

Cプログラムの理解を支援するナビゲーション機能

太田 洋介 大久保 弘崇 粕谷 英人 山本 晋一郎

愛知県立大学大学院 情報科学研究科

アブストラクト Cプログラムの理解を支援するため、コールグラフ視覚化ツールとクロスリファレンサをもつプログラムブラウザとを連携させるナビゲーション機能を実装した。連携とはこれらの二つのツールが常に同じ関数を提示することであり、利用者は情報の差異を気にすることなく二つツールを統合して扱える。また、関数の参照関係を有向グラフと捉えた三つの機能 (1) プログラムを辿る上で方向感覚を与えるためにコールグラフに閲覧経路を重ね合わせ提示する機能、(2) 閲覧履歴を整理するためのコールグラフヒストリー、(3) 閲覧した順序を示すスネーク、を持つコールグラフ視覚化ツールを実装した。閲覧履歴機能によりプログラムに存在する共有部品の抽出を行うなどのプログラムの静的解析における支援を提案した。

The Navigation Facility to Support Understanding C Programs

Yosuke OHTA, Hirotaka OHKUBO, Hideto KASUYA, and Shinichiro YAMAMOTO

Graduate School of Information Science and Technology, Aichi Prefectural University

Abstract We mounted navigation facility that made a call-graph visualization tool and a program browser with cross referencing to collaborate each other. Collaboration means that these two tools always present the same C function, so we can unify and treat two tools without noticing difference of information. We also mounted a call-graph visualization tool has three operation history features that treats function calls as a directed graph, (1) presenting recorded operation history superimposed on with the call-graph, which gives you the sense of direction on tracing the program, (2) the call-graph history view you can organize recorded operation history, and (3) the Snake presents recorded operation order. We show that the operation history feature supports static analyzation of programs such as extracting shared program components.

1. はじめに

一般にソフトウェア開発を行う上で、プログラマは他人の書いたプログラムを理解することが必要不可欠である。本研究では理解支援として、クロスリファレンサとプログラム視覚化ツールを連携させることでソースプログラム上でのナビゲーション機能を提供する。また、閲覧履歴を視覚的に提示することで利用者にプログラムの静的解析支援を行う。

2. 理解支援ツールの現状

現在までに、プログラム理解支援の手法がいくつか提案されている。それらは支援を行う対象や目的に違いがあり、適用する状況もそれによって違いがある。本章では、プログラム理解支援ツールとその特徴について考察する。

2.1 プログラム理解支援ツール

プログラム理解支援ツールの代表的なものを以下に示す。

クロスリファレンサ

クロスリファレンサとは、ソースプログラムの構成要素をリンクさせ、その関係を明らかにするためのツールである。例えば、ソースプログラムの中から、指定した関数の定義部分と参照部分を抽出する機能を持つ。既存のツールには、SPIE [2] や GLOBAL [3]、cxref [4] などがある。

プログラム視覚化ツール

プログラムのソースやデータを視覚化するツールがある。ソースを視覚化するものは主に制御構造を対象とし、例えば、静的なものではPAD図やフローチャートで表現するもの、動的なものではソースの実行している箇所をハイライトするものがある。データを視覚化するものは主にデータ構造を対象とし、例えば、静的なものではポインタ接続された構造体のようなデータ構造を図形で表現するもの、動的なものではデータの変化を表現するアルゴリズムアニメーションがある。

2.2 コールグラフビューワ

C言語のコールグラフを三次元に視覚化するVCRGL [1] が提案されている (図1)。VCRGLは、ファイルを表す複数の円盤の円周上にそれぞれのファイルにある関数のノードが配置される。円の中心に注目する関数のノードが配置されていて、その関数から関数呼び出しのエッジが放射状に表示される。関数呼び出しを辿る操作により、現在注目している関数のノードが常に円の中心に表示される。インタラクティブなコールグラフビューワという点がVCRGLの大きな特徴である。

