Lecture 9

Regular Expressions and Finite Automata

Simplification of Expressions

For small regular expressions, one can often see how to construct an equivalent automaton directly without going through the mechanical procedure of the previous lecture. It is therefore useful to try to simplify the expression first.

For regular expressions α, β, if $L(\alpha) = L(\beta)$, we write $\alpha \equiv \beta$ and say that α and β are *equivalent*. The relation $\equiv$ on regular expressions is an equivalence relation; that is, it is

- reflexive: $\alpha \equiv \alpha$ for all α;
- symmetric: if $\alpha \equiv \beta$, then $\beta \equiv \alpha$; and
- transitive: if $\alpha \equiv \beta$ and $\beta \equiv \gamma$, then $\alpha \equiv \gamma$.

If $\alpha \equiv \beta$, one can substitute α for β (or vice versa) in any regular expression, and the resulting expression will be equivalent to the original.

Here are a few laws that can be used to simplify regular expressions.

$$\begin{aligned}
\alpha + (\beta + \gamma) &\equiv (\alpha + \beta) + \gamma && (9.1)\\
\alpha + \beta &\equiv \beta + \alpha && (9.2)\\
\alpha + \emptyset &\equiv \alpha && (9.3)\\
\alpha + \alpha &\equiv \alpha && (9.4)\\
\alpha(\beta\gamma) &\equiv (\alpha\beta)\gamma && (9.5)
\end{aligned}$$

$$\begin{aligned}
\epsilon\alpha &\equiv \alpha\epsilon \equiv \alpha && (9.6)\\
\alpha(\beta+\gamma) &\equiv \alpha\beta+\alpha\gamma && (9.7)\\
(\alpha+\beta)\gamma &\equiv \alpha\gamma+\beta\gamma && (9.8)\\
\varnothing\alpha &\equiv \alpha\varnothing \equiv \varnothing && (9.9)\\
\epsilon+\alpha\alpha^* &\equiv \alpha^* && (9.10)\\
\epsilon+\alpha^*\alpha &\equiv \alpha^* && (9.11)\\
\beta+\alpha\gamma \le \gamma &\Rightarrow \alpha^*\beta \le \gamma && (9.12)\\
\beta+\gamma\alpha \le \gamma &\Rightarrow \beta\alpha^* \le \gamma && (9.13)
\end{aligned}$$

In (9.12) and (9.13), $\le$ refers to the subset order:

$$\begin{aligned}
\alpha \le \beta &\overset{\text{def}}{\iff} L(\alpha) \subseteq L(\beta)\\
&\iff L(\alpha+\beta) = L(\beta)\\
&\iff \alpha+\beta \equiv \beta.
\end{aligned}$$

Laws (9.12) and (9.13) are not equations but rules from which one can derive equations from other equations. Laws (9.1) through (9.13) can be justified by replacing each expression by its definition and reasoning set theoretically.

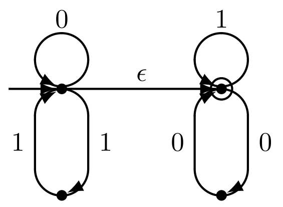
Here are some useful equations that follow from (9.1) through (9.13) that you can use to simplify expressions.

$$\begin{aligned}
(\alpha\beta)^*\alpha &\equiv \alpha(\beta\alpha)^* && (9.14)\\
(\alpha^*\beta)^*\alpha^* &\equiv (\alpha+\beta)^* && (9.15)\\
\alpha^*(\beta\alpha^*)^* &\equiv (\alpha+\beta)^* && (9.16)\\
(\epsilon+\alpha)^* &\equiv \alpha^* && (9.17)\\
\alpha\alpha^* &\equiv \alpha^*\alpha && (9.18)
\end{aligned}$$

An interesting fact that is beyond the scope of this course is that all true equations between regular expressions can be proved purely algebraically from the axioms and rules (9.1) through (9.13) plus the laws of equational logic [73].

To illustrate, let's convert some regular expressions to finite automata.

Example 9.1 $(11+0)^*(00+1)^*$



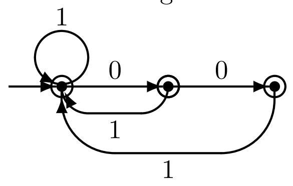
This expression is simple enough that the easiest thing to do is eyeball it. The mechanical method described in Lecture 8 would give more states and ϵ-transitions than shown here. The two states connected by an ϵ-transition cannot be collapsed into one state, since then 10 would be accepted, which does not match the regular expression. □

Example 9.2 $(1+01+001)^*(\epsilon+0+00)$

Using the algebraic laws above, we can rewrite the expression:

$$\begin{aligned}(1+01+001)^*(\epsilon+0+00) &\equiv ((\epsilon+0+00)1)^*(\epsilon+0+00)\\ &\equiv ((\epsilon+0)(\epsilon+0)1)^*(\epsilon+0)(\epsilon+0).\end{aligned}$$

It is now easier to see that the set represented is the set of all strings over $\{0,1\}$ with no substring of more than two adjacent 0's.



□

Just because all states of an NFA are accept states doesn't mean that all strings are accepted! Note that in Example 9.2, 000 is not accepted.

Converting Automata to Regular Expressions

To finish the proof of Theorem 8.1, it remains to show how to convert a given finite automaton M to an equivalent regular expression.

