様式 C-19

科学研究費補助金研究成果報告書

平成 21 年 6 月 2 日現在

研究種目：基盤研究（C）
研究期間：２００５～２００８
課題番号：１７５００００１９
研究課題名（和文）バイナリレベル軽量データ統合方式による高精度な C 言語用 CASE ツールの構築

研究課題名（英文）Design and implementation of precise C language CASE tools based on binary-level lightweight data integration

研究代表者　権藤　克彦（GONDO KATSUHIKO）
東京工業大学・大学院情報理工学研究科・准教授
研究者番号：５０２６２２８３

研究成果の概要：
現代社会を支える基盤ソフトウェアの障害を少なくする新しい手法として，本研究ではバイナリレベル軽量データ統合という方式を提案した．デバッグ情報 DWARF2 などバイナリ中のデータを解析した結果を XML 形式で格納するツール群を実装･評価してよい結果を得た．また，この方式を発展させたり，欠点を補うために，識別子データマイニング，C 前処理系解析ツール，データ競合検出ツールなども開発し，良い結果を得た．

交付額

（金額単位：円）

	直接経費	間接経費	合　計
２００５年度	1,100,000	0	1,100,000
２００６年度	900,000	0	900,000
２００７年度	1,200,000	360,000	1,560,000
２００８年度	500,000	150,000	650,000
年度			
総　計	3,700,000	510,000	4,210,000

研究分野：ソフトウェア工学
科研費の分科・細目：情報学・ソフトウエア
キーワード：CASE ツール，XML，ANSI C，バイナリレベル，データ統合

１．研究開始当初の背景

(1) 現代社会では多くの社会基盤がソフトウェア技術によって支えられている．例えば，携帯電話，銀行 ATM システム，車載システムでは 100 万ステップ規模のソフトウェアが稼動している．残念ながら，これらの基盤ソフトウェアはしばしば障害を起こすことが知られている．百億円規模の損害を出したり，場合によっては人命を脅かすため，これらの障害を取り除くことは非常に重要である．

(2) 100 万ステップ規模の大規模ソフトウェアは人間が扱える範囲を大きく超えているため，開発ツールと呼ばれるソフトウェア自身によって，人間によるソフトウェア開発を支援させる方式をとってきた．その主要な方法である従来のソースレベルデータ統合では C 言語用の CASE ツールの精度が非常に悪い．次の論文は複数のコールグラフ生成系の出力を比較して，解析結果が驚くほど大きく食い違うことを報告している．（G. Murphy ら，An Empirical Study of

Static Call Graph Extractors, ICSE, pp. 90-99, 1996.）その原因は，(1) C 言語仕様の曖昧さ(例：未規定動作)，(2) C コンパイラ独自拡張機能(例：¥texttt{asm}命令)，(3) C 言語仕様の肥大化(例：C99)，(4) C プリプロセッサ，(5) C コンパイラとツールの非連携，など多岐にわたっているが，ソースコードを解析する限り，この問題の解決は非常に難しい．

2．研究の目的

(1) 1.で述べた問題を解決するため，新しい方式としてバイナリレベルデータ統合と軽量データ統合を用いて，高精度な CASE ツール（ソフトウェア開発ツール）を実現する手法の確立を目的とする..

(2) バイナリレベルデータ統合とは実行可能ファイル中のデータ（例：テキスト・再配置情報・記号情報・デバッグ情報）用に CASE ツールが利用しやすい共通フォーマットを定義し，ツールの開発・連携を容易にすることを指す．また，軽量データ統合とは複数の共通フォーマットの存在や，情報の欠落を許すデータ統合を指す．従来は完全な統一を目指す方式が主流だったが，実用レベルのツール構築には軽量データ統合が必須と考える．

3．研究の方法

(1) 本研究では C 言語用，ELF/DWARF2，理解ツールに限定したツール構築実験を行う．より具体的には，バイナリ形式を ELF/DWARF2 に限定した軽量データ統合を行い，C 言語のための基本的なソフトウェア理解ツール，例えばクロスリファレンサやコールグラフ生成系に限定した構築実験を行い，提案手法の有効性を実際に検証する．

