6L-9

プログラミング言語 Ruby のための Unicode 正規化ライブラリの効率化

野島 歩 † 松原 俊一 †† Martin J. Dürst ††

† 青山学院大学大学院理工学研究科理工学専攻
†† 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

インタプリタ言語における国際化処理には効率的な実装が重要である．本研究では Ruby の高度な組込み機能を活用し，Unicode 正規化 [1] の効率化を実現する．

Unicode 正規化は文字列の内部表現を統一する処理である．Unicode 正規化の結合パターンは無数に存在する．しかし，実際は自然言語によってテキスト中に現れる結合パターンは数種類から数千種類に限定される．本研究ではこの特徴に着目し，結合パターンをキャッシュし，無駄な正規化処理を削減した．また，変換処理に Ruby の強力な標準メソッドを利用することで高速化を図った．その結果，既存の Ruby 用ライブラリ [2] [3] の性能を大きく上回り，コンパイルされた言語の実装に近い速度を実現した．

2 Unicode 正規化

2.1 正規化のパターン

Unicode の文字には，数通りの表現が一部存在する．

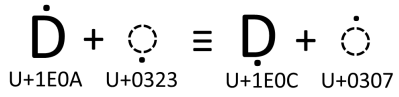

図 1: 合成文字と結合文字

図 1 のように，「Ḍ̇」という文字を表す場合，少なくとも二つの手法がある．一つは左のように，「Ḍ」(U+1E0A) に「◌̣」(U+0323) を合成する手法である．一方で右のように，「Ḍ」(U+1E0C) に「◌̇」(U+0307) を合成する手法もある．テキスト処理において表示は同じでも内部表現が異なるものが混在すると，ユーザや開発者の意図に反した動作が起きてしまう．これは Unicode 正規化 [1] で表現を統一することで解消できる．

「Ḍ̇」や「Ḍ」のように合成のベースとなる文字を基底文字，「◌̣」(U+0323) や「◌̇」(U+0307) のような基底文字に結合する文字を結合文字という．本研究では以後，Unicode 正規化の対象となる文字列パターンを結合パターンと表現する．結合パターンは基底文字と結合文字の組み合わせで構成される．

2.2 正規化形式

Unicode 正規化は，文字列表現の統一方法として四つの正規化形式 NFD, NFKD, NFC, NFKC を定めている．NFD, NFKD は分解された形で統一され，NFC および NFKC は合成された形で統一される．また，NFKD, NKFC では互換文字も正規化される．互換文字とは，「㍍」や「①」のように，幅や形，位置，角度が通常と異なる文字である．例えば，「㍍」は「メートル」に，「①」は「1」に正規化される．

3 効率化手法

3.1 Unicode 正規化実装の問題点

正規化処理は，文字列走査，分解，テーブル参照，合成など処理が多い．しかし，一般的なテキストにおいて結合パターンが含まれることは少ない．また，多種類の結合パターンが利用されるテキストはさらに少ない．実際のテキストは，ある一定の結合パターンが複数回現れることが多い．なぜなら，一つの自然言語で利用される文字は限られているからである．

以上の理由から，テキスト全体を入力として Unicode 正規化を行うことは，二つの点から非効率的である．一つは，テキストに含まれる結合パターンが少ないのにも関わらず，すべての文字を走査して正規化を試みる点である．二つ目は，テキストに含まれる結合パターンの種類には偏りがあるのに，同じパターンを何度も正規化する点である．この二点はオーバヘッドが大きい．

3.2 効率化手法の提案と実装

効率化手法の概要を図 2 に示す．これは「A ̃Ḍ ̣A ̃」を入力とした場合の処理フローを示している．効率化の中心的な着想は `gsub` メソッドと正規表現，およびハッシュ (連想配列) を応用した点である．

3.2.1 `gsub` メソッドの応用

`gsub` は，Ruby における `String` クラスの標準メソッドであり，正規表現を用いて，マッチするすべての文字列を検索・置換する．

Optimization of a Unicode Normalization Library for Ruby

Ayumu Nojima †, Shunichi Matsubara †† and Martin J. Dürst ††

†Graduate School of Science and Engineering, Aoyama Gakuin University
††Department of Integrated Information Technology, College of Science and Engineering, Aoyama Gakuin University
{nojima}@sw.it.aoyama.ac.jp,
{matsubara, duerst}@it.aoyama.ac.jp

Copyright ©2013 Information Processing Society of Japan.
All Rights Reserved.