Given an NFA

$$M \;=\; (Q,\,\Sigma,\,\Delta,\,S,\,F),$$

a subset $X \subseteq Q$, and states $u, v \in Q$, we show how to construct a regular expression

$$\alpha_{uv}^{X}$$

representing the set of all strings x such that there is a path from u to v in M labeled x (i.e., such that $v \in \widehat{\Delta}(\{u\}, x)$) and all states along that path, with the possible exception of u and v, lie in X.

The expressions are constructed inductively on the size of X. For the basis $X = \varnothing$, let $a_1, \ldots, a_k$ be all the symbols in Σ such that $v \in \Delta(u, a_i)$. For $u \neq v$, take

$$\alpha_{uv}^{\varnothing} \overset{\text{def}}{=} \begin{cases} a_1 + \cdots + a_k & \text{if } k \geq 1, \\ \boldsymbol{\varnothing} & \text{if } k = 0; \end{cases}$$

and for $u = v$, take

$$\alpha_{uv}^{\varnothing} \overset{\text{def}}{=} \begin{cases} a_1 + \cdots + a_k + \epsilon & \text{if } k \geq 1, \\ \epsilon & \text{if } k = 0. \end{cases}$$

For nonempty X, we can choose any element $q \in X$ and take

$$\alpha_{uv}^{X} \overset{\text{def}}{=} \alpha_{uv}^{X-\{q\}} + \alpha_{uq}^{X-\{q\}}(\alpha_{qq}^{X-\{q\}})^* \alpha_{qv}^{X-\{q\}}. \tag{9.19}$$

To justify the definition (9.19), note that any path from u to v with all intermediate states in X either (i) never visits q, hence the expression

$$\alpha_{uv}^{X-\{q\}}$$

on the right-hand side of (9.19); or (ii) visits q for the first time, hence the expression

$$\alpha_{uq}^{X-\{q\}},$$

followed by a finite number (possibly zero) of loops from q back to itself without visiting q in between and staying in X, hence the expression

$$(\alpha_{qq}^{X-\{q\}})^*,$$

followed by a path from q to v after leaving q for the last time, hence the expression

$$\alpha_{qv}^{X-\{q\}}.$$

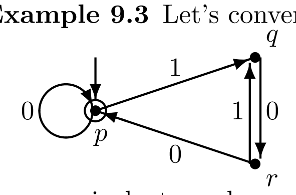
The sum of all expressions of the form

$$\alpha_{sf}^{Q},$$

where s is a start state and f is a final state, represents the set of strings accepted by M.

As a practical rule of thumb when doing homework exercises, when choosing the $q \in X$ to drop out in (9.19), it is best to try to choose one that disconnects the automaton as much as possible.

Example 9.3 Let's convert the automaton



to an equivalent regular expression. The set accepted by this automaton will be represented by the inductively defined regular expression

$$\alpha_{pp}^{\{p,q,r\}},$$

since p is the only start and the only accept state. Removing the state q (we can choose any state we like here), we can take

$$\alpha_{pp}^{\{p,q,r\}} = \alpha_{pp}^{\{p,r\}} + \alpha_{pq}^{\{p,r\}}(\alpha_{qq}^{\{p,r\}})^*\alpha_{qp}^{\{p,r\}}.$$

Looking at the automaton, the only paths going from p to p and staying in the states $\{p,r\}$ are paths going around the single loop labeled 0 from p to p some finite number of times; thus we can take

$$\alpha_{pp}^{\{p,r\}} = 0^*.$$

By similar informal reasoning, we can take

$$\begin{aligned}
\alpha_{pq}^{\{p,r\}} &= 0^*1, \\
\alpha_{qq}^{\{p,r\}} &= \epsilon + 01 + 000^*1 \\
&\equiv \epsilon + 0(\epsilon + 00^*)1 \\
&\equiv \epsilon + 00^*1, \\
\alpha_{qp}^{\{p,r\}} &= 000^*.
\end{aligned}$$

Thus we can take

$$\alpha_{pp}^{\{p,q,r\}} = 0^* + 0^*1(\epsilon + 00^*1)^*000^*.$$

This is matched by the set of all strings accepted by the automaton. We can further simplify the expression using the algebraic laws (9.1) through (9.18):

$$\begin{aligned}
&0^* + 0^*1(\epsilon + 00^*1)^*000^* \\
&\quad\equiv 0^* + 0^*1(00^*1)^*000^* && \text{by (9.17)} \\
&\quad\equiv \epsilon + 00^* + 0^*10(0^*10)^*00^* && \text{by (9.10) and (9.14)} \\
&\quad\equiv \epsilon + (\epsilon + 0^*10(0^*10)^*)00^* && \text{by (9.8)} \\
&\quad\equiv \epsilon + (0^*10)^*00^* && \text{by (9.10)} \\
&\quad\equiv \epsilon + (0^*10)^*0^*0 && \text{by (9.18)} \\
&\quad\equiv \epsilon + (0 + 10)^*0 && \text{by (9.15).}
\end{aligned}$$

□

Historical Notes

Kleene [70] proved that deterministic finite automata and regular expressions are equivalent. A shorter proof was given by McNaughton and Yamada [85].

The relationship between right- and left-linear grammars and regular sets (Homework 5, Exercise 1) was observed by Chomsky and Miller [21].