Journal of Automata, Languages and Combinatorics **25** (2020) 4, 349 – 355

DERIVATIVES OF REGULAR EXPRESSIONS WITH CUTS

Niklas Zechner

Språkbanken, Department of Swedish, University of Gothenburg
Box 100, 405 30 Gothenburg, Sweden
niklas.zechner@gu.se

ABSTRACT

Derivatives of regular expressions are an operation which for a given expression produces an expression for what remains after a specific symbol has been read. This can be used as a step in the process of transforming an expression into a finite string automaton. Cuts are an extension of the ordinary regular expressions; the cut operator is essentially a concatenation without backtracking, formalising a behaviour found in many programming languages. Just as for concatenation, we can also define an iterated cut operator. We show and derive expressions for the derivatives of regular expressions with cuts and iterated cuts.

Keywords: derivative, regular expression, automaton

1. Introduction

Derivatives of regular expressions were introduced in 1964 by Janusz A. Brzozowski [2]. The idea is to compute a resulting regular expression after a given symbol has been read by a given expression. This fundamental operation can help avoid repeating costly computations, and can be used to construct a finite automaton.

The cut operator is an additional operator that can be used in regular expressions. It can be seen as a variant of concatenation, with the difference being that the left operand always matches the maximum number of symbols. With the ordinary concatenation operator $E \cdot F$, typical implementations try to match as much as possible of the given string to E, and then backtrack if that does not work. For example, consider the expression $\mathrm{a}^* \cdot \mathrm{b}^* \cdot \mathrm{a} \cdot \mathrm{b}$, and the string "aab". A traditional algorithm will typically start by matching "aa" to the a^* part of the expression, but when that does not give an accepting result, it backtracks to matching only the first "a" to the a^* part, nothing to the b^* part, and lastly "ab" to $\mathrm{a} \cdot \mathrm{b}$. The cut operator is similar, but without backtracking. We write the cut of expressions E and F as $E \,!\, F$. If we replace one concatenation in the above example, and get $\mathrm{a}^* \cdot \mathrm{b}^* \,!\, \mathrm{a} \cdot \mathrm{b}$, this no longer matches the string "aab"; "aa" is matched to a^*, "b" is matched to b^*, and then when there is nothing to match with $\mathrm{a} \cdot \mathrm{b}$, no backtracking is done, so there is no match.