情報処理学会論文誌　プログラミング　Vol. 2　No. 1　22 (Jan. 2009)

発表概要

メモ化を用いた正規表現エンジンの実装

須賀功太[†1]　前田敦司[†1]　山口喜教[†1]

正規表現は文字列処理の基本的なツールとして，言語処理系やテキスト処理アプリケーションに広く用いられている．正規表現と入力文字列とのマッチング処理の実装には，正規表現から構成した非決定性有限オートマトン（NFA）を用い，バックトラッキングを用いて NFA の動作をシミュレートするアルゴリズムや，NFA を決定性有限オートマトン（DFA）に変換してバックトラッキングなしでマッチングを行うアルゴリズムが広く用いられている．NFA とバックトラッキングを用いる実装では，マッチングに要する時間的計算量が最悪で指数関数的になるという問題があり，一方で DFA を用いる実装では，事前に DFA を構成するための空間的・時間的計算量が指数関数的になるという問題点がある．NFA とバックトラッキングを用いる実装にメモ化を組み合わせて，最悪時の時間計算量を改善する提案がなされている．この場合の空間計算量は，正規表現パターン長と入力テキストサイズの積に比例する．また，NFA をバックトラッキングなしで（幅優先で）シミュレートするアルゴリズムも知られている．本発表では，これらの手法の効率を比較し，メモ化に用いるメモリ領域を削減する手法について考察する．

Implementation of Regular Expression Engine Using Memoization

Kota Suga,[†1] Atsushi Maeda[†1]
and Yoshinori Yamaguti[†1]

Regular expressions are widely used as a basic tool for programming language implementations and text processing applications. Most implementations of regular-expression matching engine fall into two categories: some implementations construct nondeterministic finite automata (NFAs) from given regular expression and simulate NFAs with backtracking simulation algorithms; others convert NFAs into DFAs and directly simulate them in non-backtracking algorithm. With implementation using NFA and backtracking, the time complexity becomes exponential in the worst case. While DFA-based implementation requires exponential space/time complexity in constructing DFAs. There are proposals for backtracking implementations to incoroprate memoization to improve worst-case time complxity. In this case, space requirement becomes regular expression size times input text size. Non-backtracking simulation algorithm of NFAs are also known. In this presentation, we compare the efficiency of these techniques. Arguments on a technique to reduce memoization space area is also presented.

（平成 20 年 8 月 6 日発表）

†1 筑波大学大学院システム情報工学研究科
Graduate School of System and Information Engineering, University of Tsukuba