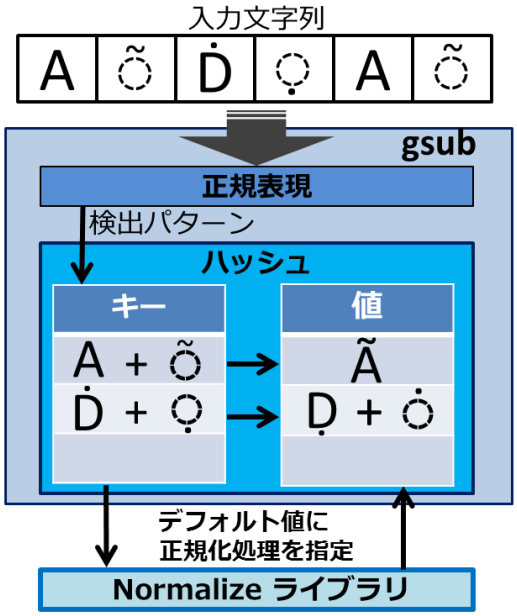

図 2: 効率化手法の概要

本研究では，正規化形式ごとにすべての結合パターンを検出できる正規表現を生成した．この正規表現は，Ruby を用いて Unicode のデータファイルから自動生成される．大きさは 49.4 KB である．

gsub は一般的に置換文字列を指定して利用する．しかし，Ruby1.9 において gsub で指定する置換文字列の代わりにハッシュを用いることが可能になった．ハッシュを用いることで，検出したパターンに対応する文字列で置換できる．これは Ruby 独自の強力な機能である．

この機能を用いて，テキスト中の結合パターンを検出し，検出パターンに対応する正規化後の文字列で置換する．gsub は Ruby 内部で C 言語で実装されているので，正規化の実装は純粋な Ruby でありながら高速な検出・変換処理が可能になる．

3.2.2 ハッシュの作成

Ruby 1.9 のハッシュはキーが存在しない場合の規定値として，ブロックと呼ばれる特定の処理を指定できる．これも Java や Perl のハッシュには無い Ruby 独自の高度な機能である．

この機能を応用し，キーが存在しない場合，そのキーを正規化した文字列を値として返し，自身に追加するハッシュを作成した．gsub で検出した結合パターンがハッシュに存在しない場合，図 2 下部のように検出パターンを Normalize ライブラリ [4] で正規化して返す．このとき結合パターンは，ハッシュにキャッシュされ，gsub は正規化された文字列に置換する．この手法により，一度正規化したパターンは以降，冗長な正規化処理をせずに変換することができる．

4 評価

本研究の効率化を用いて実装したライブラリと他の Ruby 向け Unicode 正規化ライブラリを用いて性能比較を行った．比較対象のライブラリとして，Twitter_CLDR [2] と UnicodeUtils [3] を利用した．両者とも Ruby のみで実装されている．比較処理に用いたデータは，ドイツ語[1]，ベトナム語[2]，韓国語[3]版 Wikipedia から無作為に選んだ記事で，それぞれ 88.7 KB，71.9 KB，58.7 KB のテキストである．ドイツ語は，まれに結合パターンが出現する．ベトナム語は合成文字が多く，結合パターンが出現する頻度が多い．韓国語はすべての文字が正規化可能である．表 1 に各ライブラリにおける各正規化形式の平均処理時間をまとめた．

表 1: 各ライブラリの処理時間 (秒)

ライブラリ	ドイツ語	ベトナム語	韓国語
本ライブラリ	0.0275	0.0273	0.0388
Twitter_CLDR	3.8960	3.5410	3.4595
UnicodeUtils	0.0665	0.0585	0.0430

本ライブラリはすべてのデータにおいて他のライブラリの処理速度を大きく上回っている．特に NFC に効率化の効果が大きく表れていた．これは，NFC で行われる文字列の走査や先読み，結合など複雑な処理を大幅に削減できるからである．

5 まとめ

本研究では Ruby の高度な組込み機能を応用して Unicode 正規化の効率化を実装した．その結果，他の Ruby 用ライブラリの性能を大きく上回ることができた．

今後は本ライブラリをオープンソース化し，Ruby の標準ライブラリとして組み込むことを目標とする．また現在，本ライブラリの応用として Ruby 用 IRI/URI ライブラリを開発している．

参考文献

[1] Mark Davis, Ken Whistler, and Martin J. Dürst. Unicode Normalization Forms. Unicode Standard Annex #15, The Unicode Consortium, 2012. Unicode 6.2.0.

[2] Cameron Dutro. *twitter-cldr-rb*. http://github.com/twitter/twitter-cldr-rb, 2013 年 1 月 11 日閲覧.

[3] Stefan Lang. *Unicode Utils - Unicode algorithms for Ruby 1.9*. http://unicode-utils.rubyforge.org/, 2013 年 1 月 11 日閲覧.

[4] 野島歩, 松原俊一, Martin J. Dürst. プログラミング言語 Ruby のための Unicode 正規化ライブラリの実装と評価. 情報処理学会第 73 回全国大会, 2011.

[1] http://de.wikipedia.org/wiki/FC_Bayern_München
[2] http://vi.wikipedia.org/wiki/Th%C3%A0nh_ph%E1%BB%91_H%E1%BB%93_Ch%C3%AD_Minh
[3] http://ko.wikipedia.org/wiki/%EC%95%BC%EA%B5%AC

Copyright ©2013 Information Processing Society of Japan.
All Rights Reserved.