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

平成 ２９ 年　 ６ 月　 ９ 日現在

機関番号：１３９０１
研究種目：挑戦的萌芽研究
研究期間：2014～2016
課題番号：２６５４００２７

研究課題名（和文）耐改竄性をもつプログラム言語とそのプログラム開発手法の研究

研究課題名（英文）Development methods of programs in tamper-resistant language

研究代表者
酒井　正彦（Sakai, Masahiko）

名古屋大学・情報科学研究科・教授

研究者番号：５０２１５５９７

交付決定額（研究期間全体）：（直接経費）　　2,700,000円

研究成果の概要（和文）：本研究は、プログラム難読化によるソフトウェア保護を目指し、最も難解と考えられている言語Malbolgeによるプログラム手法の開発を目指すものである。研究の結果、C言語サブセットで書かれたプログラムをMalbolgeのワードを拡張したMalbolge20のプログラムに変換できるようになった。
本研究ではまず、ワード長を20桁に伸長したMalbolge20を設計し、インタプリタを作成した。次に、低級アセンブリからMalbolge20をシステム的に作成する手法を構築し、実装した。さらに、C言語サブセットをMalbolge20に変換するコンパイラ構築の基盤となる研究を行い、コンパイラを作成した。

研究成果の概要（英文）：This research aims to develop a construction method for Malbolge, which is the most esoteric programming language, toward software protection by obfuscation. As a result, we succeeded to construct a system to convert C-subset language to Malbolge20, which is a word-length-extended Malbolge.
We first design the Malbolge20 language and construct an interpreter. Next, we developed a systematic method to generate malbolge20 codes from Low-level assembler of Malbolge, and implemented a translator. We further constructed a compiler that transforms C-subset language to Malbolge20.

研究分野：プログラム理論

キーワード：難読プログラム　Malbolge　コンパイラ

１．研究開始当初の背景

ソフトウェアの保護の重要性は広く認識されている。プログラム提供側としては埋め込まれた重要情報が抜き取られないことは重要であるし、利用者側としてはシステムの安全性を保つためにソフトウェアが改竄されていないことを保証することは必須条件である。現在、MD5 値などの一致によるプログラムの原本確認は広く用いられているが、一旦 MD 値が等しい系列対が見つかれば、それに同じ系列を付加しても MD 値は等しいという事実を利用することで通常のコードにおいて MD 値を変化させない改変はそれほど困難ではない。

プログラム難読化は、暗号を用いるのと比較して復号化することなく実行可能であり、復号により生のプログラムが出現し解析や改竄の危険がある暗号と比較すると、プログラム難読化手法は大きな利点を持つ。これまでに、プログラム難読化として制御構造の複雑化や無駄なコードの挿入などの方法が研究されてきているが、原則的には見掛け上の複雑化であり効果が限定される。

一方で、意図的にプログラムの作成や理解が困難になるように設計された言語が提案されている。１９９８年に Ben Olmsted により開発された言語 Malbolge は、それらのうちで最も難解であり、改竄防止には優れているものの、プログラミングは不可能に近いと考えられていた （文献①）。これまでに本研究代表者らは、 Malbolge の低級アセンブラ言語などを開発し、ノウハウを駆使すればなんとか簡単なループプログラムを作成できるレベルに達した（文献②～⑦）。しかしながら、依然としてプログラム作成には匠の技とも呼べるノウハウが必要であった。

２．研究の目的

プログラム難読化は、暗号を用いるのと比較して復号化することなく実行可能であり、復号により生のプログラムが出現し解析や改竄の危険がある暗号と比較すると、プログラム難読化手法は大きな利点を持つ。これまでにも難読化の研究が進んできているが、安全性において暗号化と比較可能なほど決め手となる手法はまだ知られていない。そこで、本研究は、言語設計から最初の Hello world プログラムが開発されるまでに２年を必要としたほど超難解なプログラム言語 Malbolge を用いてプログラム保護手法の基盤の確立を目指す。またプログラム難読化が可能であるかの理論的研究を行う。

３．研究の方法

次の手順で研究を進めた。

(1) ３の１０乗のメモリ空間を３の２０乗に伸長した Malbolge20 を設計し、小型なインタプリタを作成する。

(2) 低級アセンブリプログラムから Malbolge20 をシステム的に作成する手法を構築し、それを実装する。

(3) 比較的単純な手続き型言語のプログラムを Malbolge20 プログラムに変換する、コンパイラ作成のための基盤となる研究を行い、それを基にコンパイラを作成する。