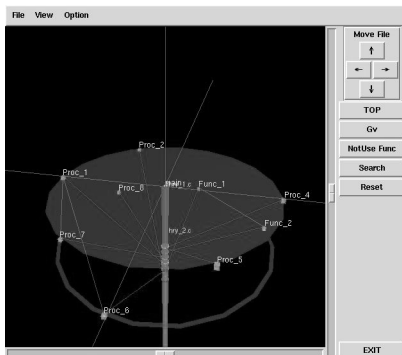


図1 VCRGLの動作画面

2.3 プログラムブラウザ

本研究では、SPIE [2] より提供されるプログラム閲覧機能をプログラムブラウザとした。SPIEは、ソースプログラムから、その中で定義されている関数、大域変数やマクロに関する各種の情報テーブルによって、利用者がソースプログラムを理解する作業を支援するツールである。出力されたXHTMLファイル群では、適切なハイパーリンクや文字のカラー表示などの修飾により、Webブラウザにおけるソースプログラムの高い読解性を実現している (図2)。また、専用のCGIスクリプトと共にWWWサーバ経由でXHTMLファイル群を参照することにより、文字列の検索ができる。

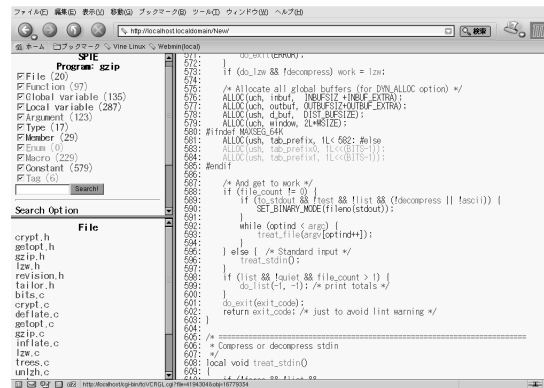


図2 プログラムブラウザの動作画面

2.4 現状における課題

現在、様々な手法によるプログラム理解支援ツールがあるが、それぞれの特徴を理解した上で利用者は理解する対象や目的に適したツールを選ばなくてはならない。詳細な情報を確認したい場合はクロスリファレンサなどのテキスト情報が必要となり、全体像や複雑な参照関係を理解する場合には図形表現による理解支援ツールを利用すると有効である。

物事を理解する際、大まかな情報から詳細な情報へ理解を進めるのが大きな流れであるのは確かである。しかし、プログラムのような複雑な構造機構を理解する場合は、多種多様の参照関係を理解することが必要となる。つまり、複数の理解支援ツールを利用するならば、ツールを切り替える機会が増す可能性がある。ツールの切り替えが増えれば、ツールを利用する理解支援効果よりも切

り替えの労力が上回り、いずれツールを利用しなくなることに繋がる。

以上のことを解決するために、多くの表現方法や視点で理解支援を統合して扱うツールが必要である。

3. 連 携

コールグラフビューとプログラムブラウザの提示する関数が常に同じになるように更新し合う連携機能を実装した。

3.1 Sapid

Sapid (Sophisticated APIs for CASE tool Development) [5] は、細粒度ソフトウェア・リポジトリに基づいた CASE ツール・プラットフォームである。Sapid はソフトウェアデータベース (SDB: Software Database)、アクセスルーチン (AR: Access Routines) などから構成されている。SDB は更に、I-model、PIDB、P-model と呼ばれるデータベースから構成される。I-model は C 言語の前処理後のソースプログラムを 12 種類のクラスと 29 種類の関連としてモデル化したデータベースである。PIDB は C 言語の前処理情報に関するデータベースである。P-model は I-model と PIDB を元に前処理指令を 11 種類のクラスと 17 種類の関連としてモデル化したデータベースである。Sapid を用いることにより、従来、CASE ツール開発者や研究者が個別に用意していた解析器やリポジトリを実現する必要がなくなり、CASE ツールの本質的な機能の開発に集中することが可能となる。

3.2 XHTML コンバータ

プログラムブラウザからコールグラフビューへの注目関数の更新を行うために CGI を用いる。プログラムブラウザにおける提示情報の更新は、関数呼び出しを示すアンカーによりジャンプをして定義箇所へページが移動したときに行う。移動先である関数をコールグラフビューに伝達するため、CGI に呼び出し先の関数のオブジェクト ID を送信する。そのために、SPIE で生成された XHTML の関数呼び出しのアンカータグに CGI 呼び出しを埋め込むためのコンバータを実装した (図 3)。SPIE で生成された XHTML のハイパーリンクのアンカーには Sapid で解析したプログラム情報のオブジェ

クト ID が埋め込まれており、コンバータはそのオブジェクト ID を CGI に送信するために関数呼び出しのリンクを CGI 呼び出しに書き換えるツールとして実装した。

