Example NFA and Equivalent DFA

Example

\[ \text{L} = \{w \text{ over } \{a_1, a_2, a_3\} | \text{w does not contain } a_i \text{ for some } i\} \]
Regular Expressions & Languages

Regular expressions are a formalism for describing patterns in strings *(used in tools like Emacs, Perl,...)*

Language of a regular expression is the set of strings described

\[
\begin{align*}
    a & \in \Sigma & \{a\} \\
    \varepsilon & & \{\varepsilon\} \\
    \phi & & \phi \\
    (R_1 \cup R_2) & (R_1, R_2 \text{ reg. exp}) & L(R_1) \cup L(R_2) \\
    (R_1 \circ R_2) & & L(R_1) \circ L(R_2) \\
    (R_1)^* & & (L(R_1))^*
\end{align*}
\]

What kind of definition is this?

---

Regular Expressions, Examples over \{a,b\}

\[
\begin{align*}
    (a \cup b) & & (a)^* & & (a \cup b)^*
\end{align*}
\]

All these parentheses are clumsy–can omit but then must know what order to carry out operations

\[
\begin{align*}
    a \cup b^* & & ??? \\
\end{align*}
\]

Precedence of operations:

\[
\begin{align*}
    * & \circ \\
    (\text{associate to left}) & \cup
\end{align*}
\]

but ( ) must always be followed if included!

Shorthand: use \( \Sigma \) to stand alphabet, ex. \( a \cup b \)

Omit \( \circ \) where understood

\[
\begin{align*}
    (\Sigma \Sigma \Sigma)^* & & \Sigma^* bb \Sigma^* & & a^* b & & a^* \phi & & a^* \varepsilon
\end{align*}
\]
Lexical Analysis (Scanning) in Compilers

First phase of the compiler: strings \(\rightarrow\) tokens
Programming-language dependent

Tokens can include
- keywords if then end
- operators < > =
- identifiers
- literals decimal integer, character

One regular expression to describe each of the token types
Qu: How recognize tokens?
    Can use NFA or DFA

What is the Regular Expression?

A Java identifier must start with a letter, underscore, or $; subsequent characters can also contain digits (Letters are A-Z, a-z, digits are 0-9)

A Java decimal integer literal consists of a sequence of digits, without a leading zero, and with an optional L or l (indicating it is a long integer)
    It must have at least one digit.

A Java floating point literal can have the following parts:
    a decimal integer literal, a decimal point, a fraction
    (decimal integer literal), an exponent (E or e followed by an optional sign and a decimal integer)
    It must have either . or an exponent.