7ZD-07

同一判定に基づく自動採点機能を持つ正規表現学習支援システムの開発

呉　晟董†　　成　凱‡

九州産業大学大学院情報科学研究科†‡

1. はじめに

正規表現は，プログラミングにおけるパターンチェックや文字列検索などで必要不可欠である．しかし，初心者にとって，正規表現は，意味のわからない記号の羅列で，一定の訓練を受けないと正規表現を使いこなすことができない．本研究では，正規表現学習支援システムを開発し，正規表現の基本や正規表現で様々なパターンを表す手法を学習者に学ばせ，理解度チェック機能を提供することで，正規表現の学習支援を目指す．

2. 正規表現学習支援システムの構成

正規表現学習支援システムは大きく二つの部分に分けられている．まず，一つ目は正規表現の基本や正規表現で様々なパターンを表す手法を解説する学習コンテンツを提供する．二つ目は理解度チェック機能を提供し，記述式の問題（図 1）を与え，解答に対する自動採点を行う．つまり，解答結果が正解と同一であるかどうかについて，正規表現の同一判定を自動的に行う．この二つの部分を通して，学習者が問題を解いていくうちに正規表現を自然に使いこなせるようになるのが，システムを開発する目的である．

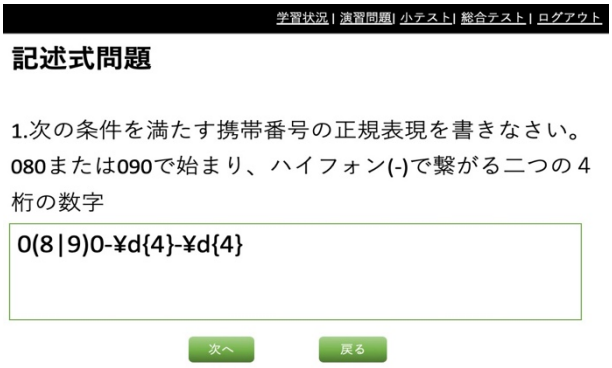

図 1 自動採点機能を持つ正規表現学習支援

3. 正規表現の同一判定の方法

正規表現は書き方が違っても同じ意味を持っていることがある．例えば，/a(b|c)/ と /ab|ac/ は書き方が違っても実質同じ言語を表している．したがって，記述式問題の自動採点をするには正規表現の同一判定が必要である．二つの正規表現 RE1 と RE2 が同じ言語を表すとき，RE1 と RE2 が同一であるという．

正規表現の同一判定は有限オートマトンの同一性によって行う Hopcroft & Karp アルゴリズム[1][2]が有名である． しかし，正規表現を有限オートマトンに変換したり，有限オートマトンを最小化したりするに時間・空間的にコストが高く，止むを得ない時に使う．本研究では，（１）変形による同一判定，（２）マッチング・テストによる同一判定を提案する．

3. 1. 変形による同一判定

正規表現を等価に変形して変形後の正規表現で同一判定を行う（図 2）．

正規表現 RE1 と RE2 を標準形 RE1＇と RE2＇に変形し，RE1＇と RE2＇の同一判定から RE1 と RE2 が同一かどうかを判定する．ここの標準形とは，演算子に OR(|)，クロージャ(*)，空文字（ε），括弧によるグループ化しか含まない形である．

① メタ文字+の変形
　a+ = aa*
② 繰り返し回数の変形
　a{3} = aaa，a{1,3} = a|aa|aaa，
　a{,3}= ε |a|aa|aaa，a{2,} = aaa*
③ 交換法則による変形
　b|x|a = a|b|x
④ 分配法則による変形
　abc|abx = ab(c|x) = ab[cx]

標準形 RE1＇と RE2＇が同一と判定できない場合は，以下の通り処理が続く．もし RE1＇と RE2＇に OR 演算子が含まない場合は，RE1 と RE2 が同一ではないと判定する．

RE1＇と RE2＇に OR を含む場合は次のステップに入る．このステップでは，文字クラスの同一性により正規表現の同一判定を行う．まず，RE1＇と RE2＇に含まれる文字クラスを抽出し，%n で表す．例えば，%1 := a|b|c，%2 := [a-c]とする．次に抽出した文字クラスの同一性を判定し，同じクラスを同じ記号で表す．例えば，上の%1 = %2 なので，RE1＇と RE2＇にある%2 をすべて%1 と置き換える．最後は RE1＇と RE2＇の構文木を構築して比較する．構文木が同じであれば RE1＇と RE2＇は同じである．もし，判定ができない場合は最後のステップに入る．

3.2. 有限オートマトンによる同一判定

このステップでは決定性有限オートマトン(DFA)を用いて正規表現の同一判定を行う.

