  - Like an NFA, but can have regular expressions on its transitions
  - Will convert an DFA into a GNFA, and then into GNFA with
    successively one less state, by using more and more complex regular
    expressions on its arcs
• We will use a GNFA with the following additional restrictions:
  - start state has transitions to every other state, but no arrows coming in
  - single accept state, with arrows coming in from every other state, but no
    arrows going out. And, not the same as the start state
  - except start and accept state, one arrow goes from every state to every
    other state and from each state to itself
  - Restricted version of GNFA makes the proof simpler

Proof

• Easy to convert DFA into GNFA
  - Make a new start state with epsilon transition to old start state
  - Make a new accept state, with epsilon transitions from old accept states
  - Any arc with multiple labels, use union instead
  - Add arrows labeled $\emptyset$ where needed
• Base Case: just has 2 states
  - One will be start, and one will be accept
  - No self-loops (aren’t allowed on start and accept)
  - Just one transition
    - Its regular expression is the regular expression of the NFA
Induction

- GNFA (in restricted form) with more than 2 states
- Proof Idea:
  - Remove a state that is not the start nor accept state
  - Repair the labels on the remaining transitions
- Proof Idea in more detail:
  - Remove state $q_{rip}$
  - Alter label from $q_i$ to $q_j$
    + Say $R_1$ was label from $q_i$ to $q_j$
    + Say $R_1$ is label from $q_i$ to $q_{rip}$
    + Say $R_2$ is self-loop label on $q_{rip}$
    + Say $R_3$ is label from $q_{rip}$ to $q_j$
  - New label is $(R_1)(R_2)^*(R_3) \cup (R_4)$

Example: Convert into a Regular Expression

```
1 \[ \xrightarrow{a} \] 2
\[ \xrightarrow{b} \]
```

```
1 \[ \xrightarrow{a} \] 2
\[ \xrightarrow{a, b} \]
```