Semantics of Regular Expressions

1 Operational Semantics

$$\frac{\vdash r_1 \text{ matches } s_1 \text{ leaving } s_2 \quad \vdash r_2 \text{ matches } s_2 \text{ leaving } s_3}{\vdash r_1 r_2 \text{ matches } s_1 \text{ leaving } s_3}$$

$$\frac{\vdash r_1 \text{ matches } s_1 \text{ leaving } s_2}{\vdash r_1 | r_2 \text{ matches } s_1 \text{ leaving } s_2}$$

$$\frac{\vdash r_2 \text{ matches } s_1 \text{ leaving } s_2}{\vdash r_1 | r_2 \text{ matches } s_1 \text{ leaving } s_2}$$

$$\frac{}{\vdash r_1 * \text{ matches } s_1 \text{ leaving } s_1}$$

$$\frac{\vdash r \text{ matches } s_1 \text{ leaving } s_2 \quad \vdash r* \text{ matches } s_2 \text{ leaving } s_3}{\vdash r_1 * \text{ matches } s_1 \text{ leaving } s_3}$$

2 Denotational Semantics

2.1 Disjunction

$$\mathcal{R}\llbracket r_1 | r_2 \rrbracket(s) = \mathcal{R}\llbracket r_1 \rrbracket(s) \ \cup \ \mathcal{R}\llbracket r_2 \rrbracket(s)$$

or, equivalently:

$$\mathcal{R}\llbracket r_1 | r_2 \rrbracket(s) = \{x \mid x \in \mathcal{R}\llbracket r_1 \rrbracket(s) \vee x \in \mathcal{R}\llbracket r_2 \rrbracket(s)\}$$

2.2 Concatenation

$$\mathcal{R}\llbracket r_1 r_2 \rrbracket(s) = \{x \mid \exists y.\ y \in \mathcal{R}\llbracket r_1 \rrbracket(s) \wedge x \in \mathcal{R}\llbracket r_2 \rrbracket(y)\}$$

or, equivalently:

$$\mathcal{R}\llbracket r_1 r_2 \rrbracket(s) = \bigcup_{y \in \mathcal{R}\llbracket r_1 \rrbracket s} \mathcal{R}\llbracket r_2 \rrbracket(y)$$

2.3 Kleene Closure

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

$$\mathcal{R}\llbracket r* \rrbracket(s) = \bigcup_{k \in 0 \ldots \infty} \mathcal{R}\llbracket r^k \rrbracket(s)$$

or, equivalently:

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

$$F : (S \to \mathcal{P}(S)) \to (S \to \mathcal{P}(S))$$

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

$$F^{k+1}(W) = FF^k(W) = \lambda s. \bigcup_{y \in \mathcal{R}\llbracket r \rrbracket(s)} F^k(y)$$

Then we want the least fixed point:

$$\mathcal{R}\llbracket r* \rrbracket(s) = \bigsqcup_k F^k(\lambda s.\{s\}) = \bigcup_k F^k(\lambda s.\{s\})$$

3 Incorrect Answers

The following definition of Kleene star is *incorrect*:

$$\mathcal{R}\llbracket r* \rrbracket(s) \neq \{s\} \cup \mathcal{R}\llbracket rr* \rrbracket$$

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

$$\mathcal{R}\llbracket r* \rrbracket(s) \neq \{s\} \cup \{x \mid \exists y.\ y \in \mathcal{R}\llbracket r \rrbracket(s) \wedge x \in \mathcal{R}\llbracket r* \rrbracket(y)\}$$

The definitions are *incorrect* because they define $\mathcal{R}\llbracket r* \rrbracket$ directly in terms of itself. Such circular definitions correspond to implementation code such as:

```
  | Star(r) -> (* incorrect *)
      matches (Or(Empty,Concat(r,Star(r)))) s
```

On regular expressions such as $r = \textsf{empty}*$, 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 $\cup_{i=0..k} \mathcal{R}\llbracket r^k \rrbracket(s)$. 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.