

実行履歴差分を用いたプログラム学習ツールの提案

内村隆聖 瀬川典久 杉野栄二 澤本潤

岩手県立大学ソフトウェア情報学部

1 はじめに

昔から、プログラム学習において他人の書いたプログラムを読むことは勉強になるとされている。近年のオープンソースコミュニティの発達により、SourceForge[1]などのWebサイトから実用レベルのプログラムを手に入れることができるようになった。そのため、プログラム学習者にとって、実用レベルのプログラムから高度なプログラミングを学習する機会が増えた。

だが実用レベルのプログラムは大規模で複雑な場合が多く、理解は容易ではない。さらにオープンソースコミュニティによって公開されたプログラムは十分なドキュメントが存在しないものも多く、プログラム理解を難しくする要因の1つであると考えられる。そこで、学習者のためにプログラム理解を支援するツールが必要であると考えられる。

そこで本稿ではプログラム理解をモデル化し、それに基づくプログラム学習ツールの開発を行う。

2 プログラム理解

本研究では、プログラム理解を仮定・確認・記録の3つのステージから構成されたと考える。まず仮定ステージでは、プログラム学習者が学習したい機能とそれが実装されているプログラムの対応を仮定する。次に確認ステージでは、仮定した内容が正しいかプログラムを実際に実行するか読むことで確認する。そして記録ステージでは、確認された内容つまり理解したことの記録を行う。

これら3つのステージを1つのサイクルとして繰り返すことで、プログラムの理解は深まっていくと考える(図1)。

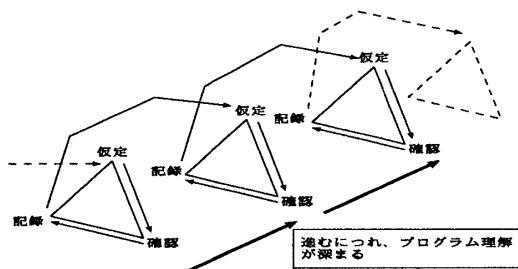


図 1: プログラム理解モデル

2.1 プログラム理解に用いられる情報

プログラム理解に利用される情報は、大きくわけて次に示す2つであると考えられる。

Proposal of program learning tool using execution slices
 Ryuusei Uchimura, Norihisa Segawa, Eiji Sugino,
 Jun Sawamoto
 Faculty of Software and Information Science, Iwate Prefectural
 University
 152-52, Sugo, Takizawa, Iwate, Japan

● 静的情報

プログラムの構造情報。プログラムを実行せずにシンボル間の依存関係を解析するプログラムスライス、プログラム依存グラフ (Program Dependence Graph : PDG) などから得られる情報が含まれる

● 動的情報

プログラムの振る舞いに関する情報。プログラムを実行し結果からプログラムの振る舞いを解析するデバグカ、プロファイラなどから得られる情報が含まれる

小野らによる研究では、2つの情報を相互に参照し試行錯誤しながらプログラム理解を進めることが良いと述べられている[2]。しかしながら、2つの情報を用いても実用レベルのプログラムの理解は依然として難しいままである。

2.2 プログラム理解を妨げる要因

理解を妨げる要因を考えたとき、以下のようなことが考えられる。

1. 十分なドキュメントをもたない
機能とプログラムの対応を仮定するのが難しくなる
2. 書き手と読み手のプログラム知識の違い
仮定した内容を確認する場合、プログラムが理解できないと確認できない
3. 規模が大きい
プログラム理解モデルにおけるサイクルは増えるごとに理解が深まっていくが、多すぎた場合にプログラムを読むことへの意欲が損なわれ続かない。さらに覚えなければならない情報が多くなり、理解が発散してしまう

2.3 プログラム理解支援

十分なドキュメントを持たないものは、静的情報を用いてプログラムの構造を知ることによって理解を支援できると考える。また、書き手と読み手のプログラム知識の違いは、動的情報を用いて調べたい部分の実際の振る舞いを知ることによって理解を支援できると考える。そして規模が大きいものに対して覚えなければならない情報は、記録することで支援できると考える。しかし、規模による多サイクルを減らすためには、静的情報および動的情報だけでは支援が不十分である。

ここで解決案として、必要な部分に範囲を絞ってプログラムを読むことが考えられる。Wongらはプログラム保守のためにプログラムの実行履歴という動的情報を利用し、その差分からプログラムの必要部分を抽出する研究を行った[3]。この研究によって実行履歴の差分から、十分にプログラムの必要部分が抽出できることがわかっている。

この方法を利用すれば大規模なプログラムについて、必要な部分に範囲を絞ってプログラムを読むことが可能である。また実行履歴という動的情報の差分で読む範囲を絞るだけでなく、そこに静的情報を加えることでプログラム理解をさらに助えることができると考えられる。

