Semantics of Regular Expressions

1 Operational Semantics

\[ \vdash r_1 \text{ matches } s_1 \text{ leaving } s_2 \quad \vdash r_2 \text{ matches } s_2 \text{ leaving } s_3 \]

\[ \vdash r_1 r_2 \text{ matches } s_1 \text{ leaving } s_3 \]

\[ \vdash r_1 \text{ matches } s_1 \text{ leaving } s_2 \]

\[ \vdash r_1 | r_2 \text{ matches } s_1 \text{ leaving } s_2 \]

\[ \vdash r_2 \text{ matches } s_1 \text{ leaving } s_2 \]

\[ \vdash r_1 | r_2 \text{ matches } s_1 \text{ leaving } s_2 \]

\[ \vdash r_1 \ast \text{ matches } s_1 \text{ leaving } s_1 \]

\[ \vdash r \text{ matches } s_1 \text{ leaving } s_2 \quad \vdash r \ast \text{ matches } s_2 \text{ leaving } s_3 \]

\[ \vdash r_1 \ast \text{ matches } s_1 \text{ leaving } s_3 \]

2 Denotational Semantics

2.1 Disjunction

\[ \mathcal{R}[r_1 r_2](s) = \mathcal{R}[r_1](s) \cup \mathcal{R}[r_2](s) \]

or, equivalently:

\[ \mathcal{R}[r_1 r_2](s) = \{ x | x \in \mathcal{R}[r_1](s) \lor x \in \mathcal{R}[r_2](s) \} \]

2.2 Concatenation

\[ \mathcal{R}[r_1 r_2](s) = \{ x | \exists y. y \in \mathcal{R}[r_1](s) \land x \in \mathcal{R}[r_2](y) \} \]

or, equivalently:

\[ \mathcal{R}[r_1 r_2](s) = \bigcup_{y \in \mathcal{R}[r_1]} \mathcal{R}[r_2](y) \]

2.3 Kleene Closure

Let \( r^0 \equiv \text{empty} \) and \( r^n \equiv r_1 r_2 \ldots r_n \) (i.e., \( r \) concatenated with itself \( n \) times).

\[ \mathcal{R}[r^*](s) = \bigcup_{k \in 0\ldots\infty} \mathcal{R}[r^k](s) \]

or, equivalently:

Consider the unwinding equation \( r^* \equiv rr^* \). We define a context \( C \) (a regexp with a hole) so that \( C \equiv r^* \). Note that \( r^* \equiv C[r^*] \). The meaning of a context is a semantic function \( F \) such that \( F[C[r^*]] = F[r^*] \). The type of \( F \) is:

\[ F : (S \rightarrow \mathcal{P}(S)) \rightarrow (S \rightarrow \mathcal{P}(S)) \]

We want the least fixed point of \( F \), where \( \text{least} \) is interpreted with respect to set inclusion \( \subseteq \). We assert that \( F \) is monotonic and continuous. Let \( F^0(W) = \mathcal{R}[\text{empty}] = \lambda s.\{s\} \). We define \( F^{k+1} \) as follows:

\[ F^{k+1}(W) = F F^k(W) = \lambda s. \bigcup_{y \in \mathcal{R}[r](y)} F^k(y) \]

Then we want the least fixed point:

\[ \mathcal{R}[r^*](s) = \bigcup_{k} F^k(\lambda s.\{s\}) = \bigcup_{k} F^k(\lambda s.\{s\}) \]

3 Incorrect Answers

The following definition of Kleene star is incorrect:

\[ \mathcal{R}[r^*](s) \neq \{s\} \cup \mathcal{R}[rr^*] \]

Using the rule for concatenation above, it is equivalent to the following also-incorrect definition:

\[ \mathcal{R}[r^*](s) \neq \{s\} \cup \{ x \mid \exists y. y \in \mathcal{R}[r](s) \land x \in \mathcal{R}[r^*](y) \} \]

The definitions are incorrect because they define \( \mathcal{R}[r^*] \) directly in terms of itself. Such circular definitions correspond to implementation code such as:

\begin{verbatim}
1 | Star(r) -> (* incorrect *)
2 | matches (Or(Empty, Concat(r, Star(r)))) s
\end{verbatim}

On regular expressions such as \( r = \text{empty}\ast \), this leads to an infinite loop (and usually a stack overflow).

There are two typical approaches for a correct implementation. The first chooses some large \( k \) (say, based on the length of the input string \( s \)) and computes \( \cup_{i=0..k} \mathcal{R}[r^i](s) \). The second actually computes the fixed point (instead of picking \( k \) in advance) by repeating the process until nothing new is added to the answer.

Regular expression matching is used almost everywhere. Note that understanding the denotational semantics actually helps one to write a real-world program correctly.