5L-4

日本語プログラミング言語における動的型付けに関する処理手法の改良

東海林　薫†　　筧　捷彦‡　　馬場　祐人††

早稲田大学基幹理工学部情報理工学科† 早稲田大学理工学術院‡ 早稲田大学大学院基幹理工学研究科情報理工学専攻††

1.背景・目的

日本語プログラミング言語とは，日本語の文章の形式に沿って記述を行うプログラミング言語である．日本語プログラミング言語の多くはインタプリタ方式のスクリプト言語として開発されているため，一般に実行速度が遅いことが課題点となっている．

そこで本研究では，日本語プログラミング言語プロデル[1]において明示的に型宣言がされていない変数(型不定変数)に対して，実行時に行うフィールドの取得・設定や実行メソッドの決定を高速化することで，実行速度を改善することを目的とする．

2.日本語プログラミング言語「プロデル」

プロデルは，スクリプト型の言語である．プロデルのソースコードの一例を図 1 に示す．

```
対象は、5
対象を処理したものを表示する //「120」が表示
対象は、「こんにちは」
対象を処理したものを表示する //「こんにちは！」が表示

［値:数値］を、処理する手順
  もし値が 0 なら 1 で抜ける
  値×（（値-1）を処理したもの）で抜ける
終わり

［値:文字列］を、処理する手順
  値&「！」で抜ける
終わり
```

図 1　プロデルのソースコード

2.1. 型不定変数のフィールドの取得・設定

プロデルでは，変数のフィールドを，「動物の尻尾」といったように助詞“の”を用いて表現する．もし，この“動物”が型不定変数であったとすると，“尻尾”というフィールドが“動物”に存在するかどうかやその属性をコンパイル時に知ることができない．

そこで，コンパイル時には，実行時に型を確認して該当のフィールドにアクセスするコードを生成することになる．

2.2. プロデルにおけるメソッド呼び出し

プロデルのメソッド呼び出し文は動詞の前に 0 個以上の補語がつく文としている．補語にはメソッドの引数となる式に助詞がついた実補語と，メソッド名の一部分となる形式補語の 2 種類が存在する．呼び出し文の構文解析の例を図 2 に示す．

プロデルでは，呼び出し文がもつ動詞・形式補語・実補語に含まれる式のクラスと助詞の組が過不足なく一致するようなライブラリ内のメソッドを実行する．

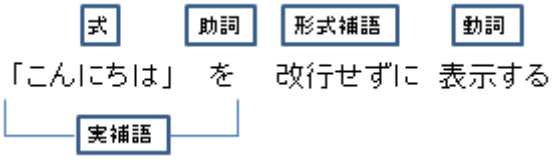

図 2　構文規則の適用例

3.既存コンパイラ「シンプルプロデル」

プロデルの簡易版コンパイラ「シンプルプロデル」の生成するコードの実行効率を上げる実験を行った．

3.1. コンパイル・実行

シンプルプロデルは次に示す方式で型不定変数に対処する．

- 実行時関数ライブラリを用意して，それらの関数を呼び出す形のコードを生成する．
- プログラム中のすべてのクラスについて，そのメンバの情報リストやメソッドのリストを生成し，実行時関数が使えるようにしておく．

3.2. 型不定変数のフィールドの取得・設定

生成されるコードは型不定変数の表すオブジェクトと，推定されているフィールド名とを引数として取得または設定を行う実行時関数を呼

Improving run-time operations for dynamically typed variables in a Japanese programming language

†Kaoru SHOJI, School of Fundamental Science and Engineering, Waseda University

‡Katsuhiko KAKEHI, Faculty of Science and Engineering, Waseda University.

††Yuto BAMBA, Graduate School of Fundamental Science and Engineering, Waseda University.