3 プログラム学習ツール

本研究において提案するプログラム学習ツールの構成を図2に示す。ツールを構成するモジュールは、2章で示した3つのステージを支援する。

仮定ステージは、実行履歴取得と差分抽出がプログラムの読む範囲を絞り支援する。確認ステージは、今回はプログラムの振る舞いに関する動的情報ではなく、実行履歴の差分情報と静的情報抽出から得られた静的情報を情報合成しインターフェースを介してユーザに提供することで支援する。記録ステージは、ユーザはインターフェース上から理解した内容を付箋のような形で直接記録できるようにすることで支援する。

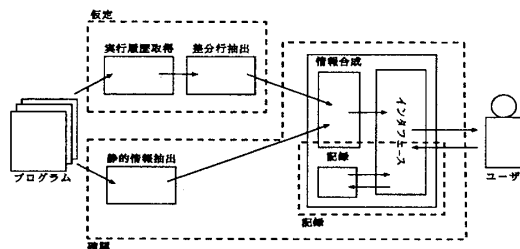


図 2: プログラム学習ツールの構成

各モジュールの実装は、以下のように実装した。

- 実行履歴取得モジュール
変更を加えた gdb を利用して、実行履歴の取得を行う。取得した実行履歴は、実行履歴ファイルとして保存される
- 差分抽出モジュール
perl を利用して実行履歴が保存された 2 つのファイルから、差分抽出を行い差分ファイルとして保存する
- 静的情報取得モジュール
今回は実装の容易さから、既存のツールである global[4] を利用する。global はシンボル間の依存関係を解析し、その関係をリンクとした HTML 化されたソースコードを出力する
- 情報合成モジュール
perl を利用して実行履歴の差分情報と、静的情報取得モジュールで得られた情報を合成する
- 記録モジュール
javascript と perl を利用して、HTML 化されたソースコード上に付箋のような形で記録が行える

インターフェースは Web ベースで実装し、各ウィンドウは以下に示す内容をユーザに提供する (図 3)。

- プログラムコード (右下)
実際のプログラムコードを表示する
- プログラム実行履歴の差分 (右上)
プログラム実行履歴の差分結果を表示する。ユーザはここから読む範囲を絞り、プログラム理解を進めることができる
- シンボル情報 (左上)
プログラム内で利用されているシンボルをアルファベット順に整理し表示できる。ユーザは、任意のシンボルに対してシンボルの宣言位置もしくは利用されている場所を見つけることができる

● プログラムのファイル構成 (左下)

プログラムのファイル構成を表示する。ユーザはリンクをたどるように、ファイルの構成を見ることができる

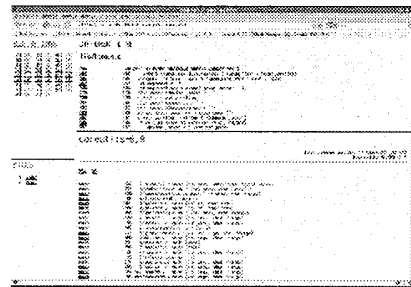


図 3: プログラム学習ツールのインターフェース

4 評価

2章に示したプログラム理解の仮定・確認・記録の各ステージに対して、以下のような評価を現在進めている。

- 仮定
実行履歴の差分によって、どの程度読む範囲を絞ることができるかについて調査する
- 確認
プログラム理解を進めるなかで行う作業時間について、提案ツールと grep 付きエディタを比較する
- 記録
記録が必要な項目、記録したことを利用する際に必要な機能等に対して、アンケート調査を行う

5 まとめ

本研究では、実行履歴から得られる動的情報の差分と静的情報を相互に参照できる環境を提供するツールの作成を行った。また、そのツールによってプログラム理解が支援できることを、3つのステージに対する評価によって示す。今回提案したツールによって、プログラムを読むことによる学習が容易になることが期待される。インターフェースが HTML であること活かせば、拡張により Web 上における複数人でのプログラム理解もサポートできると考える。今後として提案した学習ツールの確認ステージにおいて、振る舞いに関する動的情報も取り入れたツールへの改善が考えられる。

参考文献

- [1] SourceForge.net, <http://sourceforge.net/>, 2007/06/13.
- [2] 小野展弘, 本間昭次, 深谷哲司. “プログラムの静的解析と動的解析の相互補完によるプログラム理解・開発支援プロセス”. 電子情報通信学会技術研究報告 vol.100, No.441, 知能ソフトウェア工学, pp.9-16 (2000).
- [3] W. Eric Wong, Swapna S. Gokhale, Joseph R. Horgan. “Locating Program Features by using Execution Slices”, Proceeding of the 1999 IEEE Symposium on application - Specific System and Software Engineering and Technology, p.194-203 (1999).
- [4] GNU GLOBAL source code tag system, <http://www.gnu.org/software/global/global.html>, 2007/11/05.