様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　２ 年　 ６ 月 １２ 日現在

機関番号：３２６１９
研究種目：基盤研究(C)（一般）
研究期間：2016～2019
課題番号：１６Ｋ００１０６

研究課題名（和文）文脈を考慮したコード補完の実装の系統的導出に関する研究

研究課題名（英文）A systematic approach to implementing context-sensitive code completion

研究代表者
篠埜　功（Sasano, Isao）

芝浦工業大学・工学部・准教授

研究者番号：１０３６２０２１

交付決定額（研究期間全体）：（直接経費）　　1,600,000円

研究成果の概要（和文）：本研究課題では、テキストエディタを用いたプログラミングにおけるコード補完を実装する方式を提案し、試験的実装を行った。この方式は、文法がBNFで記述される言語を対象としている。コード補完の実装者は言語の文法をBNFで記述し、それを、プログラムの接頭辞（カーソル位置までのプログラム）に対する文法へ変換する。変換後の文法によりカーソル位置までのプログラムを構文解析し、（部分的な）構文木を得る。この構文木を辿ることによりカーソル位置以降の構文の候補を計算し、プログラマに提示する。Standard MLのサブセットを例とした試験的実装をweb page上で公開した。

研究成果の学術的意義や社会的意義
広く使われているEclipse、Visual Studio等の開発環境において構文補完機能が提供されているが、形式的仕様が定められていないか、あるいは公開されていない。熟練プログラマは正確に把握できない機能の使用を避ける場合がある。また学術的には1980年代にSynthesizer GeneratorやMENTOR等、構造エディタの研究が多く行われた。近年では、構造エディタでの編集の正しさを論じた研究などもある。しかし構造エディタはプログラマが自由にプログラムテキストを編集するのを妨げる。本研究は、自由なプログラム編集を妨げることなく構文補完を行うための基本方式を提案したものである。

研究成果の概要（英文）：In this research project, we proposed an approach to implement code completion for programming on text editors. This approach targets at context-free languages. Those who implement code completion describe a grammar in BNF and transform it to a grammar for prefixes of the programs, i.e., programs up to the cursor, of the language. Based on the transformed grammar they parse programs up to the cursor and obtain (partial) parse trees. By traversing the (partial) parse trees they compute candidates to be completed at the cursor and show them to programmers. We have implemented a prototype system for a subset of Standard ML as an example and made public the system on a web page.

研究分野：プログラミング言語

キーワード：構文補完　LR構文解析　derivative　コード補完　文脈自由文法　文法変換

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

ソフトウェアの開発効率の改善は以前から大きな課題であり、プログラミングの効率を向上させるために、自動インデント、識別子補完などの入力補完や、識別子付け替え等のリファクタリングなどの支援機能が開発され、広く利用されている。これら支援機能の研究は、言語処理系に対して費やされる研究と比較すると非常に少ない。また現在広く用いられている Eclipse、Visual Studio 等の開発環境において識別子付け替えや識別子補完等の支援機能が提供されているが、形式的仕様が定められていないか、あるいは公開されていない。熟練プログラマは正確に把握できない機能の使用を避ける場合がある。また学術的には 1980 年台に Synthesizer Generator [1] や MENTOR [2]等、構造エディタ（structured editor あるいは projectional editor）の研究が多く行われた。構造エディタを用いることによりプログラム編集機能や構文補完機能の仕様が定めやすく、また実装が容易になる。近年では、構造エディタでの編集の正しさを論じた研究などもある。しかしながらテキストベースのエディタと異なり、構造エディタはプログラマが自由にプログラムテキストを編集するのを妨げる。本研究においては、プログラマが自由にテキストを編集するのを妨げないということをテキストベースと呼ぶ。この不便さにより、多くのプログラマは（プログラムのソースコード以外の構造データを編集するような場合を除き）構造エディタは用いないのが現状である。本研究は、プログラミングの効率改善に向け、テキストベースのコード補完機能の実現方式を提示し、実際の言語に適用してその有効性を示そうとするものである。構造エディタ以外の学術的なコード補完の研究では、穴（placeholder）や型を明示した上で補完候補を計算するものがある。それに対し、本研究は、プログラムテキストとカーソル位置のみから、補完候補を計算するものである。

