Perl/CHISE による正規表現の拡張の試み
――文字素性による後方参照の実装実験と課題――

師　茂樹（花園大学文学部 s-moro@hanazono.ac.jp）

概要

現在、様々な分野で用いられている正規表現であるが、古典文献の分析などにおいては、符号化文字集合（文字コード）に依存した従来のメタ文字では機能不足である。本報告では、CHISE プロジェクトで公開している文字素性データベースと、Perl が持つ正規表現リテラルのオーバーロード機能を組み合せることによって実現した、文字の素性に対する後方参照メタ文字の試験的な実装について報告する。

問題の所在

現在、正規表現[1]は、Perl をはじめとする言語処理のみならず様々なテキスト・エディタ、検索アプリケーション等において実装され幅広く利用されている。その用途は、Web サーバの Log ファイルなどのような機械処理に適したテキストの処理だけでなく、エディタで書かれた原稿や電子メールなどの本文など、機械処理に必ずしも適していないテキストの処理にも多く用いられている。人文科学の分野では、それよりもさらに複雑であろう古典テキストの分析において正規表現が積極的に使われるようになってきている。古典テキストの分析においては、柔軟な検索ツールとして正規表現を用いることが現在のところほとんどであろうが、近年、テキストに隠された（と現代人が思っている）パターンを発見する手法に注目が集まっている[2]ことから、正規表現についてもテキスト・マイニング的なパターンマッチが可能かどうかについて模索されてきた。

筆者が研究対象としている漢字文献について言えば、従来ある正規表現でも、例えば、

```
(.)[^\1]\1[^\1]
```

という具合に後方参照[3]を用いた正規表現を用いれば、漢語表現でよく用いられる「無念無想」というようなパターンを抽出することが可能である。しかしながら、漢文中に見出される表現上のパターンはこのような単純なものだけではない。例えば「一日千秋」という成語であれば、

任意の漢数字.任意の漢数字.

という正規表現が書ければ（下線部はメタ文字を表す。以下同様）、ヒットさせることができるであろう[4]。すなわち、文字そのものではなく文字の属性によるパターンマッチである。このような用途に対しては、文字クラス […] を用いることで（煩瑣になろうが）比較的容易に実現されるであろうし、Perl 5.8 などで実装された Unicode のプロパティを指定できるメタ文字[5]は、そのような考え方を Unicode の定義する範囲

でサポートしたものと言ってよい。Unicode には残念ながら漢数字というプロパティはないが、Perl 5.8 であれば簡単なスクリプトを追加することで容易に拡張が可能である[6]。

もっとも、これでもまだ柔軟性に欠き、古典テキストの分析には不十分である。例えば「春眠暁を覚えず」で有名な孟浩然の漢詩は、

春眠不覺曉　處處聞啼鳥　夜來風雨聲　花落知多少

であるが、各詩句の最後の文字（曉、鳥、聲、少）の韻のパターンを調べると「○○●○」となっている。このような韻を踏むという作法は、言うまでもなく作詩する上でのもっとも基本的なルールである。これと同形式の韻文を、改行も句読点もないまったくの白文から抽出しようとする場合、

```
.{4}(.).{4}一番目の丸括弧の文字と同じ韻の文字.{4}一番目の丸括弧の文字と異なる韻の文字.{4}一番目の丸括弧の文字と同じ韻の文字
```

のように、文字の属性を参照可能な後方参照メタ文字が必要となってくる。同様に、

```
.(.).一番目の丸括弧の文字と同じ意味の文字
```

という正規表現で「何因何縁」などのような定型表現を抽出したりする用途などが考えられる。これまで気づくことのなかったパターンを発見することができれば、テキスト解釈の新たな可能性を開けてくることから、このような後方参照は漢字文献の分析に資するところが多いように思われる。しかし、現在ある正規表現は符号化文字集合（文字コード）に依存したものであるため、このようなパターンマッチを実現することは困難であろう。

本稿では、筆者が CHISE プロジェクトで開発中の Perl/CHISE の一機能として、文字の属性（CHISE プロジェクトでは素性とよぶ）による後方参照の試験的な実装を試みたことについて報告し、併せて問題点などを指摘する。諸賢の意見を乞う次第である。

Perl/CHISE とは

本題にはいる前に、CHISE プロジェクトおよび Perl/CHISE について簡単に説明しておくべきであろう。

CHISE プロジェクト

現在、計算機における文字のモデル化として一般化しているのは、言うまでもなく符号化文字集合によるものである。しかし、特に漢字処理においては、ISO/IEC 10646 や Unicode に代表される大規模な符号化文字集合が膨張し続けている（現時点で収録漢字数は 7 万字を超え、さらに 2 万字程度の追加について審議中）のにともない、文字の同一視や区別の問題はなし崩しにされ、しばしばユーザから遠く離れた政治的な場で決着を見ることも少なくない[7]。

