キーワードプログラミングの改良と実装

坂本悠輔†　佐藤晴彦†　小山聡†　栗原正仁†

†北海道大学大学院情報科学研究科

〒060-0814 札幌市北区北14条西9丁目

e-mail:†{sayuu, haru, oyama, kurihara}@complex.eng.hokudai.ac.jp

1. はじめに

キーワードプログラミングとは、ユーザーが与える少数のキーワードから正しいコードを生成することにより、ユーザーがプログラミング言語の文法規則やAPI 群の細かい部分を覚える必要を軽減する技術である。この技術においてユーザーが入力するキーワードの集合は、現在の文脈に適した表現を検索するクエリとして捉えることができる。著者らは現在、現状のキーワードプログラミングに対して、よりユーザビリティを高めるためのアイディアを複数考え、同時並行的にツールの実装も行っている。

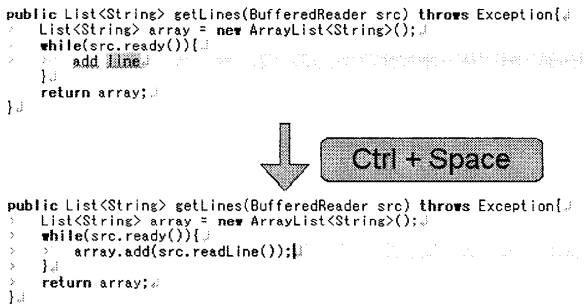

図 1 キーワードプログラミングでは、ユーザーが幾つかのキーワードを入力し、入力補完コマンドを押す事で適切なコードが挿入される。

2. キーワードプログラミング

キーワードプログラミングを提唱する既存研究[1]では、キーワードプログラミングを汎用的なプログラミング言語であるJavaに対して適用を試み、キーワード集合をメソッド呼び出し表現に対して当てはめるためのアルゴリズムを開発している。

2.1 関連研究との相違点

既存研究[2,3]においては、入力クエリとして「型」を入力する必要があったが、本技術では入力は「型」に制約されないので、自由な記述が可能となった。また、ユーザーに提供するコード片は、既成のライブラリや、フレームワークなどのソースコード集合が必要な既存研究とは異なり、プログラミング言語に備え付けのクラスライブラリのみでコードを自動生成することができるため、開発が始まったばかりの小さなプロジェクトに対しても使用可能という特徴がある。

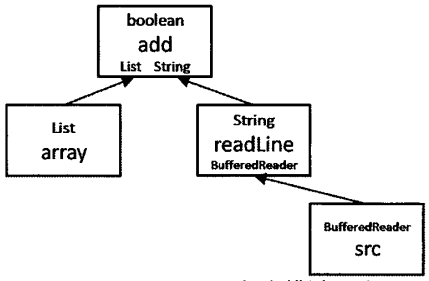

図 2 array.add(src.readLine()) を木構造で表現した図。ノードは1行目が返り値の型、2行目が関数名、3行目がレシーバと引数の型である。

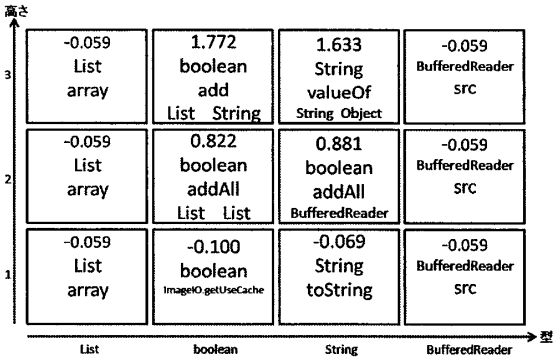

図 3 アルゴリズムによって作られる木の高さと型を軸とした関数の表。各ノードの1行目は評価値である。この表では表の1交点に対して、評価値の1番高いノードのみを表示しているが、実際はノードの評価値の高い順に数個保持する。

2.2 モデル

Java言語の基本データ型（int など）やクラス（java.lang.Object など）を型（type）と定義し、その集合をT とする。currentTimeMillis などのメソッド名や変数名をラベル（label） と定義し、その集合をL とする（複数の単語がつながっている場合は（current, time, millis）のように分割する。）。メソッドやフィールド、ローカル変数を関数（function）と定義し、（T × L × T ×…× T）というタプルで表現する。先頭のTは関数の返り値の型であり、Lが関数名、それ以降のTは引数の型である。例えば、関数String toString(int i, int radix) は、

Improvement and Implementation of Keyword Programming
†Graduate School of Information Science and Technology, Hokkaido University
Kita14 Nishi9, Kita, Sapporo, 060-0814, JAPAN
†{sayuu, haru, oyama, kurihara}@complex.eng.hokudai.ac.jp

(java.lang.String,(to, string), int, int) と表現される。キーワードクエリから生成されるコードは幾つかの関数の組み合わせとなる。それは1つの関数が1つのノードであり、それらが型で結ばれた木（図 2）として表現される。

