1 Rewriting

Partial rewrite existence

Take any regex. Clearly, for that regex to match, it must recognize each substring of the match. Therefore, a regex exists for every substring of a match.

From a DFA perspective, every state can be converted to an accepting state which will then match every substring.

Rewrite rules

$$\begin{aligned}
\mathcal{P}(s) &= (s|s_{[0,k-1]}|\dots|s_0) \\
\mathcal{P}(E?) &= \mathcal{P}(E)? \\
\mathcal{P}(E*) &= E*\mathcal{P}(E)? \\
\mathcal{P}(E|E) &= \mathcal{P}(E)|\mathcal{P}(E) \\
\mathcal{P}(E_1E_2) &= E_1\mathcal{P}(E_2)|\mathcal{P}(E_1) \\
\mathcal{P}(E\{k\}) &= \mathcal{P}(E_1E_2...E_k)
\end{aligned}$$

$\mathcal{P}(s)$ Rewrite

Given a string, its partial regex is:

$$\mathcal{P}(s) = (s[0:k]|s[0:k-1]|\dots|s[0])$$

The order of longest substring to shortest is important; many regex engines will short circuit on the first | encountered. Here is an example of an incorrect partial regex if the order is shortest to longest substring:

$$\mathcal{P}(abc) = (a|ab|abc)$$

Then, `/(a|ab|abc)/.match(abc)` will short circuit to a rather than the full match abc.

$\mathcal{P}(E?)$ Rule

$\mathcal{P}(E)$ matches any substring; $\mathcal{P}(E)? = (\mathcal{P}(E)|\epsilon)$. Therefore, this matches any substring or the empty string, satisfying the ? operator. ■

$\mathcal{P}(E*)$ Rule

IH:

$$\mathcal{P}(E^k) = E^j\mathcal{P}(E)?, 0 \le j \le k$$

For $k = 0$, then resolving the expression as $E^0\epsilon$ matches all substrings. For $k = 1$, resolving the expression as $E^0\mathcal{P}(E)$ matches all of its substrings.

$$
\begin{aligned}
\mathcal{P}(E^{k+1}) &= E^j\mathcal{P}(E)?, 0 \le j \le k+1 \\
&= (E^{k+1}\mathcal{P}(E)?|(E^j\mathcal{P}(E)?, 0 \le j \le k)) && \text{Split interval} \\
&= E^{k+1}|\mathcal{P}(E^k) && \text{Resolve } E? = \epsilon\text{, Apply inductive hypothesis}
\end{aligned}
$$

The above shows the partial transform can match on E^{k+1} or any of its E^k substrings which includes $E^k\mathcal{P}(E)$ — that is, right up until E^{k+1}. ■

$\mathcal{P}(E_1|E_2)$ Rule

$$\mathcal{P}(E_1|E_2) = \mathcal{P}(E_1)|\mathcal{P}(E_2)$$

Clearly $\mathcal{P}(E_1)$ matches all substrings of E_1 and $\mathcal{P}(E_2)$ matches all substrings of E_2; this rewrite matches all substrings of either expression. ■

$\mathcal{P}(E_1E_2)$ Rule

$$\mathcal{P}(E_1E_2) = E_1\mathcal{P}(E_2)|\mathcal{P}(E_1)$$

If s is a substring of E_1, then it is matched by $\mathcal{P}(E_1)$. Otherwise, if s is a substring of E_1 concat some substring of E_2 then $E_1\mathcal{P}(E_2)$ matches it. ■