Derivatives, States, and Languages

Robert M. Keller

12 October 2008

We previously introduced the concept of derivatives of regular expressions. For any **regular expression** R and any letter σ, the **derivative** R/σ can be computed recursively with a finite set of rules:

$\emptyset/\sigma = \emptyset$	$(R+S)/\sigma = (R/\sigma + S/\sigma)$
$\lambda/\sigma = \emptyset$	$(RS)/\sigma = (R/\sigma)S$, if $\lambda \notin R$
$\sigma/\sigma = \lambda$	$(RS)/\sigma = (R/\sigma)S + S/\sigma$, if $\lambda \in R$
$\sigma/\sigma' = \emptyset$ if $\sigma \neq \sigma'$	$(R^*)/\sigma = (R/\sigma) R^*$

We also showed that the states of a DFA accepting the language denoted by R can be constructed using the derivatives as its states, provided there is a means of identifying whether or not two such derivatives are equivalent. (This can be done, but it is not totally obvious that it is a complete method.) On the other hand, it is easy to determine whether or not $\lambda \in R$ for any R.

We now **generalize** the derivative concept to derivatives with respect to **strings** rather than just symbols. If Σ is an alphabet, let Σ^* stand for the set of all (finite) strings with letters in the alphabet. Define the derivative R/x for any $x \in \Sigma^*$ as follows:

- R/σ for $\sigma \in \Sigma$ is as given by the rules in the table
- $R/\lambda = R$
- $R/(x\sigma) = (R/x)/\sigma$, recursively

Here we have taken liberty in equating a string with one letter with the letter itself, for convenience.

By a **language** over Σ, we mean any set of strings, i.e. any subset of Σ^*. For example, each regular expression represents a language, and there is a DFA accepting such a language. A **regular language** is one that is representable by a regular expression, or equivalently, accepted by a DFA.

There are **non-regular languages**, for example

$$\{0^n1^n \mid n \in N\}$$

with N being the set of natural numbers {0, 1, 2, 3, …} and σ^n representing n copies of σ. This fact is proved below.

We now generalize derivatives to languages in general. If L is a language and $x \in \sum^*$, define

$$L/x = \{y \in \sum^* \mid xy \in L\}$$

Note that L/x is similar to a derivative, but there is not necessarily a mechanical way to compute it, as we haven't said how L is **represented**. If L is represented by a regular expression R, then L/x can be computed as R/x for any given x, according to the rules above.

Notice that it is possible that $L/x_1 = L/x_2$ even when $x_1 \neq x_2$. In fact, if L is regular, then there is only a **finite** set of possible sets L/x as x ranges over $\sum^*$. The possible values of L/x correspond to the states of the *smallest* DFA accepting L. Hence we are justified in calling the sets L/x **states**.

The definition of L/x works for any language L, not just regular ones. We can think of the sets L/x as x ranges over $\sum^*$ as the **abstract states** of the language L.

Theorem (due to Myhill and Nerode)

L is a regular language iff the set of abstract states of L is finite.

We say that two strings x_1 and x_2 are **equivalent** w.r.t. L provided that $L/x_1 = L/x_2$. We can apply the Myhill-Nerode theorem to show that certain languages are not regular. All we need to do is identify an **infinite** set of strings $\{x_0, x_1, x_2, \ldots\}$ such that **no pair** of the strings are equivalent.

Example: This language is not regular:

$$\{0^n 1^n \mid n \in N\}$$

Try to find the infinite set of inequivalent pairs. How do you demonstrate they are inequivalent?

Example: This language is regular:

$$\{x \in \{0, 1\}^* \mid x \text{ is a multiple of 3 in binary, MSB first}\}$$

What is the set of **representatives** of the equivalence classes?