CSLI Studies in
Computational Linguistics

Finite State Morphology

Kenneth R. Beesley
Lauri Karttunen

CSLI PUBLICATIONS
Center for the Study of Language and Information
Stanford, California

Regular expressions were originally invented as a metalanguage to describe languages. The formalism was subsequently extended to describe relations. But as we have noted in many places, some operators can be applied only to a subset of regular expressions. Let us now review this issue in more detail.

2.3.3 Closure

A set operation such as union has a corresponding operation on finite-state networks only if the set of regular relations and languages is CLOSED under that operation. Closure means that if the sets to which the operation is applied are regular, the result is also regular, that is, encodable as a finite-state network.

Regular languages are closed with respect to all the common set operations including intersection, subtraction and complementation (= negation). This follows directly from Kleene's proof. Regular relations are closed under concatenation, iteration, union, and composition but not, in general, under intersection, complementation or subtraction (Kaplan and Kay, 1994; Roche and Schabes, 1997).

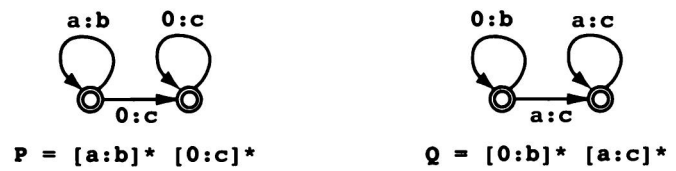

Figure 2.2: Networks for Two Regular Relations

Kaplan and Kay give a simple example of a case in which the intersection of two finite-state relations is not a finite-state relation. Consider two regular expressions and the corresponding networks in Figure 2.2. P is the relation that maps strings of any number of **a**s into strings of the same number of **b**s followed by zero or more **c**s. Q is the relation that maps strings of any number of **a**s into strings of the same number of **c**s preceded by zero or more **b**s.

Table 2.6 shows the corresponding string-to-string relations and their intersection. The left side of Table 2.6 is a partial enumeration of the P relation. The right side partially enumerates the Q relation. The middle section of the table contains the intersection of the two relations, that is, the pairs they have in common.

It is easy to see that the intersection contains only pairs that have on their lower side (i.e. as the second component) a string that contains some number of **b**s followed by exactly the same number of **c**s.

The lower-side language, $b^n c^n$, is not a finite-state language but rather a CONTEXT-FREE LANGUAGE, generated by a phrase-structure grammar that crucially depends on center-embedding: S $\rightarrow$ ε, S $\rightarrow$ b S c. Consequently it cannot be encoded by a finite-state network (Hopcroft and Ullman, 1979). The same holds

P Regular	P & Q Not Regular	Q Regular
<"", "">	<"", "">	<"", "">
<"", c>		<"", b>
<"", cc>		<"", bb>
...		...
<a, b>		<a, c>
<a, bc>	<a, bc>	<a, bc>
<a, bcc>		<a, bbc>
<a, bccc>		<a, bbbc>
...		...
<aa, bb>		<aa, cc>
<aa, bbc>		<aa, bcc>
<aa, bbcc>	<aa, bbcc>	<aa, bbcc>
<aa, bbccc>		<aa, bbbcc>
...		...

Table 2.6: Non-Regular Intersection of P and Q

of course for any relation involving this language. No operation on the networks in Figure 2.2 can yield a finite-state transducer for the intersection of P and Q; such a transducer does not exist.

The relations P and Q both have the property that a string in one language corresponds to infinitely many strings in the other language because of the iterated 0:b and 0:c pairs. It is this characteristic, the presence of "one-sided epsilon loops" in Figure 2.2 that makes it possible that their intersection be not regular.

From the fact that regular relations are not closed under intersection it follows immediately that they are not closed under complementation either. Intersection can be defined in terms of complementation and union. If regular relations were closed under complementation, the same would be true of intersection. It also follows that regular relations are not closed under the subtraction operation, definable by means of intersection and complementation.

The closure properties of regular languages and relations are summarized in Table 2.7 for the most common operations.

Although regular relations are not in general closed under intersection, a subset of regular relations have regular intersections. In particular, equal-length relations, relations between pairs of strings that are of the same length, are closed under intersection, subtraction and complementation. Such relations can be encoded by a transducer that does not contain any epsilon symbols.[1]

[1]This fact is important for us because it is the formal foundation of the two-

Operation	Regular Languages	Regular Relations
union	yes	yes
concatenation	yes	yes
iteration	yes	yes
reversal	yes	yes
intersection	yes	no
subtraction	yes	no
complementation	yes	no
composition	(not applicable)	yes
inversion	(not applicable)	yes

Table 2.7: Closure Properties

The **Xerox** calculus allows intersection to apply to all simple automata and to transducers that do not contain any one-sided epsilon pairs. The test is more restrictive than it should be in principle because the presence of one-sided epsilons in a transducer does not necessarily indicate that the relation it encodes is of the type that could yield a non-regular intersection.

2.3.4 The ANY Symbol

So far we have given examples of regular expressions and the corresponding networks for most of the operators introduced in Section 2.3.1. One notable exception is complementation. We will come to that shortly but first we need to understand the semantics of the ANY symbol, `?`, introduced in the beginning of Section 2.3.1 as one of the atomic expressions.

Let us recall that the term complement of a language `A`, denoted by `\A`, is the union of the single-symbol strings that are not in `A`. For example, the language `[\a]` contains "b", "c", ..."z". In fact `[\a]` is an infinite language because the set of atomic symbols is in principle unbounded. Our symbol alphabet is not restricted to the 26 letters of the English alphabet or to the 256 ASCII characters.

For this reason we provide a special regular-expression symbol, `?`, to represent the infinite set of symbols in some yet unknown alphabet. It is called the ANY symbol. The regular expression `?` denotes the language of all single-symbol strings. Note that this set does not include the empty string. Because we do not make a distinction between a language and an identity relation, `?` can also be interpreted as the relation that maps any symbol into itself. The corresponding network is obviously the one in Figure 2.3. But note the annotation `Sigma: {?}`. We will explain it shortly.

level rule formalism called **twolc**, which is included on the CD-ROM and described on `http://www.fsmbook.com/`. Transducers compiled from two-level rules can be intersected because the 0 symbol is treated as an ordinary symbol in the rule formalism and not as an empty string.