以下ではこれらについて具体的に述べる。

(1) Malbolge20 インタプリタ

低級アセンブリ言語には通常のプログラミング言語が持つような演算命令がなく Malbolge 特有の演算を行う命令のみであり、低級アセンブリ言語でのプログラミングには大きな困難が伴う。これを解決として低級アセンブリプログラム作成のための高級アセンブリ言語が飯澤らによって構築され、安藤らによって改良されている（文献②～⑦）。すなわち既存研究では、高級言語のプログラム→ 高級アセンブ リプログラム→低級アセンブ リプログラム→Malbolge プログラムと変換することで高級言語のプログラムから Malbolge プログラムを得るアプローチをとっている（文献②から④）。しかし実用化以前に圧倒的メモリ不足という問題点がある。これは，Malbolge のメモリは５９０４９（３の１０乗）ワードを持つが、低級アセンブリプログラムの１行分ですら Malbolge プログラムでは 数十から数百ワードになる。例えば数値に１を加算するインクリメント演算を行うだけでも、生成される Malbolge プログラムは全メモリ空間の１０分の１程度消費する。

そこで、ワード長を３の１０乗倍に拡張することにより、この問題を解決した。Malbolge 言語の構文に変更はないが、言語設計の上でワード長が問題になるのは、３進値を右に一桁だけ循環移動しつつロードする命令である rotr のみである。その意味は桁数が２倍となるだけで自然に拡張される。ほとんどの Malbolge プログラムはそのまま動作するが、rotr を１０回繰り返すことで値をロードするプログラムなど、桁数に依存する処理は繰り返し回数を２倍に変更する必要がある。

実行環境であるインタプリタを自然に拡張してしまうと初期化の段階で資源を使いすぎて全く使い物にならない。その原因は、Malbolge の仕様が、メモリの全空間をある法則に従って初期化することにある。そこで、メモリ空間を３の１０乗個のページに分け、各ページを初めて利用する際に初期化できるような情報を最初に準備することで解決した。

(2) 低級アセンブリプログラムから Malbolge20 への変換系の実装

低級アセンブリプログラムは、文献②で提案され、文献②③で実装された。これは Malbolge の各命令が実行後に書換えられてしまうという致命的な困難性を解消するために考案された言語である。このアセンブラを Malbolge20 用に書き換えた。その際の主

な問題点とその解決法は以下のとおりである。
①値の作成方法：Malbolge アセンブラ実現上の困難な点の一つは、Malbolge コードとして許されるデータ値がごく限られているため、動作に必要となるデータを Malbolge の乏しい機能を使って作成する必要がある。この部分は生成される Malbolge コードのかなりの部分を占めるため、あらかじめ表を作成しておくことで最適化なコードを作成していた。しかしながら、この手法を利用して３の２０乗の値を作成するためには表のサイズが大きすぎて作成できないという問題点が発生した。これについては、コードは長くなるものの一桁づつ作成することで解決した。
②配列機能の実現：Malbolge 低級アセンブラでは、空きエリアを１本の配列として使う機能がある。その実装においては、Malbolge が全エリアを opr 命令に基づくフィボナッチ数列のような定義に基づいて初期化するという性質を最大限に利用して、空き領域の値を一つおきにブートストラップ領域の番地になるように細工しておいて、配列の値を書き込んだ後に次の命令に戻ってくるようになっている。しかしながら、Malbolge20 の低級アセンブラが生成するブートストラップ領域が基とは異なるため、IND_OPR と呼ぶ命令ユニットの再設計が必要となった。

(3) C 言語のサブセットから Malbolge20 へのコンパイラ

ソース言語として C 言語の部分言語を採用し、一旦、新たに設計した制御付き擬似命令列に変換の後、低級アセンブリプログラムを経由して Malbolge20 プログラムを生成するコンパイラを作成した。

対象とする C の部分言語は、int 型と bool 型のデータが利用可能で、if 文、while 文、再帰可能な関数、加算演算、インクリメントとデクリメント、比較演算、論理演算、代入文、一文字入出力の機能を持つ。

中間言語として設計した制御付き擬似命令列（発表論文②）は、Malbolge の命令に近く見える演算命令と、ごく基本的な制御命令で構成される。低級アセンブリプログラムへの変換系の設計においては、次の２つの点がとくに困難であった。
①実行の基本制御：低級アセンブラでは、変数の値を参照したり更新する場合には、変数が置かれている数行上に実行する命令を置く必要があり、さらにその実行後には途中の命令をスキップして対象とする変数のすぐ下に実行が移される。このため、命令の実行後、次にどの命令を実行するかに関する情報が消えてしまうという重大な欠点がある。この問題を回避するために、このような制限のないフラグに情報を保持するようにした。
②間接参照ジャンプ：ソース言語の再帰呼び出し機能を実現するためには、低級アセンブリ言語が提供する配列機能をスタックとして利用し、間接参照ジャンプにより関数からの復帰する機能が必要である。低級アセンブリ言語の UNIT 記述機能を用いて、低級アセンブリ言語に DJMP 命令を追加することでこの昨日の実現を可能にした。

