

2023 年度 芝浦工業大学 工学部 情報工学科

卒 業 論 文

Standard ML を対象とした不完全な組パターンに対する
補完手法の提案および実装

学籍番号 **AL20047**

氏 名 庄司 歩夢

指導教員 篠埜 功

提出日 2024 年 2 月

目次

第1章	はじめに	1
第2章	関連研究	3
2.1	コード補完の有用性	3
2.2	書き忘れたコードに対するコード補完	4
2.3	構文誤りを含むソースコードに対するコード補完	4
第3章	提案手法	5
第4章	実装	6
4.1	VSCo de 上での補完候補表示	6
第5章	字句解析	7
第6章	構文解析	8
第7章	補完候補の取得	9
第8章	考察	10
第9章	まとめと今後の課題	11
	謝辞	12
	参考文献	13

第1章 はじめに

コード補完機能を利用することで、プログラム開発を効率的に行うことができる。コード補完とは、統合開発環境 (IDE) などのコードエディタから利用可能な支援機能で、入力途中のコードに対してその続きを入力する候補を自動表示する機能である。この機能を利用することによって開発者はコードをすべて直接記述する必要がなくなるため、開発の効率を向上させることができる。コード補完機能は Eclipse, IntelliJ IDEA, Visual Studio Code 等の統合開発環境では標準的に搭載されており、多くの開発者がコード補完を利用しているとの調査結果がある [1]。また、補完機能と関連して二つの関連研究を挙げる。三浦 [2] は Processing 言語において初学者がどのようにコード補完を利用するのか調査した結果、補完機能を利用することで括弧の入力数が有意に下がり、補完機能の有用性を示した。石原らは、既存の手法では書きかけのコード中に書き忘れがあった場合にそれらを補完することができないため、書き忘れたコードも補完対象とすることでコード補完の機会がどの程度向上するかについての調査を行った。その結果書き忘れが起こっていた場合でも適切な補完が行われると示した。したがってコード補完はコード開発をするうえで必要である。

しかし、すべてのプログラミング言語に対して補完機能が十分に整っていないのが現状である。実際に、関数型言語の Standard ML [3] はパターンマッチングによって複雑な条件分岐の記述を簡潔に書くことができたり、コンパイル時に型が検査され安全性が高いなどの特徴がありながらも、補完機能が充実していない。それに対して佐藤は [4]、統合開発環境である Visual Studio Code [5] 上で Standard ML のサブセット言語を対象にして関数宣言における括弧の記述忘れと引数の記述忘

れに対して変数名の補完が行われた。しかし、この研究は括弧の記述忘れと引数の記述忘れという書き間違いが発生していた場合に対して変数名を補完するものであり、括弧を補完したり引数を補完するものではない。したがって本研究では統合開発環境である Visual Studio Code で Standard ML を対象とした不完全な組パターンに対する補完手法を提案する。

本論文の構成は次の通りである。まず 2 章において関連研究について述べる。3 章において補完機能の実装概要について述べる。4 章において字句解析手法について述べる。5 章において構文解析手法について述べる。6 章において補完候補の取得について述べる。7 章において実装方法について述べる。8 章において考察を述べる。9 章においてまとめと今後の課題を述べる。

第2章 関連研究

関連研究として、コード補完の有用性、書き忘れたコードに対するコード補完、構文誤りを含むソースコードに対するコード補完について本研究との関連性について調査する。

2.1 コード補完の有用性

三浦 [2] は、効率的なプログラミングのために一般的に導入されている補完機能に着目し、初学者がどのように補完機能を利用するか Processing 言語を対象に調査した。その結果、自動補完を活用する学習者とそうでない学習者に二分された。利用頻度の高い補完候補には、for 文や if 文、画面描画関連の関数呼び出し、および void であった。for 文や if 文のスニペットには、括弧やカーリーブラケットを含んでいたため、補完機能を利用した編集行為の前後の文と、補完機能を利用しなかった編集行為の前ばの文と比較すると、括弧やカーリーブラケットの入力数が下がった。そのため、無限ループを生じる記述ミスの回数を有意に減少させることが確認された。しかし、この研究では、補完候補を表示するために Tab を入力しなければならないため、自動補完を活用する学習者が二分化されてしまった。したがって、本研究では補完候補が表示されるべきところで常に表示させるようにすることで、補完が使われないことを防ぐ。

