実行時コンパイルを用いた正規表現評価器の実装

新屋 良磨

当研究室では, Concinuation based C という, 状態遷移記述に適した C の下位言語を提案している.Continuous bsed C は ステートメントより大きく, 関数よりも小さなプログラミング単位としてコードセグメントを持ち, コードセグメントからの継続を基本としている. 本研究では, 与えられた正規表現から, 等価な 有限状態オートマトンに変換し, オートマトンにおける状態遷遷移を Continuous based C/C による継続/関数呼び出しに変換する正規表現コンパイラを Python で実装した.

1 はじめに

近年, 実行時コンパイルによる高速化 (Just-in-Time Compile) が様々なプログラムで用いられている. これらは, コンパイラ理論の発展により実行時コンパイルにかかるオーバーヘッドよりも, コンパイルによって得られる機械語レベルのプログラムの実行速度が上回る場合において有効であり, たとえば Java の HotSpot や Python の PyPy など, 仮想マシンを持つ言語処理系の最適化技術として利用されている.

実行時コンパイルが可能な対象として, 正規表現評価器に着目した. 現在, 正規表現の評価器は, プログラミング言語の組み込み機能やライブラリ等, さまざまな実装が存在するが, それらの殆どは仮想マシン方式を採用している [4]. 仮想マシンを採用いた実装でも, 正規表現を内部表現に変換する処理を行っており, それらを “コンパイル” と呼ぶことが多い. 本研究で実装した評価器の “実行時コンパイル” とは, 正規表現を内部形式に変換することではなく, 正規表現 から実行バイナリを生成することを指す (3. 3 節). 本研究では, 実行バイナリの生成にはコンパイラ基盤である LLVM, GCC を用いており, 評価器全体の実装としては Python で実装した.

本論文では, まず正規表現のコンパイル方法について説明し, 実装した評価器の性能調査のために, 正規表現を用いてテキストマッチ処理を行う grep と同等の機能を実装し, GNU grep との比較を行う.

2 正規表現

2. 1 正規表現によるテキストマッチ

正規表現は与えられた文字列を受理するかどうかを判定できるパターンマッチングの機構であり, sed, grep, awk を始めとするテキスト処理ツールに広く利用されている. 正規表現には定められた文法によって記述され, 例えば, 正規表現 “a*b” は”a” の 0 回以上の繰り返し直後, “b” で終わる文字列 (“b”, “ab”, “’aaaab’) を受理し, “a(b|c)” は “a” で始まり, 直後が “b” または “c” で終わる文字列 (“ab”, “ac”) を受理する.

2. 2 正規表現の演算

本論文では, 以下に定義された演算をサポートする表現を正規表現として扱う.

1. 連接 二つの言語 L と M の連接 (LM) は, L に族する列を一つとり, そのあとに M に族する列

Implimentation of Regular Expression Engine with Just-In-Time Compilation.

Shinya Ryoma, 琉球大学工学部情報工学学科, Dept. of Information Engineering, Ryukyu University.

コンピュータソフトウェア, Vol.16, No.5 (1999), pp.78–83.
[研究論文] 1999 年 8 月 3 日受付.

を連接することによってできる列全体から成る集合である.

2. 集合和 二つの言語 L と M 集合和 (L|M) は, L または M(もしくはその両方) に属する列全体からなる集合である.
3. 閉包 言語 L の閉包 (L*) とは, L の中から有限個の列を重複を許して取り出し, それらを連接してできる列全体の集合である.

この 3 つの演算によって定義される表現は, 数学的には正則表現と定義されている. 本論文では, 特に区別のない限り, 正則表現と正規表現を同じものとして扱う.

2.3 grep

正規表現は, テキストのパターンをシンプルに記述できるという利点から, テキストファイルから, 任意のパターンにマッチするテキストを検索するなどの用途に使用される.

GNU grep は, それを実現するソフトウェアの一つであり, 引数として与えられたファイルから, 与えられた正規表現にマッチするテキストを含む行を出力する機能を持っている.

