Combining Regular Expressions with (Near-)Optimal Brzozowski Automata

Michiel Frishert and Bruce W. Watson

Technische Universiteit Eindhoven,
Department of Mathematics and Computer Science,
P.O.Box 513, NL-5600 MB Eindhoven, The Netherlands
michiel@michielfrishert.com, bruce@bruce-watson.com

Derivatives of regular expressions were first introduced by Brzozowski in [1]. By recursively computing all derivatives of a regular expression, and associating a state with each unique derivative, a deterministic finite automaton can be constructed. Convergence of this process is guaranteed if uniqueness of regular expressions is recognized modulo associativity, commutativity, and idempotence of the union operator. Additionaly, through simplification based on the identities for regular expressions, the number of derivatives can be further reduced.

Alternative approaches to computing the derivatives automaton that we have found either store duplicate copies of the parse trees, or compute and then determinize the (non-deterministic) partial derivatives automaton. An implementation using an approach similar to ours was published by Mark Hopkins in a 1993 note on the comp.compilers newsgroup, see [2].

Regular expressions are commonly represented as parse trees, and computation of derivatives can easily be implemented using such trees. The regular expression represented by a subtree of a parse tree is called a *subexpression*. Due to the nature of the Brzozowski derivatives, the same subexpression is often contained in more than one derivative. Such common subexpressions can be removed through the process of *global common subexpression elimination* (GCSE).

Our implementation uses n-ary parse trees rather than binary parse trees. This avoids the need of binary trees to keep the tree left (or right) heavy and sorted for fast equivalence detection. When creating a new node in the parse tree, with a given set (or list) of child nodes, we test for equivalent nodes by checking the parents of one of the child nodes. By storing the set of parents for each node in the parse tree, and through hash tables this can be done in near-constant time.

The n-ary parse tree along with GCSE ensure that equivalence is detected modulo associativity, idempotence and commutativity, which guarantees termination of the algorithm. Rewrite rules are used to implement simplification through the identities. The generic rewrite system allows us to add additional rewrite rules if desired. The rewrite rules are applied before GCSE.

Derivatives can be computed in two fashions: lazily (top-down), or eagerly (bottom-up). The first approach derivatives are computed as they are needed to find the derivatives of the parse tree root. The second approach computes derivatives for all subexpressions as they become available from parsing. In many cases, the exact same set of derivatives are computed in both methods, because

M. Domaratzki et al. (Eds.): CIAA 2004, LNCS 3317, pp. 319–320, 2005.

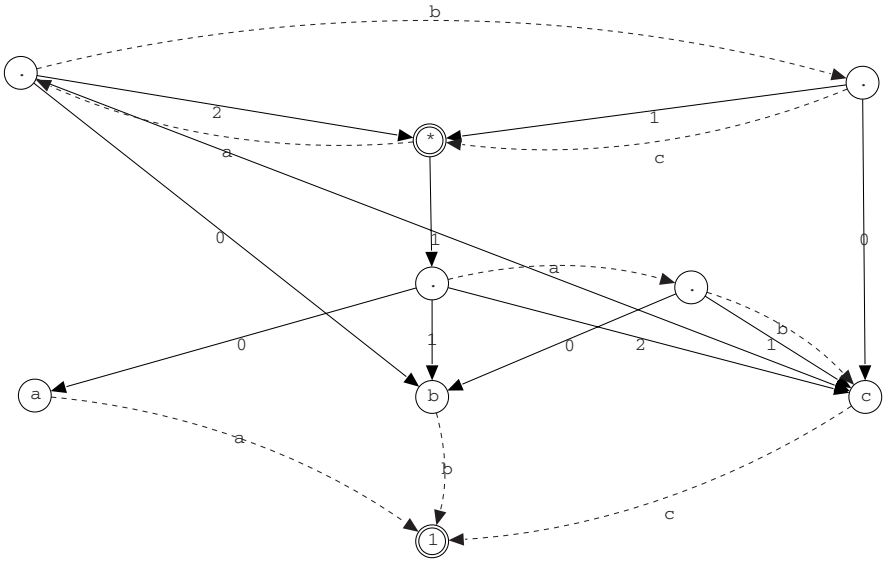

Fig. 1. Combined Parse Trees for the Derivatives of $(abc)^*$

the derivatives of a regular expression are defined in terms of the derivatives of its subexpressions. For the top-down approach there are cases where we can avoid computing the derivatives of subexpressions as follows: for each node in the parse tree we maintain a first-symbol set, which is the set of symbols for which that node will have a non-empty derivative. This can be computed directly from the parse tree. We only compute derivatives for those symbols in the first-symbol set. The improvement becomes obvious for example for the intersection of regular expressions ab and cd. Because the intersection of their first-symbol sets is empty, we do not compute derivatives of ab or cd, or for their subexpressions in turn. This benefits only the top-down approach, as the bottom-up approach already computes the derivatives of both subexpressions before ever encountering the node intersecting the two.

Figure 1 shows the combined parse graph for the derivatives of $(abc)^*$. Solid, straight lines form the parse graph, while dashed, curved lines represent the derivatives relations. The numbered edges indicate the order of concatenation in the parse graph.

References

1. Brzozowski, J.A.: Derivatives of Regular Expressions. Communications of the ACM **11** (1964) 481–494
2. Hopkins, M.: Regular expression software. Posted to the comp.compilers newsgroup (1994) ftp://iecc.com/pub/file/regex.tar.gz.