４．研究成果

本研究課題による一連の研究により、C 言語サブセットで書かれたプログラムを Malbolge20 プログラムに変換できるようになった。変換によって得られた Malbolge20 コードは入力と比較して巨大であり実行効率も非常に悪いが、入力プログラムのサイズ n に対して O(n^2)程度のプログラムサイズであると見積もっている。また、得られたプログラムの解読困難性の理論的な評価は行えておらず、さらなる研究が必要である。実用的な手法とするためには、アルゴリズムを秘匿したいプログラムの部分のみを Malbolge20 に変換し、それと普通のプログラムを組合せて実行させる仕組みを構築するアプローチがよいと考えている。

＜引用文献＞

① Wikipedia, Malbolge, http://en.wikipedia.org/wiki/Malbolge.
②飯澤 恒、坂部俊樹、酒井正彦、草刈圭一朗、西田直樹、難読プログラミング言語 Malbolge におけるプログラム構成手法、電子情報通信学会技術研究報告、Vol.105, No.129, pp.25-30, 2005.
③長坂 哲、酒井正彦、坂部俊樹、草刈圭一朗、西田直樹、難解言語 Malbolge のチューリング完全性について、電子情報通信学会技術研究報告、Vol.110, No.227, pp.55-60, 2010.
④安藤聡、酒井正彦、坂部俊樹、草刈圭一朗、西田直樹、Malbolge の高級アセンブリ言語への加算命令の追加、日本ソフトウェア科学会第 28 回大会講演論文集、5A-3, pp.1-12, 2011.
⑤安藤聡、酒井正彦、坂部俊樹、草刈圭一朗、西田直樹、Malbolge の高級アセンブリ言語への配列機能の追加、電子情報通信学会技術研究報告、Vol.112, No.23, pp.43-49, 2012.
⑥安藤聡、酒井正彦、坂部俊樹、草刈圭一朗、西田直樹、三値関数を実現する Malbolge 命令列の発見のための SAT エンコーディング、電子情報通信学会技術報告、Vol.112, No.275, pp.7-12, 2012.
⑦安藤聡、酒井正彦、坂部俊樹、 草刈圭一朗、西田直樹、Malbolge 低級アセンブリプログラミングにおける制御命令の配置設計のための SAT ソルバの利用、電子情報通信学会技術報告、Vol.112, No.373, pp.25-30, 2013.

５．主な発表論文等
〔雑誌論文〕（計３件）

①河邉翔平、酒井正彦、西田直樹、関浩之、難読性の高い Malbolge コードを生成するコ

ンパイラのための中間言語、電子情報通信学会技術報告、査読なし、Vol.116, No.127, 2016, pp.105-101.
②Masahiko Sakai and Hidetomo Nabeshima, Construction of an ROBDD for a PB-constraint in Band Form and Related Techniques for PB-solvers, IEICE Transaction on Information and Systems, 査読有、Vol.E98-D, 2015, pp.1121-1127.
③加藤起騎、酒井正彦、坂部俊樹、西田直樹、Malbolge 低級アセンブラにおけるコード配置アドレスの決定法、電子情報通信学会技術報告、査読なし、Vol.114, No.127, 2014, pp.99-104.
〔学会発表〕（計１件）
① Masahiko Sakai and Tatsuki Kato, Esoteric Programming Language Malbolge and its Low-Level Assembler, Meeting of IPSJ Special Interest Group on Programming, Asahikawa, Jul 20 2014.
〔その他〕
ホームページ等
http://www.trs.css.i.nagoya-u.ac.jp/Malbolge/index.html.ja（和文）
http://www.trs.css.i.nagoya-u.ac.jp/Malbolge/index.html.en（英文）

６．研究組織
(1)研究代表者
酒井　正彦　（SAKAI, Masahiko）
名古屋大学・大学院情報学研究科・教授
研究者番号：　５０２１５５９７

(4)研究協力者
加藤　起騎　（KATO, Tatsuki）
河邉　翔平　（KOBE, Shohei）
坂梨　元軌　（SAKANASHI, Genki）