様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２５年　６月　５日現在

機関番号：１３９０１
研究種目：挑戦的萌芽研究
研究期間：2010　～　2012
課題番号：22650003
研究課題名（和文）
ソフトウェアプロテクションのための超難読言語Ｍａｌｂｏｌｇｅの研究
研究課題名（英文）
On Esoteric language Malbolge for software protection
研究代表者
酒井　正彦（MASAHIKO SAKAI）
名古屋大学・情報科学研究科・教授
研究者番号：50215597

研究成果の概要（和文）：　本研究の目的は、プログラムの解読・改ざんが大変困難なプログラム言語である Malbolge を、ソフトウェア保護の目的に利用できるようにすることである。最も大きな問題点である、該当の言語でのプログラム作成の困難性を克服するため、Malbolge のチューリング完全性を示し、Malbolge プログラム作成手法の大枠を確立した。

研究成果の概要（英文）：　The purpose of this research is to develop theories for Malbolge, which is one of the most difficult program languages to read/alter, in order to use the language for software protection.　To overcome the biggest problem, that is the difficulty of writing program, we have shown its Turing completeness and established a basic method to produce programs of Malbolge.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
2010 年度	600,000	0	600,000
2011 年度	700,000	210,000	910,000
2012 年度	700,000	210,000	910,000
総　計	2,000,000	420,000	2,420,000

研究分野：
科研費の分科・細目：
キーワード：

１．研究開始当初の背景

ソフトウェアの保護は、特に、ソフトウェアを利用するシステムの安全性を保つために必要となる技術である。例えば、映像などのコンテンツ配信システムにおいて、その再生ソフトウェアを解読・改ざんによりコンテンツの無制限な複製も可能になりうる。

これまでに、全く解読・改ざん不可能なプログラムを作成することは不可能であり、それにかかる時間を増やすことしか出来ないことが知られている[Barak ら 2001 年]。そのため、これまではプログラムの難読化のために、制御構造を複雑にしたり無駄なコードを挿入する方法、プログラムの一部を暗号化しておいて実行時に復号してから実行する方法、プログラムを複数の断片に切り刻んでおいて実行時に適切な順序で実行する方法などが研究されてきている。しかしながら、これらの現在の技術によって難読化されたプログラムは、多少の時間をかけることによって解読が可能である。

一方で、INTERCAL など、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ると考えられており、改ざん防止には優れていると考えられるものの、プログラミングは

不可能に近いと考えられていた。実際にこれまでに知られているプログラムは、Hello world や文字列を出力するもの、入力をそのまま出力するプログラムぐらいであった。

２．研究の目的

本研究の目的は、プログラムの解読・改ざんが大変困難なプログラム言語である Malbolge を、ソフトウェア保護の目的に利用できるようにすることである。最も大きな問題点は、該当の言語でのプログラム作成が不可能と言われるほど難しいことにある。本研究ではその常識を覆し、Malbolge のチューリング完全性を示し、また、Malbolge プログラム作成手法の確立を目指すものである。

３．研究の方法

以下の手順で研究を進めた。

(1) 99 Bottles of Beer の Malbolge プログラムを作成した経験を生かして、まず、これに用いた繰り返し構造のプログラミングなどの個々の技術を整理・抽象化する。
(2) Malbolge プログラムに変換可能な中間言語を設計する。これは、機械語に対するアセンブル言語に相当する。
(3) チューリング完全な言語のうちで出来る限り単純な言語から、中間言語への変換を与える。これにより Malbolge のチューリング完全性を証明する。
(4) 比較的単純な手続き型言語のプログラムを、Malbolge プログラムに変換するコンパイラを試作する。
(5) 本研究の方法により得られる Malbolge プログラムの難読性を評価する。

４．研究成果

研究成果を述べる前に、準備としてプログラミング言語 Malbolge について述べる。

Malbolge [Olmstead 1998] は仮想機械上で動作する機械語として定義されインタプリタによってその意味が定められている。仮想機械は三つのレジスタ(A,C,D)とメモリ(mem) を持ち、値は三進数十桁(10trits)で表現される。よって 0000000000t ～ 2222222222t の範囲の値が表現でき、メモリのアドレス空間も mem[0]～mem[59048]で定義される。表１に Malbolge の命令を示す。Opr の関数 OP(X,Y) は X と Y の各桁同士で表２の trit 演算 op を行う。以下に関数 OP と, Rot で用いられる関数 ROTR の計算例を示す.

OP(0120120120t, 0001112222t)
= 1001022212t
ROTR(0001112222t) = 2000111222t

表１　Malbolge の命令

命令	表記	説明
i	Jmp	ジャンプ. C:=mem[D].
j	MovD	D レジスタの更新. D:=mem[D].
p	Opr	演算命令. A,mem[D]:=OP(A,mem[D]).
*	Rot	右ローテート. A,mem[D]:=ROTR(mem[D]).
/	Input	入力. A:=getchar().
<	Output	出力. putchar(A).
o	Nop	無操作. 何も行わない.
v	Halt	終了. プログラムの実行を停止.
その他	Nop'	無操作.

表２　trit 演算 op(x,y)

		x		
		0	1	2
y	0	1	0	0
	1	1	0	2
	2	2	2	1

低級アセンブリ言語は、Malbolge プログラムの命令が実行後に書き換えられてしまうという問題を避けてループプログラムを作成するために飯澤、酒井らが設計した言語である。低級アセンブリ言語は Malbolge と同じメモリ空間を持つ仮想機械として定義される。レジスタは PC と A の２つを持ち、値の範囲は Malbolge と同じである。低級アセンブリプログラムは、メモリアドレスを表すラベルを付加可能なデータの列によって定義される。各データは変数と、命令、および、フラグのいずれかであり、一行で記述される。命令には演算命令(U_JMP, U_ROT, U_OPR, U_MOV_PC, U_INPUT, U_OUTPUT)と, U_JMP を除く各演算の復元命令(R_ROT, R_OPR, R_MOV_PC, R_INPUT, R_OUTPUT)がある。このため、実行時には演算命令が実行された後、復元命令が実行されるようにプログラムを記述する必要がある。フラグはフリップフロップの役割を果たしており、これによって実行を制御する。

高級アセンブリ言語は, １を足すなどのごく基本的な機能を持つ言語である。各命令はの引数には変数やラベルを用いることが可能である。その基本的な命令は、

① X:=X+1 を計算するインクリメント命令 INC X
② X:=X-1 を計算するデクリメント命令 DEC X
③ Y:=X の代入命令を行う変数コピー命令 MOV X,Y
④ X が 0 の場合, PC が Y でラベル付けされた命令に飛ぶ条件分岐命令 BRANCH X,Y
⑤ X:=getchar() を計算する入力命令 INPUT X
⑥ putchar(X) を計算する出力命令 OUTPUT X

⑦ 実行を停止する停止命令 STOP

からなる。高級アセンブリプログラムは基本モジュール方式（図 1）により低級アセンブリ言語で実現される。

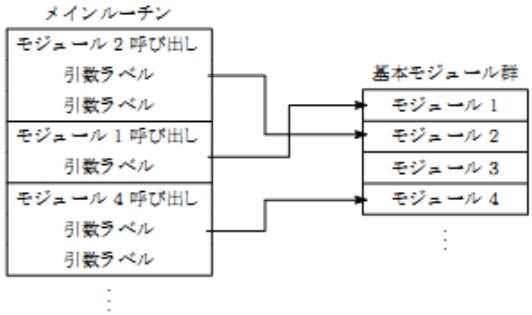

図 1　基本モジュール方式

(1) 弱チューリング完全性[雑誌論文(5)]

Malbolge の弱チューリング完全性、すなわち、入力のサイズに制限がある場合のチューリング完全性を、扱える値の最大値が存在するという制限を持つNプログラムをMalbolgeプログラムに還元することによって示した。この還元のため、以下のようにNプログラム(NNP)の各命令を高級アセンブリ言語で実現した。

① 判定命令：NNP の判定命令の実現は二つの変数X,Y の等しさを判定する必要がある。二つの変数を少なくともどちらか一方が0 になるまでデクリメントを行うことにより、高級アセンブリ言語の条件分岐命令のみで判定命令を実現できる。

② 入力命令：NNP の入力命令は n 個の任意の自然数が入力対象となるのに対して、高級アセンブリ言語の入力命令は1 文字入力(getchar) であるため、実現には各引数毎に以下の様々なモジュールの作成が必要となる。**モジュール 1**：特定の文字が入力されるまで入力を繰り返す。**モジュール 2** ：入力された文字コードを数に変換する。**モジュール 3** ：任意の n 桁の自然数と 1 桁の自然数(0 ～9)を使って n+1 桁の数へと統合する。ここで、引数が n 個の場合にはこれらのモジュール 1～3 を組み合わせて作成したモジュールを n 個並べればよい。

③ 出力命令：NNP の出力命令を実現するためには、以下のモジュールの作成が必要となる。**モジュール a**：出力値の桁数を求める。**モジュール b**：値を最上位桁の数とそれ以外の値に分ける。**モジュール c**：出力する数を ASCII 文字に変換する。

④ その他の命令：1 の加算命令と初期化命令は、高級アセンブリ言語のインクリメント命令と ROT 命令等で容易に実現できる。

(2) 高級アセンブリ言語への加算の追加[雑誌論文(4)]

高級アセンブリ言語には、Malbolge 特有の演算のほかには、1 の加算と減算しか演算を持たない。そこで、加算演算の追加を行った。これには、加算を行うための低級アセンブリプログラムの開発と、それを基本モジュールとして処理系に組み込むことが必要となる。

Malbolge は表 2 に示される 3 進数の演算 op しか使えないため、まず、op を用いて入力の二数に対し桁上げを考えない加算を行う関数 sum（表 3）と、入力の二数の加算に関する桁上げのみの計算を行う関数 carry（表 4）を実現した。

表 3　sum(X,Y)

		x_t		
		0	1	2
y_t	0	0	1	2
	1	1	2	0
	2	2	0	1

表 4　carry(X,Y)

		x_t		
		0	1	2
y_t	0	0	0	0
	1	0	0	1
	2	0	1	1

これらの trit-wise の関数を使って以下のようにアルゴリズムが設計される。

入力　10trits で表される二つの値
出力　入力の二数を足し合わせた数
アルゴリズム

1. 入力の二数を変数 X，Y に入れる
2. X，Y に対して関数 sum の演算を行う
3. X，Y に対して関数 carry の演算を行う
4. 2 の演算結果を X にコピーする
5. 3 の演算結果を左に 1trit シフトし，最下位 trit を 0 クリアの後，Y にコピーする
6. 2 から 5 を 10 回繰り返す
7. X を出力する

(3) 高級アセンブリ言語への配列機能の追加[雑誌論文(3)、学会発表(1)]

Malbolge のチューリング完全性を示すため、扱える値の制限を外す必要がある。このため、間接参照のための手法を開発し、高級アセンブリ言語に配列と同等の機能を追加した。実際には、領域確保命令 ALLOC と間接参照命令 STOREI、LOADI を高級アセンブリ言語に追加した。各命令の仕様は以下のとおりである。VAR ALLOC とはメモリ中で使用可能な範囲の先頭アドレスを格納した特殊な変数であり、INDEX はインデックスレジスタで

ある。

- ALLOC X,Y :
  X:=VAR ALLOC,
  VAR ALLOC:=VAR ALLOC+Y,
  IP:=IP+1
- STOREI X :
  [INDEX]:=X,
  IP:=IP+1
- LOADI X
  X:=[INDEX],
  IP:=IP+1

これら命令の追加はプリプロセッサの拡張, 特殊な命令ユニット INDIRECT UNIT の構築, インタプリタの拡張により表5のように実現した。

表5　命令の実現

ALLOC X,Y	STOREI X	LOADI X
MOD_MOV	MOD_MOV	MOD_MOV
VAR_ALLOC-2	X-2	IN_DIRECT
VAR_RET	VAR_RET	VAR_RET
X-2	IN_DIRECT	X-2
VAR_RET	VAR_RET	VAR_RET
MOD_ADD		
Y-2		
VAR_ALLOC-2		
MOD_ADD		
Y-2		
VAR_ALLOC-2		

(4) 三値関数の実現法[雑誌論文(2)]

Malbolge 特有の演算 op を用いて、sum(表3)や carry(表 4)などの与えられた三値関数を実現する命令列を機械的に求めるための手法を提案した。

まずこの問題を命令列の探索問題として定式化し、この問題をブール論理式の充足可能性判定問題(SAT)に変換する手法を与え、実際に実装・実験を行った。その結果、いかなる三値関数も、二つの変数を利用した長さ 15 以下の命令系列で計算できることが判明した。

(5) 低級アセンブリプログラムの制御構造の設計法[雑誌論文(1)]

低級アセンブリ言語の開発により Malbolge のループプログラムの作成が可能になったものの、低級アセンブリプログラムでは変数を引数とする命令はその変数宣言の直前に記述しなければならず実行制御が必要不可欠なことと、制御命令には無条件ジャンプとアクセスの度にジャンプとスルーが入れ替わるフリップフロップジャンプしか存在しないことから、低級アセンブリ言語でのプログラミングにも困難が伴う。これまで高級アセンブリ言語の実現のための基本モジュールを作成してきたが、その際にはこれらの制御命令の配置設計を非常に苦労して手探りで行った。そこで、Malbolge の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に SAT ソルバを利用した手法を提案することで、この問題の解決を試みた。まず制御命令の配置問題を定式化し、その問題の SAT エンコーディングを提案・実装した。FF ジャンプと呼ばれる命令の種類を最小にする手法 MIN-FKIND と同命令の配置数を最小にする手法 MIN-PNUM の二種類について実験を行い、おおむね命令長 20 程度までであれば、自動配置が可能であることがわかった。

実際のプログラミングでは、ほとんどの場合 20 より長い命令系列を扱う必要があり、今後の研究が期待される。

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５件）

(1) 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, 査読無, Vol.112, No.373, 2013, pp.25-30.

(2) 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, 査読無, Vol.112, No.275, 2012, pp.7-12.

(3) 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報告, 査読無, Vol.112, No.23, 2012, pp.43-49.

(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, 査読無, No.5A-3, 2011, 12 pages.

(5) 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, 査読無, Vol.110, No.227, 2010, pp.55-60, 2010.

〔学会発表〕（計３件）

(1) 安藤聡, 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 における高級アセンブリ言語への加算命令の追加, 第 13 回プログラミングおよびプログラミング言語ワークショップ (PPL2011), 論文集 p.214,

札幌市, 2011 （ポスター・デモ発表）.
(2) 長坂哲, 安藤聡, 酒井正彦, 坂部俊樹, 草刈圭一朗, 西田直樹, 難解言語 Malbolge におけるプログラミング環境の構築と改良, 第 13 回プログラミングおよびプログラミング言語ワークショップ(PPL2011), 論文集 p.214, 札幌市, 2011 （ポスター・デモ発表）.
(3) Masahiko Sakai, Introduction to Esoteric Language Malbolge, Japan-Vietnam Workshop on Software Engineering 2010 (JVSE 2010), Hanoi, pp.15-19, Dec. 2010. (招待講演).

〔その他〕
http://www.trs.cm.is.nagoya-u.ac.jp/Malbolge/

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

(2)研究分担者
坂部　俊樹（SAKABE Toshiki）
名古屋大学・大学院情報科学研究科・教授
研究者番号：60111829

(3)連携研究者　なし
（　）

研究者番号：