1. 標準形に変換
   RE1'<-RE1の標準形
   RE2'<-RE2の標準形
2. 直接比較
   もしRE1'=RE2', Trueを返して終了
   もし, RE1'とRE2'にORが含まない場合
   RE1'!=RE2', Falseを返して終了
3. 文字クラスの同一性による判定
   (1) RE1'とRE2'に含まれる文字クラスを抽出し, %nで表す.
   (2)抽出した文字クラスの同一性を判定し, 同じクラスを同じ記号で表す.
   (3) RE1'とRE2'の構文木を構築
   構文木が同じであれば, Trueを返して終了
4. オートマトンによる同一判定
   (1) 構文木からNFAを構築
   (2) NFAをDFAに変換する
   (3) DFAを最小化する
   (4) DFAの開始状態が同値であれば, Trueを返し, 同値でないならば, Falseを返して終了

図 2 正規表現変形による同一判定

まず, 構文木から有限オートマトンDFAを構築する. DFAにおける二つの状態pとqから同じ文字列を読み進むとき, 両者がともに受理状態, または, 両者がともに受理状態ではないならば, pとqは同値であるという. 図3に示す二つのDFAについて, AからEまでの5個の状態を持つ一つのDFAを表すものと考える.

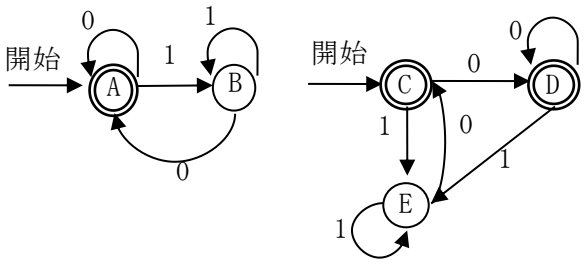

図 3 同じ言語を受理する異なるDFA

穴埋めアルゴリズム[2]を適用すると, 図4に示すような結果が得られる. 状態AとCが同値であることがわかる. これらの状態は元の二つのオートマトンの初期状態であったため, 元の二つのDFAは同じ言語を受理することが結論される.

決定性有限オートマトンの状態数が増えると計算量が大きくなるため, 判定するのに時間がかかる.

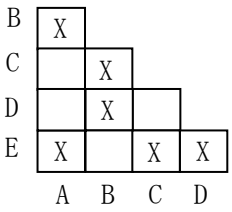

図 4 穴埋めアルゴリズムによる同値状態判定

3.3.マッチング・テストによる同一判定

変形による同一判定ではオートマトンの状態数が多くなるにつれコストが高くなる. もう一つの方法はRE1にマッチするテストデータを生成して, RE2にマッチするかを判定する(図5). マッチしないならば両者は同一ではない. すべてマッチした場合に, 次はRE2からテストデータを生成して, RE1にマッチするかを判定する. ここでもマッチしないならばRE1とRE2は同じではない. この方法は, 二つの正規表現が違う場合しか判定できないが, 判定する時間が上記の方法よりは早い.

1. RE1にマッチする文字列をk個生成する
2. 文字列 $s_i(i=1,2,\cdots,k)$ がRE2にマッチするかを判定する
   マッチしないならば, Falseを返して終了
3. RE2にマッチする文字列をk個生成する
4. 文字列 $s_i(i=1,2,\cdots,k)$ がRE1にマッチするかを判定する
   マッチしないならば, Falseを返して終了
5. すべてのテストが通る場合
   Trueを返して終了

図 5 マッチング・テストによる同一判定

4. まとめと今後の課題

本研究では, 正規表現学習支援システムとシステムにおける正規表現の同一判定の方法について述べた. 今後システムとこれらの同一判定機能を実装する予定である.

参考文献

[1] J.E.Hopcroft and R.M.Karp(1971): A linear algorithm for testing equivalence of finite automata. Technical Report 71-114, University of California.
[2] J. ホップクロフト. R. モトワニ. J. ウルマン. オートマトン言語理論　計算論Ⅰ[第2版], pp. 174-189, 2003.