２．研究の目的

本研究では、さまざまな支援機能の中で特にコード補完に着目し、テキストベースでかつ、構文に関する文脈に応じた的確な補完候補を仕様を明確にした上で計算できる方式の考案およびコード補完機能の仕様からの系統的実装を目的とする。コード補完はプログラミングにおいて基本的で有用な機能であり、Eclipse や Microsoft の開発環境などにおいて広く用いられており、コード補完が有効に用いられることはプログラミングの効率に大きな影響を及ぼす。また、コード補完の仕様を明確にした上でどのように補完候補を計算するかは自明ではない。またそれに基づいて Standard ML 等の言語に対してコード補完機能を提供する。これにより、プログラマがテキスト編集の邪魔をされることなく、かつ確信を持ってコード補完機能を用いることができるようになり、プログラミングの効率が向上する。

３．研究の方法

構文に関する文脈を考慮したコード補完の実装の系統的導出技法に関し、理論、実装の両面から研究を推進する。

[研究項目 A(理論)]: コード補完の基本方式の確立

申請者らの既存研究で提案した、型を考慮した識別子補完方式およびどのような構文誤りを許すかを明確に提示することのできる識別子補完方式を発展させ、キーワード、式、構文等の補完方式を開発する。

[研究項目 B(実装)]: 実装の系統的導出技法の考案

上記方式に基づいた実装を系統的に導出する技法を考案し、Emacs 等のエディタにおいてコード補完機能を提供する。また、実装においては対象言語のコンパイラ中のコードの再

利用を検討する。

４．研究成果

まず、コード補完の単純な場合として、構文に関する文脈を考慮したキーワードの補完を yacc の誤り回復機能に基づいて行うことを試みた。入力中の構文が不完全なソースコードに対応するため、Yacc による LR 構文解析の誤り回復機能を用いることとし、キーワード補完機能の実装を仕様から系統的に導出する手法を考案し実装した。この成果は情報処理学会プログラミング研究発表会において発表した[3]。本手法においては、カーソル位置の（入力途中の）キーワードの字句を識別するため、入力途中であることを示す字句を字句解析および構文解析の仕様記述に追加する。構文解析を行うので、補完候補計算において構文に関する文脈が考慮される。実装の導出は Byacc という構文解析器生成系のソースコードを用いて行った。Yacc の構文規則記述およびユーザが指定した補完対象キーワードを入力とし、構文規則記述への字句 error の機械的挿入により補完候補計算プログラムの自動生成を行うツールを作成した。例として、C 言語の Yacc の仕様記述ファイルを入力とし、C 言語のキーワードの補完を行うプログラムを生成した。補完対象キーワードは自動生成ツールの使用者が指定できる。

また、コード補完等の様々な機能を持つプログラム開発システムにおけるソースコードのリファクタリング機能に関し、関数型言語 Standard ML を対象に関数適用がギャップとなっているクローンを検出する新たな手法を提案し実装した。この成果は PEPM 2017 という国際会議において発表した[4]。