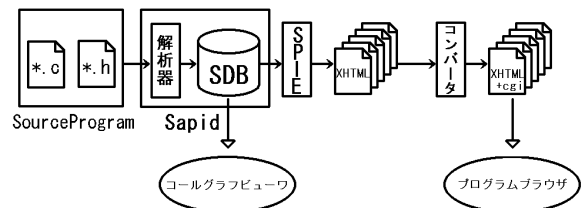


図 3 2つのツールのプログラム情報の取得

3.3 ツールの相互更新

プログラムブラウザからの更新とコールグラフビューからの更新は共に CGI を用いて行う。プログラムブラウザからの更新は、関数呼び出しでリンクにより CGI を呼び出し、CGI が TCP/IP によりコールグラフビューにオブジェクト ID を送信することで実現した (図 4)。

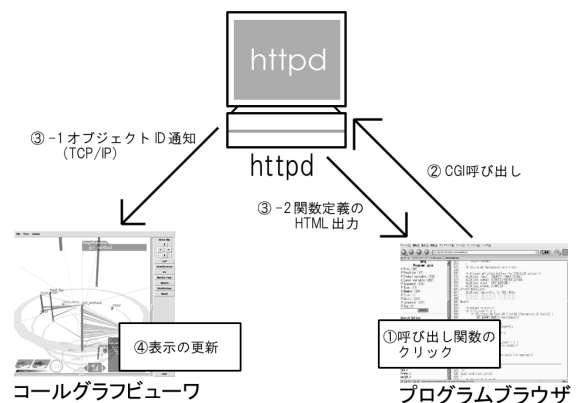


図 4 プログラムブラウザからコールグラフビューへの通知

コールグラフビューからの更新は関数を迎る操作を行うと、新たに注目関数になる関数のオブジェクト ID をシステムコールにより、ブラウザに CGI として読み込ませることで CGI から注目関数のアンカーを受信する (図 5)。

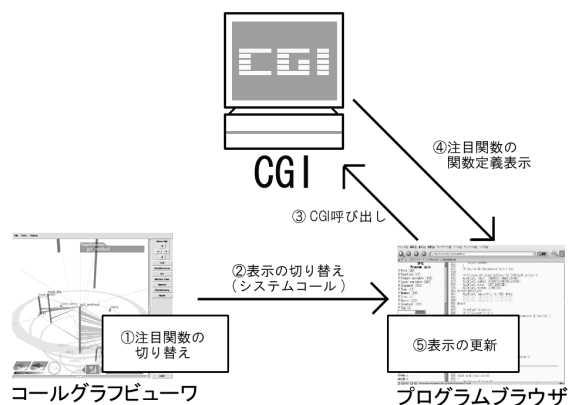


図 5 コールグラフビューからプログラムブラウザへの通知

以上の機構により、コールグラフビューでの注目関数とプログラムブラウザで表示している関数を、どちらのツールで関数を辿る操作をしても互いに更新し合うことが可能となった。

4. 閲覧履歴機能

4.1 閲覧経路の統合

本研究で閲覧経路の視覚化を行う目的は、プログラムの参照関係の繋がりを直観的な表現で提示し、ソースプログラムの理解支援を行うことである。さらに、コールグラフビューだけではステートメントの細かい内容まで情報の提示が行き届かない点をプログラムブラウザで確認し、また、プログラムブラウザ上での操作もコールグラフでのそれと同様に閲覧履歴として扱う。

テキストベースのソースプログラム理解支援ツールの SPIE と、コールグラフビューの操作履歴を統合して扱うことで、ソースプログラム上でのナビゲーション機能を実現する。Web ブラウザの簡単な履歴によるナビゲーションに比べ、プログラムブラウザであることに特化した閲覧履歴機能をコールグラフビューに組み込み、SPIE の有用性を向上させる。

また、閲覧経路を多様な手法で表現することで、利用者が複雑な辿り方をしてても容易に閲覧経路を把握できることを目指す。

4.2 履歴の記録

閲覧履歴に関わる機能を実行する前に、コールグラフビュー上での閲覧履歴を記録する必要がある。閲覧履歴機能を利用しない場合は、コール

グラフビューは関数呼び出しが参照・被参照に関わらず関数を自由に辿ることが出来る。しかし、閲覧履歴を記録する場合、利用者は関数の呼び出し経路による動作の変化やモジュールの主従関係などを調査するため、関数の参照関係が非常に重要であることから、閲覧履歴は関数呼び出しを有向グラフと捉えて記録する。そのために辿った関数呼び出しを有向エッジとして捉え、呼び出し方向へ辿った場合に有向エッジを記録し、呼び出し方向を逆へ辿った場合は return と捉え、有向エッジは記録しない。

