NFA to Regular Expression – an Alternative

There are several ways to obtain regular expressions for finite automata that preserve the generated languages. This note describes one particular technique that is easily executed and can yield rather nice expressions.

First, let us note some calculation laws of regular expressions we will use:

Lemma 1 *Let $u, v, w \in REG(T)$. Then:*

- $i(u + v) = i(v + u)$ *(commutativity of $+$)*
- $i((u + v) + w) = i(u + (v + w))$ *and* $i((uv)w) = i(u(vw))$ *(associativity)*
- $i(u(v + w)) = i(uv + uw)$ *and* $i((v + w)u) = i(vu + wu)$ *(distributivity)* □

Proof Directly from definition 2.2.23.

From now on, we will identify regular expressions and their interpretation (i.e. the languages they generate) for the sake of notational ease; this is consistent with common general definitions of concatenation, union and Kleene star on languages. The results do not change if we are precise.

We will need the following result [Ard61] which allows us to eliminate recursion in language equations in favor of Kleene stars:

Lemma 2 (Arden's Lemma) *Let L, U, V be (regular) languages over alphabet Σ with $\varepsilon \notin U$. Then,*

$$L = UL \cup V \quad \Longleftrightarrow \quad L = U^*V$$

holds. □

Now let $A = (\{q_0, \ldots, q_n\}, \Sigma, \to, q_0, F)$ a finite automaton without ε-transitions[1]. We define sets $Q_0, \ldots, Q_n$ such that Q_i contains all words that are accepted by A *starting in* q_i, respectively. Obviously, $Q_0 = \mathcal{L}(A)$.

We can describe these sets by an equation system with one equation for each set Q_i:

$$Q_i = \bigcup_{q_i \xrightarrow{a} q_j} aQ_j \ \cup \begin{cases} \{\varepsilon\} & , q_i \in F \\ \emptyset & , \text{else} \end{cases} \tag{1}$$

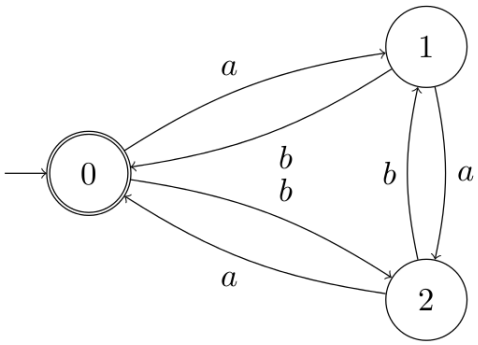
These equations apparently fit above interpretation. The system can now be solved by inserting equations into each another and using Arden's Lemma; the solution for Q_0 respectively the found representation as regular expression is the desired result.

Formal proof of correctness and general applicability of this technique is left out here.

[1] For arbitrary automata, you can always obtain an ε-free equivalent one by applying powerset construction.

Raphael Reitzig, 2011

Example 1 (by Georg Zetzsche) Consider the graph:



By equation (1), we get the following system of equations:

$$\begin{aligned} Q_0 &= aQ_1 \cup bQ_2 + \varepsilon \\ Q_1 &= bQ_0 \cup aQ_2 \\ Q_2 &= aQ_0 \cup bQ_1 \end{aligned}$$

Now plug equation three in the second, both rewritten as regular expressions:

$$\begin{aligned} Q_1 &= bQ_0 + a(aQ_0 + bQ_1) \\ &= abQ_1 + (b + aa)Q_0 \\ &= (ab)^*(b + aa)Q_0 \qquad (*) \end{aligned}$$

For $(*)$, we apply Arden's Lemma with $L = Q_1$, $U = i(ab)$ and $V = i((b + aa)) \cdot Q_0$. Note that all three languages are regular and $\varepsilon \notin U = \{ab\}$, enabling us to apply the lemma. Now we plug this result into the first equation:

$$\begin{aligned} Q_0 &= a(ab)^*(b + aa)Q_0 + baQ_0 + bb(ab)^*(b + aa)Q_0 + \varepsilon \\ &= ((a + bb)(ab)^*(b + aa) + ba)Q_0 + \varepsilon \\ &= ((a + bb)(ab)^*(b + aa) + ba)^* \qquad \text{(by ARDEN's Lemma)} \end{aligned}$$

Thusly, we have found a regular expression the language accepted by above automaton. Note that it is quite succinct but not uniquely determined; solving the equation system with a different sequence of manipulations leads to other – equivalent! – expressions. □

The avid reader may try to prove Arden's Lemma on their own. For completeness, one version is given here.

Proof (Arden's Lemma) Show the two implications claimed separately:

$\Longrightarrow$**:** Let $L \subseteq \Sigma^*$ with $L = UL \cup V$, $\varepsilon \notin U$. We have to show that $L = U^*V$; show the respective inclusions separately[2]:

$\supseteq$**:** We show that

$$U^*V = \left(\bigcup_{i\in\mathbb{N}} U^i\right) V = \bigcup_{i\in\mathbb{N}} U^iV \subseteq L$$

by showing that $U^iV \subseteq L$ for all $i \in \mathbb{N}$; induction over i:

IA: Clearly, $U^0V = V \subseteq UL \cup V = L$ by assumption on L.

IH: Assume that $U^iV \subseteq L$ for some $i \in \mathbb{N}$.

IS: The simple calculation

$$\begin{aligned} U^{i+1}V &= UU^iV \\ &\subseteq UL \qquad \text{(by induction hypothesis)} \\ &\subseteq UL + V = L \end{aligned}$$

concludes the inductive step.

$\subseteq$**:** Proof by contradiction; assume $L \not\subseteq U^*V$. Then, there is a *shortest* word $w \in L$ with $w \notin U^*V$. Since $L = UL \cup V$, we have $w \in UL$ or $w \in V$. But clearly, $w \notin V \subseteq U^*V$, hence $w \in UL$. Therewith, $w = u \cdot w'$ with $u \in U$, i.e. $u \neq \varepsilon$. Since w is shortest word in $L \setminus U^*V$ and $|w'| < |w|$, we have $w' \in U^*V$. But then, $w = uw'$ with $u \in U$ and $w' \in U^*V$ implies that $w \in U^*V$ which yields a contradiction to the choice of w. Therefore, there can be no such w, i.e. $L \subseteq U^*V$.

$\Longleftarrow$**:** This direction is immediately proven by:

$$L = U^*V = (U^+ + \varepsilon)V = UU^*V + V = UL + V$$ ∎

References

[Ard61] Dean N. Arden. Delayed-logic and finite-state machines. In *Proceedings of the 2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, FOCS '61, pages 133–151. IEEE Computer Society, Washington, DC, USA, 1961. URL `http://dx.doi.org/10.1109/FOCS.1961.13`.

[2] As presented in class by Roland Meyer, 2011.