2.2 書き忘れたコードに対するコード補完

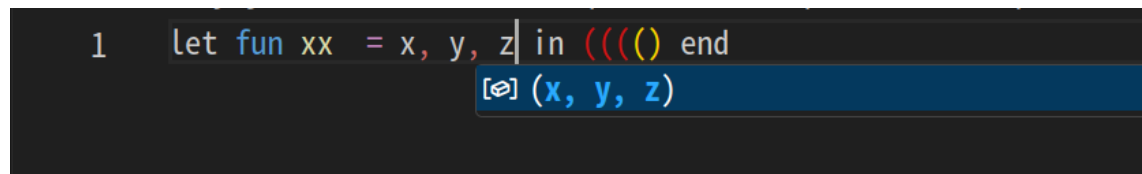
石原らは、既存の手法では書きかけのコードの途中で書き忘れがあった場合にそれらを補完することができないため、書き忘れたコードも補完対象とすることでコード補完の機会がどの程度向上するかについて調査した [6]. この調査実験は Java で実施され、被検体がタスクを実行していく中で書き忘れがどの程度発生していて、書き忘れに対する補完がどの程度行われたかを調査するものであった. その結果、書き忘れは全タスクのうち約 80 % で発生しており、その中で最も多い書き忘れはメソッドの引数に関するものであった. また、それらの書き忘れが起こっていたタスクうち 24 % のタスクにおいて書き忘れたコードに対する適切な補完候補が存在していることを示した. よって本研究で実装しようとしている不完全な組パターンに対する補完も有益であると期待される.

2.3 構文誤りを含むソースコードに対するコード補完

先行研究 [4] では、Standard ML のサブセット言語において error トークンを挿入することで構文誤りが存在しても補完が行われる実装が行われた. 実装は統合開発環境である VScode で Standard ML のサブセットを対象に、関数宣言による括弧の記述忘れと引数の記述忘れに対して変数名の補完が行われた. 本研究では、このサブセット言語を変更して不完全な組パターンに対して補完を行う.

第3章 提案手法

不完全な組パラメータを補完する方法を提案する．Standard ML のサブセット言語に対して補完を行う．補完は，字句解析，構文解析，補完候補の取得，補完候補の表示の順に行っていく．以下に実際に関数の引数が複数ある際に，括弧の記述を忘れたの補完の例を示す 3.1.



```
1 let fun xx = x, y, z | in (((() end
    (x, y, z)
```

図 3.1: 括弧の補完例

```
{todo:
今回の研究で使うサブセット言語をここに記述
括弧の補完候補のスクリーンショット
}
```

第4章 実装

4.1 VSCode 上での補完候補表示

{todo: 括弧の補完候補のスクリーンショット }

第5章 字句解析

字句解析器を生成するプログラムである Flex[7] というツールを用いて字句解析を行う。

第6章 構文解析

構文解析器を生成するパーサージェネレーターの1つである Bison[8] を用いて実装を行う。

第7章 補完候補の取得

Language Server で補完候補の計算を行う [9].

第8章 考察

第9章 まとめと今後の課題

謝辭

参考文献

- [1] Gail Murphy and Mik Kersten. How are java software developers using the eclipse ide? *IEEE Software*, Vol. 23, No. 4, pp. 76–83, 2006.
- [2] 三浦元喜. 初学者向け processing プログラミング環境におけるコード補完機能の導入と実践. 教育システム情報学会誌, Vol. 37, No. 2, pp. 167–172, 2020.
- [3] Robin Miler, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. 1997.
- [4] 佐藤直輝. Standard ML に対する構文誤りを許すコード補完機能の実装. 芝浦工業大学情報工学科卒業論文, 2022 年度.
- [5] Visual Studio Code - Code Editing. Redefined. <https://code.visualstudio.com/>.
- [6] 石原知也, 肥後芳樹, 楠本真二. 書き忘れたコードに対するコード補完について. 電子情報通信学会論文誌, No. 4, pp. 415–427, 2016.
- [7] Flex - the GNU project. <http://gnu.ist.utl.pt/software/flex/flex.html>.
- [8] GNU Bison - The Yacc-compatible Parser Generator. <https://www.gnu.org/software/bison/manual/>.
- [9] Language Server Protocol. <https://microsoft.github.io/language-server-protocol/>.