Regular Expressions

Example: \((a \mid b)^*\)

\[
\begin{align*}
L_0 &= \{\epsilon\} \\
L_1 &= L_0 \cdot \{a, b\} \\
&= \{\epsilon\} \cdot \{a, b\} \\
&= \{a, b\} \\
L_2 &= L_1 \cdot \{a, b\} \\
&= \{a, b\} \cdot \{a, b\} \\
&= \{aa, ab, ba, bb\} \\
L_3 &= L_2 \cdot \{a, b\} \\
&= \cdots
\end{align*}
\]

\[
L = \bigcup_{i=0}^{\infty} L_i = \{\epsilon, a, b, aa, ab, ba, bb, \ldots\}
\]

Language of Regular Expressions

Let \(R\) be the set of all regular expressions over \(\Sigma\). Then,

- **Empty String**: \(\epsilon \in R\)
- **Unit String**: \(\alpha \in \Sigma \Rightarrow \alpha \in R\)
- **Concatenation**: \(r_1, r_2 \in R \Rightarrow r_1 r_2 \in R\)
- **Alternative**: \(r_1, r_2 \in R \Rightarrow (r_1 \mid r_2) \in R\)
- **Kleene Closure**: \(r \in R \Rightarrow r^* \in R\)

Regular Expressions

- \(a\): stands for the set \(\{a\}\) that contains a single string \(a\).
- \(a \mid b\): stands for the set \(\{a, b\}\) that contains two strings \(a\) and \(b\).
  - Analogous to **Union**.
- \(ab\): stands for the set \(\{ab\}\) that contains a single string \(ab\).
  - Analogous to **Product**.
- \([ab][a][b]:\ stands for the set \(\{aa, ab, ba, bb\}\).
- \(a^*\): stands for the set \(\{\epsilon, a, aa, aaa, \ldots\}\) that contains all strings of zero or more \(a\)'s.
  - Analogous to closure of the product operation.
Computing the Semantics of Closure

Example: \( L((a \mid b)^*) \)

\[
L_0 = \{\epsilon\} \quad \text{Base case}
\]

\[
L_1 = \{\epsilon\} \cup \{(a, b) \cdot L_0\}
= \{\epsilon\} \cup \{(a, b) \cdot \{\epsilon\}\}
= \{\epsilon, a, b\}
\]

\[
L_2 = \{\epsilon\} \cup \{(a, b) \cdot L_1\}
= \{\epsilon\} \cup \{(a, b) \cdot \{\epsilon, a, b\}\}
= \{\epsilon, a, b, aa, ab, ba, bb\}
\]

\[
L((a \mid b)^*) = L_{\infty} = \{\epsilon, a, b, aa, ab, ba, bb, \ldots\}
\]

Another Example

\[
L((a^*b^*)^*):
\]

\[
L(a^*) = \{\epsilon, a, aa, \ldots\}
\]

\[
L(b^*) = \{\epsilon, b, bb, \ldots\}
\]

\[
L((a^*b^*)^*) = \{\epsilon\}
\cup \{\epsilon, a, b, aa, ab, bb, \ldots\}
\]

Semantics of Regular Expressions

\textit{Semantic Function} \( L \): Maps regular expressions to sets of strings.

\[
L(\epsilon) = \{\epsilon\}
\]

\[
L(\alpha) = \{\alpha\} \quad (\alpha \in \Sigma)
\]

\[
L(r_1 \mid r_2) = L(r_1) \cup L(r_2)
\]

\[
L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)
\]

\[
L(r^*) = \{\epsilon\} \cup (L(r) \cdot L(r^*))
\]
Using Regular Definitions for Lexical Analysis

Q: Is $abababbb$ in $L((a^*b^*)^*)$?

A: Hm, well, let's see.

$L((a^*b^*)^*) = \{ \epsilon \} \cup \{ a, b, aa, ab, bb \} \cup \{ a, a, b, aa, ba, bb \} \cup \{ a, a, b, ab, b, bbb, bb \} \cdots \}$

= ???
Finite State Automata

Represented by a labeled directed graph,

- A finite set of states (vertices),
- Transitions between states (edges),
- Labels on transitions are drawn from $\Sigma \cup \{\epsilon\}$,
- One distinguished start state,
- One or more distinguished final states.

Finite State Automata: An Example

Consider the Regular Expression $(a \mid b)^*a(a \mid b)$.

$L((a \mid b)^*a(a \mid b)) = \{aa, ab, aab, baa, bab, aaaa, aaab, abaa, abab, baaa, \ldots\}$.

The following automaton determines whether an input string belongs to $L((a \mid b)^*a(a \mid b))$:

```
    1
      v
    a - a
    2 - 3
    b - b
```

Recognizing Finite Sets of Strings

Identifying words from a small, finite, fixed vocabulary is straightforward.

For instance, consider a stack machine with push, pop, and add operations with two constants: 0 and 1.

We can use the automaton:

```
    0
    p - s
    a - d
    b - p
    push
    pop
    add
```
Recognition with an NFA

Is $abab \in \mathcal{L}((a \mid b)^*a(a \mid b))$?

Input: a b a b
Path 1: 1 1 1 1
Path 2: 1 1 1 2 3 Accept
Path 3: 1 2 3 ⊥ ⊥

Accept

Determinism

$(a \mid b)^*a(a \mid b)$:

Non-deterministic:
(NFA)

Deterministic:
(DFA)

Acceptance Criterion

A finite state automaton (NFA or DFA) accepts an input string $x$ if beginning from the start state:

... we can trace some path through the automaton
... such that the sequence of edge labels spells $x$
... and end in a final state,