Q: Is it sufficient for $S = V \backslash A$?

A: No! Our set F of finishing/accepting states should be nonempty. So we add an element $\{f\}$ to $V \backslash A$, where our acceptor will end up when a word in our language. Thus, $S = (V \backslash A) \cup \{f\}$ and $F = \{f\}$. $F \subseteq S$ as needed.

Q: How do we define t?

A: Use the production rules in P! For every rule of type (i), which is of the form <A>→ b<B> set t(<A>,b) =<B>. This works out well because our nonterminals <A> and <B> are states of the acceptor and the terminal $b \in A$ so t takes an element of $S \times A$ to an element of S as needed. Now look at production rules of type (ii), <A>→ b and of types (iii), <A>→ ε. Those are applied when we finish constructing a word w in our language L, **i.e.** at the very last step, so our acceptor should end up in the finishing state f whenever a production rule of type (ii) or (iii) is applied. Write a production rule of type (ii) or (iii) as <A>$\rightarrow w$, then we can set $t(<\mathrm{A}>, w) = f$. We have finished constructing t as well. Technically, $t : S \times (A \cup \{\varepsilon\}) \rightarrow S$ instead of $t : S \times A \rightarrow S$, but we can easily fix the transition function t by combining the last two transitions for each accepted word.

Remark: The same general principles as we used above allow us to go from a finite state acceptor to a regular grammar. This gives us the following theorem:

Theorem: A language L is regular $\Leftrightarrow L$ is recognised by a finite state acceptor $\Leftrightarrow L$ is generated by a regular grammar.

8.5 Regular expressions

Task: Understand another equivalent way of characterizing regular languages due to Kleene in the 1950's.

Definition: Let A be an alphabet.

1. $\emptyset$, ϵ, and all elements of A are regular expressions;
2. If w and w' are regular expressions, then $w \circ w'$, $w \cup w'$, and w^* are regular expressions.

Remark: This definition is an inductive one.

NB It is important not to confuse the regular expressions $\emptyset$ and ϵ. The expression ϵ represents the language consisting of a single string, namely ϵ, the empty string, whereas $\emptyset$ represents the language that does not contain any strings. Recall that a language L is any subset of

$$A^* = \bigcup_{n=0}^{\infty} A^n = A^0 \cup A^1 \cup A^2 \cup \cdots,$$

where $A^0 = \{\epsilon\}$, the set of words of length 0, $A^1 =$ the set of words of length 1, and $A^2 =$ the set of words of length 2.

Precedence order of operations if parentheses are not present:

First $*$, then $\circ$ (concatenation), then $\cup$ (union).

Examples: (1) $A = \{0, 1\}$

$$\begin{aligned} 1^* \circ 0 &= \{w \in A^* \mid w = 1^m 0 \text{ for } m \in \mathbb{N}, m \geq 0\} = \{0, 10, 110, 1110, \ldots\} \\ &= 1^*0. \end{aligned}$$

We can omit the concatenation symbol.

(2) $A = \{0, 1\}$

$$\begin{aligned} A^* \circ 1 \circ A^* &= \{w \in A^* \mid w \text{ contains at least one } 1\} \\ &= \{u \circ 1 \circ v \mid u, v \in A^*\} = A^*1A^* \end{aligned}$$

(3) $A = \{0, 1\}$

$$(A \circ A)^* = \{w \in A^* \mid w \text{ is a word of even length}\}.$$

Recall that $L^* = \bigcup_{n=0}^{\infty} L^n$, where $L^0 = \{\epsilon\}$, $L^1 = L$, and inductively $L^n = L \circ L^{n-1}$. Here $L = \{00, 01, 10, 11\}$.

(3') $(A^* \circ A^*)^* = A$.

(4) $A = \{0, 1\}$ $\quad (0 \cup \epsilon) \circ (1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.

(5) $\epsilon^* = \{\epsilon\}$.

(6) $\emptyset^* = \{\epsilon\}$. The star operation concatenates any number of words from the language. If the language is empty, then the star operation can only put together 0 words, which yields only the empty word.