閲覧履歴は関数の呼び出し関係の理解を目的とするため、履歴の記録時に呼び出し関係のない関数を注目関数にした場合には記録を中断する。

4.3 閲覧履歴機能のインタフェース

閲覧履歴機能を利用している際に、閲覧履歴機能特有の操作を支援するためのインタフェースを実装した (図 6)。

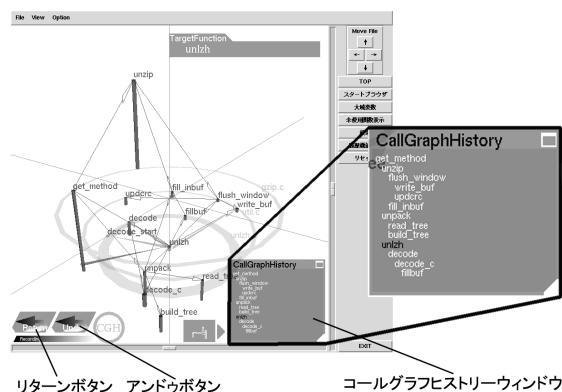


図 6 インタフェース

コールグラフヒストリーウィンドウ

コールグラフヒストリーウィンドウは、閲覧履歴を元にコールグラフを生成し、辿った関数の中で現在の注目関数の位置づけを確認することができる。

リターンボタン

履歴を記録している場合は、複数の関数から注目関数が参照されていても、注目関数の呼び出し元の関数は履歴から特定することができる。そこで、

閲覧履歴機能の利用時に、呼び出し元の関数を探す手間を省くことができるように、リターンボタンを実装した。リターンボタンにより、呼び出し元の関数へ容易に戻るることができる。

アンドゥボタン

履歴を記録する際に、誤って辿るつもりがなかった関数を辿ってしまった場合、前の関数に戻り閲覧履歴を戻っただけ削除する必要がある。そこで、前の関数へ一手分閲覧履歴を削除しつつ戻るためのアンドゥボタンを実装した。アンドゥにより履歴の記録を開始した関数まで戻ることができる。

4.4 閲覧経路

コールグラフビューワ上の閲覧経路は、関数の呼び出し方向を示すため矢印により表現した。また、一度閲覧された関数は表示が閲覧したことを示す色に変化する(図7)。コールグラフビューワのインタフェースは、現在の注目関数と呼び出し関係のある関数だけが表示されるが、その上に閲覧済みの関数を閲覧経路と併せて提示することは、利用者に閲覧した関数呼び出しを確認させ、ソースプログラム上における方向感覚を与える。そこで、閲覧した関数は注目関数との呼び出し関係に関わらず提示するようにした。

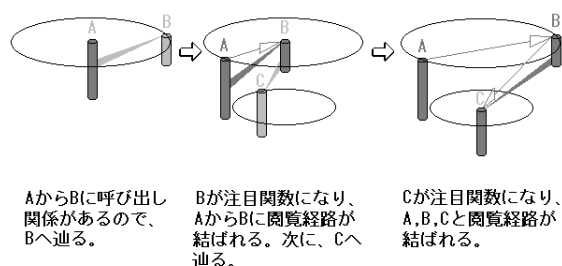


図7 閲覧経路の表示

4.5 コールグラフヒストリー

コールグラフヒストリーは、閲覧履歴を元にコールグラフをツリー形式で表現したものである(図8)。コールグラフヒストリーは、閲覧履歴を開始した関数をツリーのルートに配置し、ルートからの深さを示すリング上に閲覧した関数を閲覧履歴を

元に配置している。そして、辿った関数呼び出しを関数間にエッジを結ぶことで表現している。複数の呼び出しエッジが一つの関数を参照していることもあり、ツリーではなく閉路を含む有向グラフになっている。