2.3 アルゴリズム

以下に述べるようなステップで入力クエリから望ましい出力を算出している。

1. 動的計画法を用いている。はじめに全ての型 T の高さ n の木を作り、次にその評価値を利用して全ての型 T の高さ n+1 の木を作る、これを繰り返し、型と木の高さからなる表（図 3） を作成する。
2. 作成した表から現在の文脈に望ましい返り値の型の木を抽出し、それを出力とする。

関数に対する評価値はキーワードと一致するラベルを持つ関数が特に高い値を持つようなルールで与えられる。

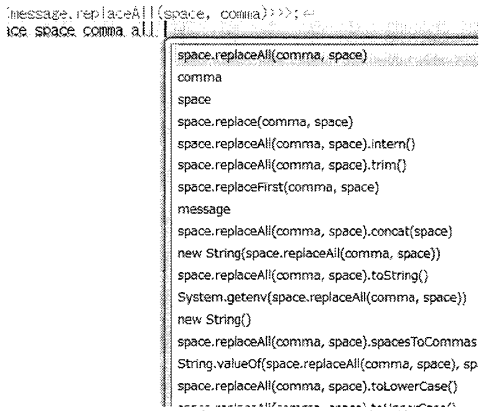

図 4 キーワードを入力し、ツールを起動したところ。

このツールは統合開発環境 Eclipse のコンテント・アシスト機能の部分を拡張したプラグインとして実装を行った。

3. 改良と実装

既存研究では、キーワードから生成される出力がただ1つ（アルゴリズム非公開のものは最大3つ）である。必ずユーザーの欲しい候補が生成されるとは限らないキーワードプログラミングにおいて、提示される候補数がこの既存研究のように少ない場合、欲しい候補がその中に含まれない可能性が高いと考えられる。したがって、本研究では提示候補を出来る限り増やし、沢山の候補の中からユーザーにそのコンテキストに適している候補を選択してもらうという方針を取ることにし、アルゴリズムの改良を行った。また、既存研究とは異なり、このように沢山の候補がある場合（図 4）、欲しいものを探すのには労力が必要となる。本研究ではその労力の軽減を目的として、ユーザーが頻繁に使用する候補についてはなるべく上位にランクインするように使用頻度を評価値の評価ルールの1つに加えた。

3.1 複数候補を提示するアルゴリズム

既存研究で公開されているアルゴリズムは1つの候補しか出力できないものであったので、それを改良し複数候補の提示を可能にした。既存研究のアルゴリズムの動的計画法の部分（2.3 のステップ 1）において作成した表（図 3）について、その1つの要素は1つの木（図 2）のルートノードとなっている。このルートノードは自分の子ノードの情報を持っていないために、表を作成した後、表から木を抽出しなくてはならない（2.3 のステップ 2）。本研究では、予め要素のルートノードに自分の子ノードの情報を持たせておくことにより、表を作成した後に木を抽出するという部分を省くことができた。また、表の交点にある既存研究では数個のみ保持していた評価値の高いルートノードを本研究では数百個保持させることにした。これによって出力時には、その保持しているルートノードの個数分だけ複数候補を出力できるようになった。

3.2 ユーザーの候補使用頻度の考慮

ユーザーが提示された候補を選択するたびに、その候補が何回選ばれたかという回数を保持しておく。具体的には「キーワード」と文脈から判断される「期待される返り値の型」、「生成コード」、「候補使用回数」を保存しておき、同じキーワード、同じ期待される返り値の時に同じ生成文字列が選択された場合、回数を1つ増やすということにした。生成コードだけでなく「キーワード」と「期待される返り値の型」までを考慮する理由は、同じユーザーであれば同じコンテキストでは大抵同じキーワードを使用して同じ結果を得たいと考えるであろうという推測に基づいている。

4. 今後の課題

今後の課題の1つ目は、既存研究の関数への評価ルールは、例えば『キーワードに含まれているラベルがあれば+1 点』、『関数がローカル変数であれば（コンテキストに近い方が望ましいため）+0.001 点』などというように固定的である。適当に定められているこれらの点数の値を、機械学習を用いることにより更新し適切な値になるようにしたい。2つ目としては、類義語への対応である。現在はキーワードに類義語が入っていても対応できない。例えば add というラベルを持つ関数はユーザーがキーワードとして append を入れたとしても評価値が高くならず候補として上位に来ることはない。これを解決するために、キーワードプログラミング専用の類語辞典を既存のオープンソース群などから作成することを考えている。

参考文献

[1] G. Little, R. C. Miller. Keyword programming in Java. Automated Software Engineering, 16(1), pp.37-71, 2009.
[2] N. Sahavechaphan, K. Claypool. XSnippet: Mining For Sample Code. Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented Programming Systems, Languages, and Applications (OOPSLA 2006), pp. 413-430.
[3] D. Mandelin, L. Xu, R. Bodik, D. Kimelman. Jungloid Mining: Helping to Navigate the API Jungle. Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation, pp. 48-61.