このような現状を踏まえ、符号化文字集合に依存しないまったく新しい文字処理技術を目標に守岡知彦氏や g 新部裕氏らによって構想されたのが My Symbolic System や UTF-2000 モデルである。CHISE プロジェクト[8]は、守岡氏による後者（現在では Chaon モデルと呼ばれる）の実装である XEmacs UTF-2000（現在

では XEmacs CHISE）の開発に端を発する（[5]）。

UTF-2000 / Chaon モデルでは、文字を字形・発音・意味や、既存の文字コードのコードポイントなど、様々な素性（feature）[9]の集合によって表現する。例えば、現代の一般的な日本語で使われる「説」という文字と、現代中国語（普通話）で用いられる「说」という文字が、それぞれどのような文字素性の集合になるかを大雑把に図示すると図 1 のようになる。

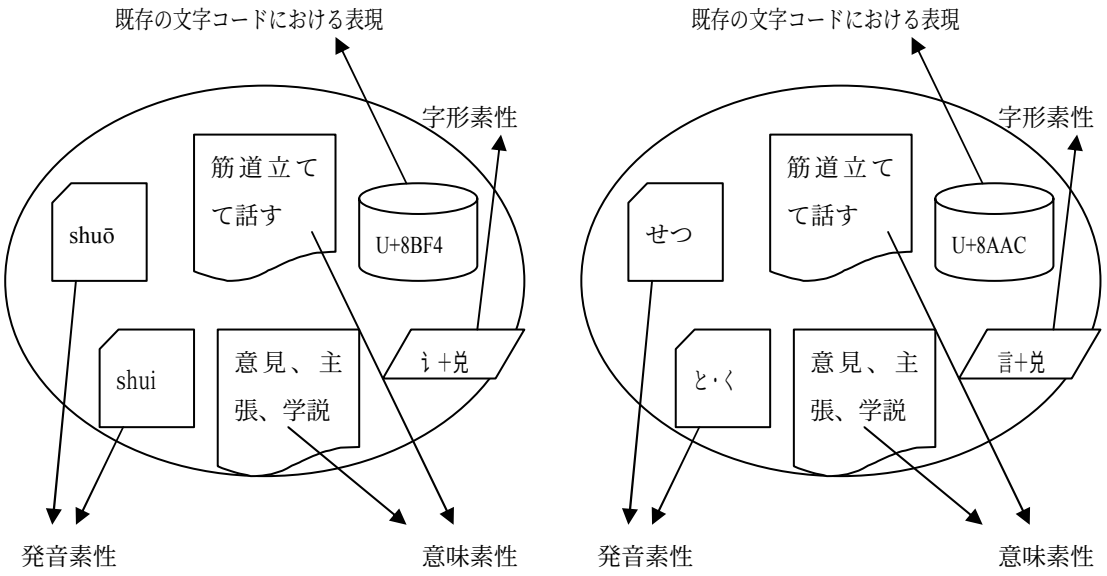

図 1　Chaon モデル

この二つの“文字”（丸で囲まれた部分）を比較すると、意味素性だけで言えばほとんど同じだが、字形素性どうしでは「兑」の部分が一致するも「言」と「讠」が（派生関係にあるが厳密に言えば）一致せず、発音素性や Unicode のコードポイントはまったく異なる、と言う具合になる。文字コードモデルのように単一のコードポイントのみで文字を表現しようとすると、文字は常に離散的な状況で処理されることになり、結果として異体字を包摂すべきか分けるべきか、というような二者択一的な議論に終始することになるが、実際にはコンテクストに応じて同一視される場合も区別される場合もあるのが文字の本来のあり方であろう。このモデルを用いれば、コンテクストによって注目する素性を変化させることで同一視することも区別することも可能であり、文字の実際のあり方を適切にモデル化することができるのではないかと考えている。また、Chaon モデルでは、文字素性を追加したり、新しい組み合わせを定義したりするによって、随時新しい文字が誕生させることもできる。文字コードモデルのように現存する文字に番号を振るのとはまったく逆の方向である。

ところで、前述の XEmacs による実装は、言うまでもなくエディタを中心とした統合環境であり、電子的な文書の作成、利用、管理、交換などにおいては十二分な機能を有するものの、例えば高品位な印刷要求には応えられないなど、文書処理全体をカバーできるわけではない。そこで、上記のような文字処理モデルを計算機環境全体に適用することを目指した CHISE プロジェクトがスタートすることになった。現在のところ、文字素性を蓄積、管理する文字データベース（XEmacs CHISE 附属のデータベース、漢字構造