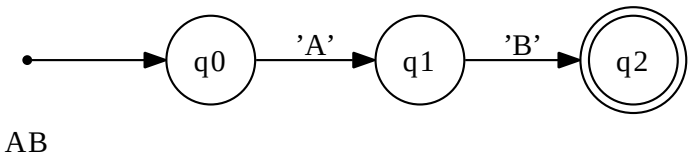
“与えられた正規表現にマッチするテキストを含む” というのは, 行の先頭から末尾まで正規表現によるマッチングを行い, 正規表現が受理状態になった時点で “含む” という解釈を行う. つまり, 正規表現 ”(a|s)t” は, ”at” または” st“を受理する正規表現であり, テキスト行”math.“の 2 3 文字目の”at” に一致するので grep は “math.” を出力する. また正規表現”a*” は, “a” の 0 回以上の繰り返しを受理する正規表現であり, 空文字も受理するので, grep は全ての行を出力することになる.

3 正規表現評価器の実装

正規表現は等価な NFA に, また NFA は等価な DFA に変換することが可能である [1]. 以下にその変換手方を説明する.

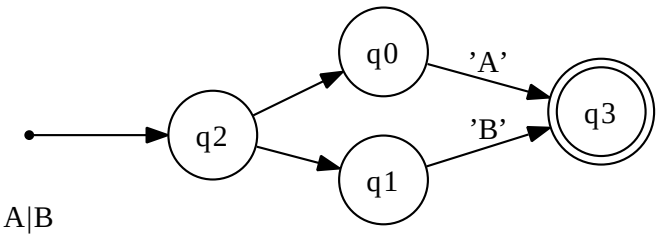
3.1 正規表現から NFA への変換

NFA は, 入力に対して複数の遷移先持つ状態の集合である.



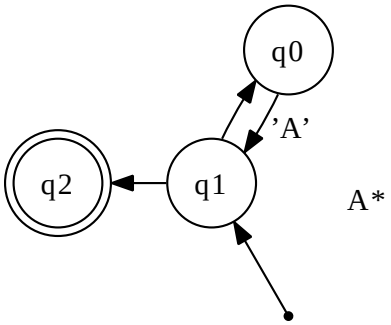
AB

図 1 “A” と “B” の連接



A|B

図 2 “A” と “B” の集合和



A*

図 3 “A” の閉包

正規表現が, 等価な NFA に変換できるということを,2.2 で定義した 3 つの演算について対応する NFA に変換できることから示す.

1. 連接 図 1 は正規表現 “AB” に対応する NFA となる.
2. 集合和 図 2 は正規表現 “A—B” に対応する NFA となる.
3. 閉包 図 3 は正規表現 “A*” に対応する NFA となる.

図 2, 3 において, ラベルのない矢印は無条件の遷移

を現しており,ε-動作と呼ばれる. また, 二重丸で囲まれた状態は受理状態を現しており, NFA において入力が終了した時点で, 受理状態を保持している場合に限り, その文字列を受理したことになる.

3.2 NFA から DFA への変換

3.3 DFA から実行バイナリの生成

DFA からの実行バイナリ生成には, 3 種類の実装を行った.

1. (**DFA -¿ Continuous based C -¿ gcc によるコンパイル**)
2. (**DFA -¿ C -¿ gcc によるコンパイル**)
3. (**DFA -¿ LLVM 中間表現 -¿ LLVM によるコンパイル**)

4 評価

...

参 考 文 献

[1] Hopcroft, J, E. Motowani, R. D, Ullman. : オートマトン言語理論計算論 I (第二版), pp. 39–90.
[2] Thompson, K : Regular Expression Search Algorithm, 1968
[3] Cox, R : Regular Expression Matching Can Be Simple And Fast, 2007
[4] Cox, R : Regular Expression Matching: the Virtual Machine Approach, 2009
[5] Cox, R : Regular Expression Matching in the Wild, 2010
[6] 長谷川 勇 : 国際正規表現ライブラリの開発