また、文法変換に基づく構文補完候補計算方式を考案し、試験的実装を行った。研究成果は ACM PEPM 2018、ACM PEPM 2020 という国際会議において発表した[5][6]。試験的実装は Standard ML のサブセットに対して行った。これは Emacs というテキストエディタ用の Emacs モードであり、web page 上で公開した[7]。この構文補完候補計算方式は、対象言語の BNF による構文規則を入力とし、途中までのプログラムの構文を出力する、文法変換による方式である。ただし、試験的実装では文法変換は実装しておらず、手動で行っている。この方式では、変換された文法を対象としてカーソル位置まで構文解析を行い、得られた（部分的なプログラムに対する）構文木を辿ることによってカーソル位置以降の構文の候補を計算し、ユーザに提示する。また、PEPM 2020 において、（途中までではない）対象言語の LR 構文解析器を用いた構文補完も可能であるという予想を提示した。PEPM 2020 の発表時の質疑応答において、カナダの McMaster 大学の Jacques Carette から、文法の derivative が関係するのではないかという意見を得た。文法の derivative について、正規言語や文脈自由言語を記述する式を対象として、derivative の言語を記述する式を得る研究がある[8]。また、上記の予想に関し、Chonnam National University の Kwanghoon Choi 氏と議論し、彼が最近開発した構文解析器生成ツール[9]を使って、LR 構文解析器の内部状態を用いることにより補完候補の計算ができるのではないかという見通しを得た。これらは今後の課題である。

参考文献

[1] Thomas Reps and Tim Teitelbaum. The Synthesizer Generator. *ACM Software Engineering Symposium on Practical Software Development Environments*, 1984.

[2] Veronique Donzeau-Gouge, Gerard Huet, Gilles Kahn, and Bernard Lang. Programming environments based on structured editors: the MENTOR experience. Technical Report RR-0026, INRIA, 1980.

[3] **白楊, 篠埜 功**, LR **構文解析のエラー回復機能を用いたキーワード補完機能の系統的導出, 情報処理学会 第** 109 **回プログラミング研究発表会, 浜松市福祉交流センター**, 2016 **年** 6 **月** 9 日～10 **日**.

[4] Tsubasa Matsushita, Isao Sasano, Detecting code clones with gaps by function applications, *ACM SIGPLAN 2017 Workshop on Partial Evaluation and Program Manipulation (PEPM 2017)*, Paris, France, January 16-17, 2017.

[5] Isao Sasano, An approach to generating text-based IDEs with syntax completion, *ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM 2018)*, short presentation, poster, demo, Los Angeles, California, United States, January 8-9, 2018.

[6] Isao Sasano. An approach to generate text-based IDEs for syntax completion based on syntax specification. *ACM SIGPLAN 2020 Workshop on Partial Evaluation and Program Manipulation (PEPM 2020)*, New Orleans, Louisiana, United States, January 20, 2020.

[7] `http://www.cs.ise.shibaura-it.ac.jp/pepm2020/`

[8] Matthew Might, David Darais, and Daniel Spiewak. Parsing with derivatives: a functional pearl. *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP2011)*, pp. 189-195, 2011 .

[9] Jintaeck Lim, Gayoung Kim, Seunghyun Shin, Kwanghoon Choi, and Iksoon Kim. Parser generators sharing LR automaton generators and accepting general purpose programming language-based specifications. *Journal of KIISE*, Vol. 47, No. 1, pp. 52-60, 2020.

５．主な発表論文等

〔雑誌論文〕　計0件

〔学会発表〕　計4件（うち招待講演　0件／うち国際学会　3件）

１．発表者名
Isao Sasano

２．発表標題
An approach to generate text-based IDEs for syntax completion based on syntax specification

３．学会等名
ACM SIGPLAN 2020 Workshop on Partial Evaluation and Program Manipulation (PEPM 2020)（国際学会）

４．発表年
2020年

１．発表者名
Isao Sasano

２．発表標題
An approach to generating text-based IDEs with syntax completion

３．学会等名
ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM 2018), short presentation, poster, demo（国際学会）

４．発表年
2018年

１．発表者名
白楊, 篠埜 功

２．発表標題
LR構文解析のエラー回復機能を用いたキーワード補完機能の系統的導出

３．学会等名
情報処理学会 第109回プログラミング研究発表会

４．発表年
2016年

１．発表者名
Tsubasa Matsushita, Isao Sasano

２．発表標題
Detecting code clones with gaps by function applications

３．学会等名
ACM SIGPLAN 2017 Workshop on Partial Evaluation and Program Manipulation (PEPM 2017)（国際学会）

４．発表年
2017年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考