String Regular Expression Abstract Domain

Ben Hardekopf

1 Regular Expressions Lattice

Let Σ be an alphabet. Let R be the set of normalized regular expressions over Σ and $r, r_1, r_2 \in R$. We will assume that any operation producing a regular expression also normalizes that expression, thus for any given regular language there is exactly one regular expression. The regular expressions lattice is $(R, \sqsubseteq, \sqcup, \sqcap)$ where:

- $\top$ is defined as $\Sigma*$.
- $\bot$ is defined as $\emptyset$ (i.e., the empty language).
- $\sqcup$ is defined as $r_1 \sqcup r_2 = r_1 \mid r_2$ (i.e., union).
- $\sqcap$ is defined as $r_1 \sqcap r_2 = r_1 \,\&\, r_2$ (i.e., intersection).
- $\sqsubseteq$ is defined as $r_1 \sqsubseteq r_2$ iff r_2 contains r_1 (recall that language containment for regular expressions is decidable).

1.1 Lattice Properties

The lattice is non-noetherian because given any finite regular language we can successively add infinitely more strings it doesn't yet contain, and hence there are infinite ascending chains.

2 Abstraction and Concretization Functions

Let $L(r)$ be the language denoted by r.

$$\alpha : \mathcal{P}(\Sigma*) \to R$$

$$\alpha(x) = \begin{cases} \bot & \text{if } x = \{\} \\ r \text{ s.t. } L(r) = x & \text{if } x \text{ is regular} \\ \top & \text{otherwise} \end{cases}$$

Note that in general there is no "closest" regular approximation of a non-regular language; we could do better than $\top$ but it would be unreasonably complicated.

$$\gamma : R \to \mathcal{P}(\Sigma*)$$
$$\gamma(r) = L(r)$$

3 Abstract + Operator

Define $\widehat{+}$ as $r_1 + r_2 = r_1 \cdot r_2$, i.e., regular expression concatenation.

$\widehat{+}$ is monotone if a' contains a and b' contains b implies $a' \cdot b'$ contains $a \cdot b$. This is necessarily true by the definition of language concatenation.

4 Abstract $\leq$ Operator

Defining a precise $\widehat{\leq}$ operator on regular expressions would be extremely complex, but we can do better than the naive "set everything to $\top$" strategy. In fact, we can be at least as precise as the prefixes domain: define $lcp(r)$ as the longest common prefix across all strings in the language denoted by r. This can be determined, for example, by computing the minimal DFA for r and finding the longest series of transitions from the start state until we reach a state that has more than one outgoing transition. Then use the prefixes domain definition of $\widehat{\leq}$.