情報データベース)、CHISE 環境で共有されるライブラリ libchise、文書編集環境兼データベースのフロントエンドである XEmacs CHISE、言語処理系の Ruby/CHISE と Perl/CHISE、高品位組版システムである Ω/CHISE、フォント自動生成システム KAGE などが開発、公開されており（[6][5])、Linux をはじめ Mac OS X などへの移植も同時に進められている。その成果のほとんどが CVS で公開され、メーリングリストを中心としたオープンな場での開発のほか、公開シンポジウムなども積極的に開催している。

Perl/CHISE

Perl/CHISE は筆者が開発している Perl における Chaon モデルの実装であり、モジュールとして提供されている[10]。文字データベース（現状では XEmacs CHISE 用の Berkeley DB 版文字データベース）を Perl から直接参照することが可能であるほか、文字オブジェクトの集合としての側面に焦点を当てて開発を進めているため、現時点では実験的な性格が強く実用性は乏しい。以下、若干のコード例を示す。

```
# sample 1
use CHISE;
# 諸橋大漢和辞典 6942 番という素性を持つ文字オブジェクトを生成
my $s1 = CHISE->new(daikanwa => 6942);
# 文字素性をすべて表示
print $s1->dumpAttr;
# 総画数を表示
print $s1->strokes, "\n";

# sample 2
use CHISE;
# 新たに文字オブジェクトを定義
my $s1 = CHISE->define_char(strokes => 12, radical => 9);
# 文字素性集合の重なり具合を比較
my $s2 = CHISE->define_char(strokes => 12, radical => 9, daikanwa => 694);
my $c = $s1->compare($s2);
if ($c == $CHISE::EXCLUSIVE) {
  print "排他的\n";
} elsif ($c == $CHISE::HAVE_INTERSECTION) {
  print "共通部分あり\n";
} elsif ($c == $CHISE::PROPER_SUBSET) {
  print "完全部分集合\n";
} elsif ($c == $CHISE::PROPER_SUPERSET) {
  print "完全上位集合\n";
} elsif ($c == $CHISE::EQSET) {
  print "完全一致\n";
```

```
}
```

文字素性による後方参照の実装実験

以下、Perl による文字素性を用いた後方参照の実装について説明する。今回は、従来ある Perl/CHISE のモジュール（CHISE.pm）内ではなく、別モジュール（CHISE_REG.pm）として実装している。その理由は、Perl 5.8 が持つ正規表現リテラルのオーバーロード機能を用いて Perl の正規表現の一部拡張するという方法をとった結果、これが CHISE.pm の他の部分に悪影響（動作不良、速度低下など）を及ぼすからである。

また、実際にデータベースをアクセスする機能については、CHISE_REG.pm とは独立したスクリプト（chisereg.pl）で行っている。その理由は、現在、Berkeley DB にアクセスするために Perl の標準モジュールである DB_File モジュールを用いているが、正規表現リテラルのオーバーロード機能を用いることによって正常に動作しない場合があったためである（筆者の環境では Bus Error が頻発した）。これは BerkeleyDB モジュールでも同様であった。この点については、今後、データベースへのアクセスを libchise 化することによって解決されると思われるため、暫定的な実装であることをお断りしておきたい。

さて、本稿執筆時点では、次のメタ文字のみ利用可能である。

メタ文字	意味
`\same_`*`feature`*`_`*`n`*	*n* 番目の丸括弧の組に対応するすでにマッチした文字と、同じ素性 *feature* と値の組を持つ文字にマッチ

例えば、以下のサンプル・スクリプト[11]では、「山」「川」ともに同じ総画数を持つため「matched!」という出力が得られる。現在のところ、UTF-8 の文字列のみに対してパターンマッチが可能である。

```
#!/usr/bin/perl -w
use CHISE_REG;
use utf8;
#use re "debug";

if ('山川' =~ /(.)\same_total-strokes_1/) {
  print STDERR "matched!\n";
} else {
  print STDERR "unmatched...\n";
}
```

上のスクリプトの場合、CHISE_REG.pm の内部では、

1. オーバーロードによって、正規表現リテラル中の「`\same_total-strokes_1`」を、CHISE_REG.pm 内のサブルーチンを呼び出す正規表現「`(??{CHISE_REG->chise_backref($1, 'total-strokes')})`」に置換。
2. サブルーチン `chise_backref` の内部では、

i. 第 1 引数の文字（上の例では「山」）と、第 2 引数で表される素性の値を標準入力で chisereg.pl に渡す。

ii. chisereg.pl 内部においては、

① 渡された引数に基づいて、文字素性データベースにアクセスする。上の例では 3 が得られる。

② 逆に第 2 引数で表される素性を持つ文字の中から、①で得られた値を持つものを検索する。上の例では「山」「川」「三」などが得られる。

③ 得られた文字集合の UCS 値をデータベースから求め、文字クラス（`[\x{5c38}\x{5e72}\x{5dfe}…]`[12]）として返す。`\x{…}` に変換するのは、CHISE_REG.pm と chisereg.pl 間で文字のやり取りをする際、UTF-8 として認識されない場合があるためである。

3. 最終的に得られる正規表現は「`/(.)[\x{5c38}\x{5e72}\x{5dfe}…]/`」となり、`$target` とマッチする。

という手順で処理が行われる。

ちなみに、*feature* の指定を、文字コードに関する素性にすると、通常の後方参照と同等の機能が実現される。

課題と問題点

以上、ごく初期段階の実装を簡単に説明したに過ぎないが、冒頭に述べた問題をある程度克服する可能性を示せたのではないかと思う。しかし、言うまでもなく、現時点では課題も多く、また極めて重要なものを含む多くの問題が指摘しうる。以下、思い付く点について列挙したい。

今後の予定

本稿には間に合わなかったが、以下の機能を随時追加する予定である。

- 素性指定の多元化

現在は、単一の素性についてのみパターンマッチが可能であるが、複数の素性を指定出来るようにする（例えば、一番目の丸括弧の文字と同じ総画数と部首をもつ文字のように）予定である。しかし、その表記方法については、煩瑣になる可能性が高いため、検討を要する。

- 文字素性によるマッチング

Perl 5.8 などで実装された Unicode のプロパティによるメタ文字に相当するものを、CHISE の文字素性を用いて実装する。これも素性指定を複数可能にする予定であるが、上と同様の課題がある。

- エラーチェックの強化、特に通常の正規表現との整合性の確保

文字素性の不足

これは Perl/CHISE 自体の問題ではないが、今回の実装の有用性は文字データベースに依存している点には留意したい。特に古典テキストの分析での使用を考えた場合、現在の文字データベースは符号化文字集合や部首、構造情報などについては充実しているものの、音韻や意味、歴史的な情報などについては決定

的に不足しており、実際の分析にはまだ使えるような状態ではない。今後、データベースの充実が望まれる[13]。

Perl の問題

本実装は、Perl 5.8 の正規表現リテラルのオーバーロード機能に依存しているため、Jeffrey Friedl 氏が指摘する、

- オーバーロード機能が正規表現リテラルのリテラル部分だけに適用され、展開された部分には適用されない
- オーバーロード処理には正規表現に適用されている修飾子を知るすべがない

などの問題点（[1], 7.8.7）をそのまま継承している。今後、実装が進み、複雑な処理をする段階になって、問題が顕在化する可能性がある。

モデルとしての限界

Chaon モデルは文字を素性の集合としてモデル化したものであり、前述のごとくコンテクストに応じてふるまいを変えることが可能である。しかし、コンテクストを表現するのは文字のレイヤーではなく、例えば XML に代表されるマークアップ言語など、別のレイヤーにおいて記述されなければならない（[6]）。したがって、文字素性に着目した今回の試みも、現時点ではコンテクストを考慮することができない。

例を挙げれば、前掲した孟浩然の詩を邱奕南氏作成の MS-DOS 用ソフトウェア「絶律平仄檢驗程式」[14]で分析すると、以下のような結果が得られる。

```
絶律平仄檢驗程式 1.0V - 邱奕南

詩句：
  春眠不覺曉
  處處聞啼鳥
  夜來風雨聲
  花落知多少
平仄：
  平平？仄仄
  仄仄？？仄
  仄平平仄平
  平仄平平仄
韻部：
  曉 - 上聲 17 篠韻
  鳥 - 上聲 17 篠韻
  聲 - 下平 8 庚韻
  少 - 上聲 17 篠韻 去聲 18 嘯韻
```

この出力結果の中、平仄の分析において「？」となっている部分は、文脈によって発音が変化する漢字

であるため、文字単位での分析では判断ができなかったことを示す。つまり、今回の試験的な実装においても、上の「？」に相当する文脈に依存する文字については適切な動作が期待できないのである。

文字列やコンテクストをどのように処理するかについてはCHISEプロジェクト全体の課題でもあるため、今後議論を深めていかなければならないだろう。

謝辞

本報告は、科学研究費補助金・基盤研究 C「次世代中国古典文献データベース構築の基礎的研究」（課題番号 14510494、研究代表者: 村越貴代美慶応大学助教授）による成果の一部である。

参考文献

[1] Jeffrey E. F. Friedl. *Mastering Regular Expressions, 2nd Edition.* O'Reilly. 2002. 田和勝訳『詳説正規表現 第 2 版』（オライリー・ジャパン、2003）

[2] 漢字文献情報処理研究会編『電脳中国学 II』（好文出版、2001）

[3] 川幡太一「新 ISO/IEC 10646 と Unicode の漢字を検証する」（『漢字文献情報処理研究』2、2001）

[4] 近藤泰弘「コンピュータによる文学語学研究にできること　―古典語の「内省」を求めて―」（全国大学国語国文学会夏季大会シンポジウム「情報技術は文学研究をいかに変えるか」要旨、2001、http://klab.ri.aoyama.ac.jp/public/paper/20010602.pdf）

[5] 守岡知彦・江渡浩一郎・苫米地等流・宮崎泉・師茂樹「CHISE Project」（『漢字文献情報処理研究』4、2003 年 10 月）

[6] 師茂樹「タグ付き言語と文字コード」（小林龍生・安岡孝一・戸村哲・三上喜貴編『インターネット時代の文字コード』、共立出版、2001）

[7] 師茂樹「GB18030 とは何か　大陸の戦略」（『漢字文献情報処理研究』2、2001）

[8] 師茂樹「ソフトウェアレビュー　CHISE プロジェクト」（『漢字文献情報処理研究』3、2002）

---

[1] 本稿で言う「正規表現」は、数学的に定義された狭義の用語ではなく、実際にはこの定義を越えているが「正規表現」という名前で実装されているものを指す広い意味での用語である。

[2] 近藤みゆき氏とともに N グラムモデルを用いた新たな研究方法を提案した近藤泰弘氏が、「情報技術によってのみ可能な古典研究」としてあげる次の 2 点は重要である。「1. 徹底的に網羅的な研究（すべての単語・すべての文字の単位にまで網羅性を及ぼすことが可能になる）。2. それによって現代人には通常認知できないデータの構造性や規則性を探り出す。それは、現代人の古典語に対する「内省」(introspection)（語感）の欠如を補うことができ、文学研究に貢献する。なぜなら、古典文学の正しい読みにとって、「内省」（文法的直観と言語外知識など）の欠如は大きな障害のひとつだからである。」（[4]）

[3] backreference の訳語についてはゆれがあるが、ここでは「後方参照」とする。[1]日本語訳 p. 19 の訳注参照。

[4] 厳密に言えば、この正規表現では「一二三四」などの意図しない文字列にもヒットしてしまうので、実際には Perl 的に書けば「`\p{漢数字}\P{漢数字}\p{漢数字}\P{漢数字}`」などとしなければならないだろう。

[5] [1], 3.4.2.4 以下参照。

[6] Perl 5.8 付属の perldoc perlunicode 参照。

[7] 文字コード標準化における政治的な側面については[3][7]を参照。

[8] http://cvs.m17n.org/chise/, http://www.kanji.zinbun.kyoto-u.ac.jp/projects/chise/　CHISE は " CHaracter Information Service

Environment ”の略。

[9] 従来これを表す用語は「属性（attribute)」であったが、R. ヤーコブソンによる弁別素性（distinctive features）理論に基づいて、議論の末「素性（feature)」に改められた。ヤーコブソンの音素論と Chaon モデルとの類似性（および、Unicode における character の定義と、プラーグ学派における音素の定義との類似性）をご指摘いただいた山崎直樹氏（大阪外国語大学）に感謝申し上げたい。

[10] Perl/CHISE の CVS レポジトリは cvs.m17n.org の/cvs/chise/perl になる。ちなみに、/cvs/chise/perl/Chise_utils 以下は宮崎泉氏・苫米地等流氏にかかるΩ/CHISE 用のツール群が収められており、Perl/CHISE とは直接的な関係はない（がコードは大いに参考にさせていただいた）ので注意されたい。

[11] CVS レポジトリ中の sample3.pl 参照（ただし、今後内容が変更されることは大いにあり得る)。

[12] ここでの「…」は一部省略を表し、正規表現で任意の文字を表すドットではない。次も同じ。

[13] 現在筆者は、川幡太一氏らとともに、より適切な文字処理のためのデータベースである Kanji Database Project を進行中である（http://kanji-database.sourceforge.net/、http://kura.hanazono.ac.jp/kanji/kanjidb.html)。

[14] [2]の付録 CD-ROM に収録されている。