コールグラフビューワではファイルリングにより関数の属するファイルが示されるが、コールグラフヒストリーではこれを色分けにより提示する。呼び出しエッジの色は呼び出し元の関数と同じになる。ファイル毎に色を変えることにより、興味のあるファイルの機能や利用のされ方を参照関係から理解することができる。

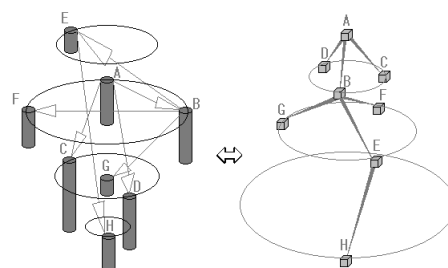


図8 コールグラフヒストリーによる閲覧経路の整理

4.6 スネーク

閲覧経路と閲覧履歴は関数呼び出しを有向グラフと捉えて情報を提示した。しかし、ソースプログラム上でのナビゲーションにおいて利用者は有向グラフとして捉えるだけでなく、単に閲覧した順序を確認する必要もある。そこで、閲覧した順序で関数をラインが辿っていくアニメーション機能を加えた、これをスネークと呼ぶ。

スネークは閲覧履歴を記録し始めた関数から順に関数を素早く辿っていく。一度、スネークが辿った関数はスネークの色に変わる。スネークは閲覧した関数を全て辿り終わると、繰り返し最初の関数から辿るようになっている。また、コールグラフもスネークの現在地に同期して関数名が変色し、スネークの現在地をコールグラフでも確認できる。

5. 利用例

本研究の利用例として gzip に対しツールを適用

[illegible]

図 9 コールグラフ履歴の機能毎の色分け

6. おわりに

本研究では、ソフトウェア開発におけるソースプログラムの再利用や保守に対する要求が高まっていることを受けて、ソースプログラムの理解を支援する方法の一つとして、クロスリファレンスのプログラムブラウザとプログラム視覚化ツールのコールグラフビューワの連携により、プログラム上でのナビゲーション機能を実装した。また、閲覧経路の提示により関数の呼び出し経路による動作の変化の理解支援として、利用者に静的解析を支援する機能を実装した。以下に今後の課題を挙げる。

ファイルの配置法

コールグラフビューワのファイルを示すリングは関数の多い順に中心になるように配置されている。しかし、ソフトウェアはモジュール毎に機能分けを行うことから、ファイル間に呼び出し関係に違い

がある。そのため、関数の数でリングの配置を決めるよりも、ファイル間の呼び出し関係で配置を決定した方が視認性が向上する。

視点の最適箇所への自動移動

コールグラフィビューワを利用する上で、利用者はオブジェクトが見え易い視点位置を探って操作しなければならない。インタフェースとしては、複雑な関数関係を比較的分かりやすい状態で提示するように設計されているが、視点位置によって見え易さに大きな違いがある。

そこで、注目関数が入れ替わるときなどに最もオブジェクトが見え易い視点位置を算出し自動で移動する機能により、視認性をより向上させることが期待できる。

謝 辭

御指導頂いた、愛知県立大学情報科学部 稲垣康善教授と名古屋大学大学院情報科学研究科 阿草清滋教授に感謝します。また、本研究は、日本自転車振興会の補助金を受け、(財)機械システム振興協会、(財)ソフトウェア工学研究財団が実施したプロジェクト『組み込みシステムの理解支援環境に関するフィージビリティスタディ』の一部として行なわれました。ここに感謝します。

参考文献

- [1] 安原継二, “オブジェクト属性を利用したソフトウェアの視覚化”, 日本ソフトウェア科学会 FOSE2000, pp.189–196, 2000
- [2] 大橋洋貴, “ハイパーテキストに基づいたソースプログラム・レビュー支援ツール”, 電子情報通信学会, ソフトウェアサイエンス研究会, 電子情報通信学会技術研究報告 SS98-28, pp.15–22, 1998
- [3] Shigio Yamaguchi, “GNU Global-Source Code Tag System for C, C++, Java and Yacc.”
<http://www.gnu.org/software/global/>
- [4] Andrew M. Bishop. The Cxref Homepage.
<http://www.gedanken.demon.co.uk/cxref/>
- [5] 福安直樹, 山本晋一郎, 阿草清滋, “細粒度リポジトリに基づいた CASE ツール・プラットフォーム Sapid”, 情報処理学会論文誌, Vol.39, No.6, 1998, pp.1990–1998