び出す．実行時関数はオブジェクトからの型情報を基に，そのオブジェクトのクラスにそのフィールドが存在するかどうか検索する．この検索は，継承関係を遡って再帰的に行い，該当するアクセスを行うコードを得る．こうして得られたコードを実行することで，フィールドの取得・設定が行われる．

3.3. メソッドの決定

メソッドの決定を実行時に行うのは呼び出し文の引数に型不定変数が含まれる場合である．生成されるコードは呼び出し文の動詞名と補語情報を引数としてメソッド決定を行う実行時関数を呼び出す．実行時関数はメソッドライブラリからその動詞の多重定義リストを取得して，その中から補語情報が全て一致するようなメソッドを検索する．このようにしてメソッドの決定が行われる．

4. 提案手法・実装

既存コンパイラの実行時関数に手を加えて，型不定変数に対する実行時間の短縮を試みた．

4.1. フィールドの取得と設定

実行時関数が検索して得たフィールドの取得・設定用のコードをクラスのメンバの情報内にキャッシュするように改造した．したがって，実行時関数は，まずキャッシュを調べてコードがキャッシュされていればそれを使うようになる．

4.2. メソッドの決定

実行時関数は，ある呼び出し文から初めてメソッドの決定をするときに，引数の補語情報の型不定変数が関与しない情報を基に多重定義リストからメソッド候補を絞り込んだリストを作成しておく．2 回目以降はこの作成したリストからメソッドの検索をすることで，検索対象となるメソッドの絶対数が減る．加えて，メソッドの検索処理を簡略化して，型不定変数が関わる部分のみを調べるように改訂した．

5. 評価実験

実装した機能の性能評価を行うために，改良前と改良後のシンプルプロデルについて，次のような評価実験をした．

5.1. 実験 1

あるクラスのフィールドを計 1000 万回取得するプログラムを実行し，その実行時間を計測する．なお，取得するフィールドは，以下の 3 つの場合を想定した．

① 1 つのフィールドを取得
② 2 つのフィールドを交互に取得
③ 2 つのフィールドを 2 回ずつ交互に取得

5.2. 実験 2

1 つのメソッドを 5 万回呼び出すプログラムを実行し，メソッドの決定処理にかかった総時間を計測する．なお，対象のメソッドは，以下の 4 つの場合を想定した．

① 要素数 1 個の多重定義リストのメソッド
② 要素数 20 個の多重定義リストの先頭メソッド
③ 要素数 20 個の多重定義リストの末尾メソッド
④ 要素数 20 個の多重定義リストの末尾メソッド (絞り込みの効果がなかった場合)

5.3. 実験結果

実験 1，実験 2 について，それぞれ 10 回計測したときの平均値を表 1，表 2 に示す．実験に用いたほぼすべてのプログラムに対して改良後の実行速度が向上していることがわかる．改良前よりも実行速度が落ちている場合もあるが，許容範囲内の速度低下であるといえる．

表 1　実験 1 の評価実験結果

取得フィールド	改良前 ms	改良後 ms
1 つのフィールド	1884.6	571.1
2 つのフィールド(交互)	2260.6	1137.3
2 つのフィールド (2 つずつ交互)	2204.0	948.5

表 2　実験 2 の評価実験結果

対象メソッド	改良前 ms	改良後 ms
多重定義 1 個	46.7	52.5
多重定義 20 個(先頭)	47.2	56.3
多重定義 20 個(末尾)	364.7	112.4
多重定義 20 個(末尾) (絞り込みなし)	466.9	239.4

6. まとめ

今回の研究で，動的型付けに関する処理手法を改良することで，実行速度の改善を実現することができた．今回はシンプルプロデルに実装したが，インタプリタ版のプロデルにも同様の効果があげられると思われる．ただし，実験に用いたコードは非常に単純なものであり，一般的なプログラムにおいても実行速度の向上があるか調査することが今後の目標と考えている．

参考文献

[1]日本語プログラミング言語「プロデル」，閲覧日:2013/01/09，http://rdr.utopiat.net/
[2] C# によるプログラミング入門，閲覧日:2013/01/09，http://ufcpp.net/study/csharp/index.html
[3]馬場祐人，筧捷彦:日本語プログラミング言語における字句解析，情報科学技術フォーラム講演論文集 8(1), 61-64, 2009-08-20