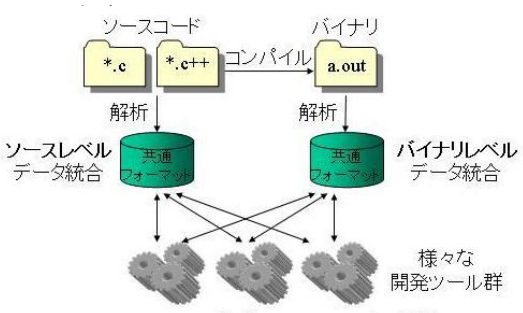

図1. ソースレベルとバイナリレベルデータ統合

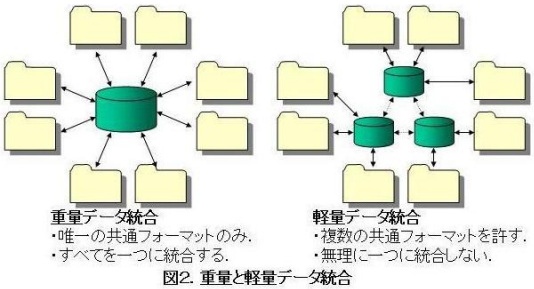

図2. 重量と軽量データ統合

次のツール開発および研究成果を挙げた．

(1) DWARF2 に基づくツール群

バイナリレベルの軽量データ統合方式（図 1）（図 2）を用いた C 言語用ツール群として，DWARF2-XML(図 3)，x_debug，dxref，rxref，bscg(図 4)，AXES などの実装実験した．また，我々の DWARF2-XML と従来手法の libdwarf API の比較によりデータ統合方式の優位性を示した．

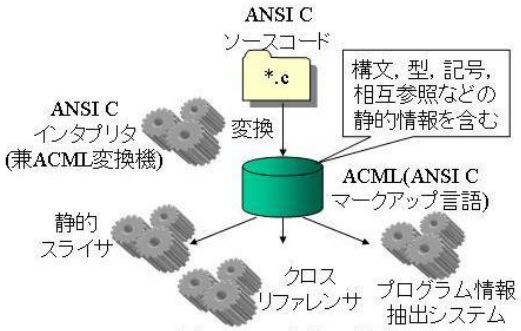

図3: ACMLとその応用

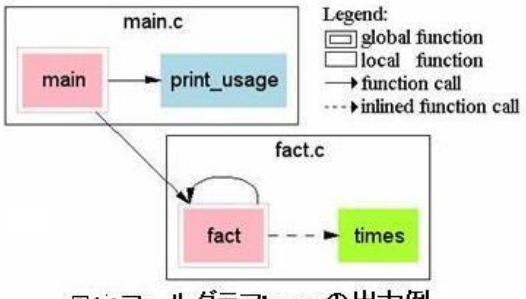

図4: コールグラフbscgの出力例

(2) 識別子データマイニング手法

データ統合方式を補完する識別子データマイニング手法を導入した．組み込み分野では割り込みを多用するため，クロスリファレンサなどの従

来の開発環境だけではプログラム理解が困難である．これを補うため，識別子から重要な概念キーワードを抽出するアルゴリズム ckTF/IDF を提案し，我々が構築した教育用 OS である udos に適用し，良い結果を得た．

(3) C++言語用コールグラフ生成系

C++言語はその複雑な言語仕様のため、C 言語以上にツール構築が困難であることが知られている。本研究では関数呼び出しの直前のバイナリコードを軽量解析することで、仮想関数にも対応した高精度なコールグラフ生成系の構築に成功した。

(4) コード・文書間の追跡性ツール

ソースコードと外部文書の関係は現実のソフトウェア開発ではほとんど管理されておらず、保守性悪化の大きな原因となっている。本研究では我々が開発した教育用 OS (udos)を例題に、コード・文書間の追跡性や追跡対象となるキーワードに関する研究を行った。従来法である TF/IDF 法をプログラム向けに改良した ckTF/IDF 法を提案した．また，プログラム上の実装レベル制約とソースコード間の追跡性保存に関する研究を行った．また，教育用 OS(udos)の設計法として，中レベル抽象，薄い中間層，追跡性に関する研究も行った．

(5) C 言語におけるシステムヘッダファイル使用の検査ツール

C プログラム中の#include をチェックするツール「簡単#include 検査君」を開発した．教科書用の 158 個の小さなサンプルコードに実験的に適用し，58 個のファイルから#include ミスを発見できた．経験として，ツールは#include ミスの発見に非常に有効だったこと，#include ミスは予想よりもはるかに多かったこと，などの貴重な知見を得た．

表 1 「簡単#include 検査君」を用いた実験結果

サンプルコードのカテゴリ（章）	間違いの個数					
	1 型	2 型	3 型	4 型	5 型	合計
低水準入出力	36	0	0	0	0	36
標準入出力	0	4	0	0	0	4
ファイルシステム	9	1	0	0	0	10
プロセス	10	4	0	0	0	14
ソケット	8	4	0	0	0	12
C の復習	1	0	0	0	0	1
pipe と dup2	6	11	3	0	0	20
setjmp	0	2	0	0	0	2
シグナル	5	6	0	0	1	12
端末	0	3	0	1	0	4
合計	75	35	3	1	1	115

