Languages and Computation (COMP 2049) Lab 02

Grammars and Finite Automata

(1) Assume that we are designing a programming language, and the floating-point numbers in the language must be formed according to the following rules:

- Each number may be signed or unsigned.
  - unsigned as in `3.14`, signed as in `+3.14` or `-3.14`;
- The numerical part (also called the value field) must start with a non-empty sequence of digits.
  - For instance, in the number `+322.432`, the value field is `322.432`, which starts with the sequence of digits `322`.
- The value field may optionally include a decimal point '.', in which case it must be followed by some other digits;
  - `3` and `3.14` are acceptable, but `3.` is not acceptable.
- There may be an optional exponent field, in which case, it must contain the letter 'e', followed by a (signed or unsigned) integer.
  - For instance, `3.14e+38` or `-1.2e24` are acceptable, but `7.17e` and `7.17e-` are not acceptable.

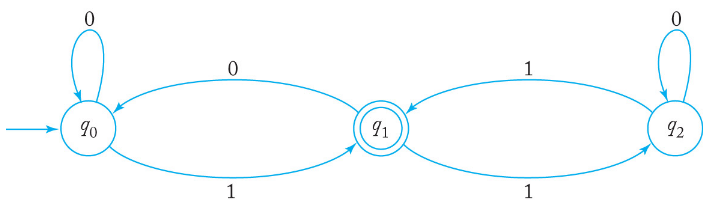
Design a grammar that generates these numbers.

Solution

The following is one possible grammar, in which $\langle \text{ number } \rangle$ is the start symbol:

$$
\begin{aligned}
\langle \text{ number } \rangle &\to \langle \text{ sign } \rangle \langle \text{ digits } \rangle \langle \text{ rest } \rangle \\
\langle \text{ sign } \rangle &\to + \mid - \mid \lambda \\
\langle \text{ digits } \rangle &\to \langle \text{ digit } \rangle \langle \text{ digits } \rangle \mid \langle \text{ digit } \rangle \\
\langle \text{ rest } \rangle &\to \langle \text{ exponent } \rangle \mid . \langle \text{ frac } \rangle \\
\langle \text{ frac } \rangle &\to \langle \text{ digits } \rangle \langle \text{ exponent } \rangle \\
\langle \text{ exponent } \rangle &\to \lambda \mid \mathrm{e} \langle \text{ sign } \rangle \langle \text{ digits } \rangle \\
\langle \text{ digit } \rangle &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

(2) Give a description of the language accepted by the following deterministic finite automaton (DFA):



Solution

Take the alphabet $\Sigma = \{0, 1\}$ and let $L \subseteq \Sigma^*$ be the language accepted by the DFA above. As the empty string is not accepted, then every string in L must be non-empty. So, assume that $w \in \Sigma^* \setminus \{\lambda\}$. For some $k \geq 1$, we may write $w = a_1 \ldots a_k$, in which each a_i is in Σ. Let $j \geq 0$ be the index of the last 0 in the string w. If w has no occurrences of 0, then we let $j = 0$. Then, we have:

$$w \in L \iff k - j \text{ is an odd number.}$$

In simple terms, $w \in L$ if and only if it ends with an odd number of 1's. For example:

- `001` and `111` are both in L;
- `011` and `110` are not in L.

(3) Assume that $\Sigma = \{a, b\}$. Show that the following language is regular by drawing a DFA that accepts it:

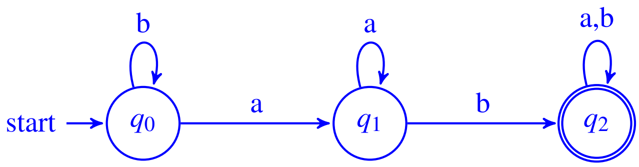
$$L = \{w \in \Sigma^* \mid w \text{ does not contain the substring } ab\}.$$

Solution

As the language is quite simple, the DFA may be designed directly. We take a different approach which may be useful in similar cases. We first constructs a DFA that accepts the complement $\overline{L}$ of L, i. e., the DFA that accepts:

$$\overline{L} = \{w \in \Sigma^* \mid w \text{ contains the substring } ab\}.$$

The following is a DFA for $\overline{L}$:



Next, we turn every final state of the above DFA into a non-final one, and vice versa. The resulting DFA will accept L: