1J-07

Ruby における Unicode プロパティ処理の拡張と効率化

小山 拓美　　松原 俊一　　Martin J. Dürst
青山学院大学理工学部情報テクノロジー学科

1 はじめに

オブジェクト指向スクリプト言語 Ruby は多言語のテキスト処理, 強力な文字列走査や正規表現検索の機能がある. Ruby における多言語のテキスト処理機能は他スクリプト言語の Perl や Python に対して性能面で劣らないことが示された [1]. Unicode は文字の属性情報 (プロパティ) を定めており, UCD (Unicode Character Database) に Unicode プロパティとして定義されている [2]. 多言語のテキスト処理では正規表現の文字クラスとして, Unicode プロパティが使用可能である. 例えばテキスト中の大文字や平仮名のみを抽出する処理ができる. しかし Ruby では多値プロパティのサポートが少ない.

本研究では Ruby で使用できる Unicode プロパティを追加した. また, Unicode プロパティのデータ構造を新たに作成した. これにより, 多値プロパティのサポートを容易にし, 高速化と省メモリ化を実現した. さらに, 文字からのプロパティ値の特定も可能とした.

2 Ruby における Unicode プロパティ

Ruby には Ruby 2.0 に正規表現エンジンとして Onigmo が導入され, Unicode プロパティが正規表現の文字クラスとして使用可能となった.

2.1 現実装

現実装ではプロパティ値ごとのデータをそれぞれ反転リストと呼ばれる配列で表している. 反転リストは昇順に格納された数値範囲の組合せを格納している. コードポイントが連続して同じプロパティ値を持つことが多いことからこのデータ構造が用いられてきた. プロパティ値の反転リストの各組合せの数値範囲内に文字があるかどうか二分探索することで, プロパティ値の有無を判定している.

2.2 現実装の問題点

プロパティ値それぞれが反転リストを持つため, プロパティ値の種類が増えると消費メモリが多くなる. また, 反転リストによっては探索範囲が大きくなり, 処理効率が悪くなる. そこで 1 回の処理でマッチングができる実装を提案する.

3 Unicode プロパティの新実装

14 種類の多値プロパティのサポートを追加した. また, プロパティのデータ構造を新たに作成した. Unicode では各文字がプロパティ値を持っている. しかしプロパティ値がすべて共通の文字が多い. Unicode 9.0 では 1,114,112 個のコードポイントの内, 128,172 文字収録している. サポート範囲を拡大した上での Ruby における Unicode プロパティのサポート状況では, コードポイントそれぞれが持つプロパティ値すべての組合せ (同値類) はたったの 3747 種類となる.

3.1 同値類

プロパティ値すべてを 192 ビットで表現する. 反転リストに代わるプロパティ値のデータは 3747 個の要素を持つ一つの配列で作成する. また, プロパティ値がどのビットで表現するかを示すための配列を作成する. 図 1 にプロパティ値の有無の判定例を示す. 図の mask はプロパティのビット領域を示し, value はプロパティ値を表現するビット列である.

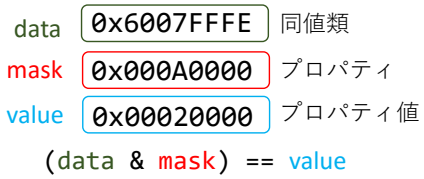

図 1: プロパティ値の有無の判定例

3.2 DAG

コードポイントがどの同値類に属するかを示すインデックスが必要である. しかしそのインデックスをコードポイントごとの配列にすると膨大となる. 文字は文字体系ごとに割り当てられているため, 同値類となっている文字が集まりやすい. そこでコードポイントを 256 個ずつに区切ったものをスライスとした. スライスへの参照をインデックスとし, スライスの要素が他のスライスと同一になる場合インデックスの参照も同一にできるため, データ圧縮が可能. 次ページ図 2 にその結果でできた DAG (有向無閉路グラフ) を示す. コードポイントの上位 13 ビットと下位 8 ビットを用いることでスライスから同値類を

Improvement of Efficiency and Coverage of Unicode Property Processing in Ruby
Takumi Koyama, Shunichi Matsubara and Martin J. Dürst
Department of Integrated Information Technology, College of Science and Engineering, Aoyama Gakuin University
5–10–1 Fuchinobe, Chuo-ku, Sagamihara, Kanagawa 252-5258, Japan
duerst@it.aoyama.ac.jp