タイプ	間違いタイプの説明
1 型	プラットフォームの違いによるヘッダファイルの過不足
2 型	デバッグ・テスト終了後のヘッダファイル消し忘れ
3 型	不要な#include をした
4 型	必要な#include のし忘れ
5 型	同じヘッダファイルの重複#include

(6) C 言語の前処理系解析ツール

デバッグ情報である DWARF2 はマクロ情報の一部を提供するものの，高精度な CASE ツール構築には十分ではない．この問題を解決するため C 言語の前処理系解析器の研究を行った．従来の方法では，既存の前処理系を修正する方法と，前処理系のエミュレータを用いる方法が知られていたが，どちらも移植性や精度に問題があった．我々の研究では追跡子と呼ぶソースコード埋め込み(source code instrumentation)方式を新たに考案して実装・実験を行った．gcc-4.1.1(約 63 万行)などの実用規模のソースコードに適用して，よい結果を得た．

(7) シグナルによるデータ競合の検出ツール

本プロジェクトがこれまで構築してきた高精度 CASE ツールは主に静的な解析器が中心であったが，動的な解析が必要となるケースも存在する．シグナルによるデータ競合の検出がこのケースに該当する．この研究では，ソースコードの解析や修正を必要とせず，バイナリレベルで高精度にデータ競合を検出する手法を新たに考案して実装・実験を行った．bash-3.0(約 9 万行)などの実用規模のプログラムに適用してよい結果を得た．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計６件）

①権藤克彦,川島勇人,今泉貴史:TBCppA: 追跡子を用いたC前処理系解析器
コンピュータソフトウェア, 査読有, 25-1, 2008, pp.105-123

②田原貴光, 権藤克彦：DRACULA：シグナルによるデータ競合の検出ツール
電子情報通信学会論文誌, 査読有, J91-D[2], 2008, pp.449-458

③大場勝, 権藤克彦：プログラム理解を支援するコンセプトキーワードの自動抽出法 ckTF/IDF 法の提案
情報処理学会論文誌, 査読有, 48-8, 2007, pp.2596-2607

④権藤克彦, 大場勝：中レベル抽象・薄い中間層・追跡性の実践によるコンパクトな教育用オペレーティングシステム udos の設計と実装
電子情報通信学会論文誌, 査読有, J90-D[5], 2007, pp.1194-1208

⑤大場勝, 権藤克彦：プログラム理解のための実装レベル制約とソースコード間の追跡性の整理保存法
電子情報通信学会論文誌, 査読有, J90-D[6], 2007, pp.1445-1461

⑥権藤克彦, 鈴木朝也, 川島勇人：
C 言語用 CASE ツールへの DWARF2 デバッグ情報の応用
コンピュータソフトウェア, 査読有, 23-2, 2006, pp.175-198

〔学会発表〕（計５件）

① T. Tahara, K. Gondow, S. Ohsuga: DRACULA: Detector of Data Races in Signals Handlers, 15th Asia-Pacific Software Engineering Conference (APSEC2008), 2008/12/3, 中国・北京

②K. Gondow, H. Kawashima, T. Imaizumi: TBCppA: a Tracer Approach for Automatic Accurate Analysis of C Preprocessor's Behaviors, 8th IEEE Int. Working Conf. on Source Code Analysis and Manipulation (SCAM2008), 2008/9/28, 中国・北京

③Y. Terashima, K. Gondow: Static Call Graph Generator for C++ using Debugging Information, 14th Asia-Pacific Software Engineering Conference (APSEC2007), IEEE Computer Society Press, 2007/12/5, 愛知県名古屋市

④ M. Ohba, K. Gondow: Maintaining Traceability Links between Implementation-level Restrictions and Source Code for Program Understanding, 10th IASTED Int. Conf. Software Engineering and Applications (SEA2006), page-no.514-146, 2006/11/13, Dallas, TX, USA

⑤ M. Ohba, K. Gondow: Toward Mining "Concept Keywords" from Identifiers in Large Software Projects, Int'l. Workshop on Mining Software Repositories (MSR2005),
pp.48-52, 2005/5/17, St. Louis, Missouri, USA

〔図書〕（計１件）

①冨永和人, 権藤克彦: ピアソンエデュケーション, 例解 UNIX プログラミング教室, 2007, 472 ページ

６．研究組織
(1)研究代表者

権藤 克彦 (GONDO KATSUHIKO)
東京工業大学・大学院情報理工学研究科・准教授
研究者番号：50262283

(2)研究分担者
なし

(3)連携研究者
なし