割り出せる．結果，すべて異なる要素の場合 1,114,112 / 256 で 4352 個のスライスができるが，スライス数は 210 個まで減少した．よって DAG を使用したことでデータ量を約 20 分の 1 に削減できた．

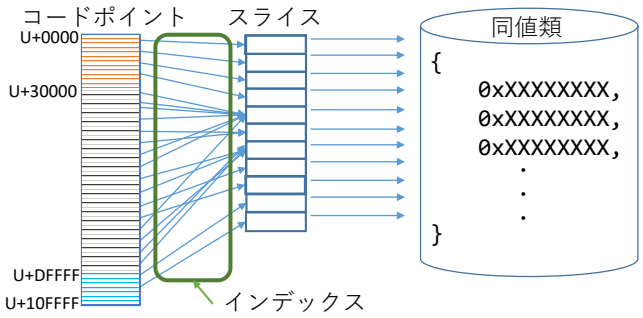

図 2: DAG の構造

3.3 文字からのプロパティ値の特定

本研究の実装で，文字がどのようなプロパティ値を持つかわかるようになった．よって文字からのプロパティ値の特定が可能となった．

4 評価

本研究で実装した Unicode プロパティの正規表現でのマッチングについての処理速度と消費メモリを評価するために，既存の Ruby と比較した．使用した既存の Ruby のバージョンは Ruby 2.5.0 dev (trunk 57204) である．

4.1 サポートされたプロパティの数

サポートされたプロパティの数を表 1 に示す．多値プロパティのサポートが増え，プロパティ値の数は約 2 倍になった．

表 1: サポートされたプロパティの数

	旧実装	新実装
二値プロパティ	55	55
多値プロパティ	7	21
プロパティ値	553	1009

4.2 処理速度

処理時間の計測を Ruby の標準ライブラリの benchmark を用いて行った．処理時間は 3.33 GHz Intel Core i5 Windows 10 Enterprise 64 bit OS で動作する x86_64 Cygwin で計測した．それぞれの実装の一文字のプロパティ値の有無をチェックする関数に限定して処理時間を計測した．U+0061 ～ U+1F600 内に分布している 10 個の文字に対し，1 種類の Unicode プロパティのプロパティ値の有無を判定する．合計 10 文字の処理を 1000 万回行ったときの処理時間を表 2 に示す．結果から，既存の実装は反転リストの要素数が多くなるほどマッチングの時間が増加する．それに対し，本研究の実装はプロパティ値に関わらず旧実装におけるどのプロパティ値の時間よりも速い．

表 2: 処理時間の比較 (単位秒)

プロパティ名	反転リスト範囲数	旧実装	新実装
In_Hiragana	1	1.438	1.015
Hiragana	4	2.110	1.187
Han	16	3.843	1.016
Greek	36	4.625	1.016
Ps	75	5.859	1.063
Common	175	8.078	1.157
Age=3.0	369	9.125	1.188

4.3 消費メモリの削減

本研究では Unicode プロパティのデータ構造を見直したことで消費メモリの削減も実現した．表 3 にメモリ消費量を示す．

表 3: メモリの比較 (単位バイト)

	内訳	データ量	合計
新実装	インデックス	4,352	213,920
	スライス	107,520	
	同値クラス	89,928	
	プロパティ値	12,120	
旧実装	反転リストの大きさ	2,136	240,976
	数値範囲の組合せ	238,840	

5 まとめと今後の課題

本研究では Ruby で正規表現の文字クラスとして使用できる Unicode プロパティの範囲を拡大した．同値類となる文字が多いことに着目し，反転リストでの二分探索を DAG でのテーブル探索に置換えたことにより，高速化と省メモリ化を実現した．

本研究では Ruby における Unicode プロパティすべてをサポートしていないため，範囲のさらなる拡大は今後の課題である．例えば Name のような文字それぞれに別の値が与えられるプロパティは，別の探索アルゴリズムを考える必要がある．

参考文献

[1] 松本行弘. Ruby における実用的な多言語処理の実装. 情報処理学会論文誌プログラミング（PRO）, Vol. 2, No. 2, pp. 27–36, mar 2009.

[2] Mark Davis and Ken Whistler. Unicode character database. Unicode Standard Annex #44, The Unicode Consortium, 2016. Unicode 9.0.0.