Reversible Computation and Reversible Programming Languages

1. 研究分野

可逆コンピューティング
ー可逆プログラミング
ー可逆言語

2. 目的

可逆プログラミング言語について，可逆高級言語Janusを中心にまとめる．

3. 背景

チューリング機械やRAMモデルのような非可逆な計算モデルは，各計算ステップで情報を破棄し，前のテープで書かれていた値やレジスタの値を新しい値に上書きする．通常，これらの計算を逆に辿ることはできない．対して，可逆コンピューティングの計算モデルは，計算の履歴を記憶することで，可逆的な計算を可能にする．したがって，メモリの空きが存在する限り，可逆的な計算が可能である．
　情報の破棄(ビットの破棄)には，物理的に放熱が必要である．したがって，理論的には計算によってビットが消去されなければ，熱は放出されない．可逆コンピューティングでは，計算によるビットの消去は発生しない．したがって，可逆コンピューティングの研究は，低消費電力コンピュータにつながる．

4. アプローチ

JanusはLutzとDerbyによって発明されたプログラミング言語である．ここでは，Janusについて，独自の形式化を用いて拡張したものを取り上げる．Janusは，一般的な可逆言語を設計するためのモデルとなり得る可逆言語である．全ての代入と制御構造が可逆的であり，プロシージャの逆実行が可能である．

言語について

- 可逆的な代入
- 可逆的な制御演算子(if, loop)
- スタック操作(push, pop)
- ローカル変数
- プロシージャ呼び出し(call, uncall)
- スキップ…？
- 整数変数，一次元整数配列，整数スタックの変数宣言
- 配列は0から
- 32ビット整数(符号付き)
- 変数と配列の要素の初期状態はゼロクリア，スタックは空
- グローバル変数はない
- 論理値 true は0以外の任意の整数，false は0

Syntax Domains

$prog \in \text{Progs}$	$s \in \text{Stms}$	$d \in \text{Vdecs}$	$\odot \in \text{ModOps}$
$p \in \text{Procs}$	$e \in \text{Exps}$	$t \in \text{Types}$	$\otimes \in \text{Ops}$
$q \in \text{PIds}$	$x \in \text{Vars}$	$c \in \text{Cons}$	

Grammar

$prog$ ::=	$p_{main}\ p^*$	Janus program
d ::=	$x \mid x\texttt{[}c\texttt{]}$	scalar and array
t ::=	`int` $\mid$ `stack`	data types
p_{main} ::=	`procedure main ()` $(\texttt{int}\ d \mid \texttt{stack}\ x)^*\ s$	main procedure
p ::=	`procedure` $q\texttt{(}t\ x\texttt{,}\ldots\texttt{,}t\ x\texttt{)}\ s$	procedure definition
s ::=	$x\ \odot\texttt{=}\ e \mid x\texttt{[}e\texttt{]}\ \odot\texttt{=}\ e$	assignments
	`if` e `then` s `else` s `fi` $e \mid$	conditional
	`from` e `do` s `loop` s `until` $e \mid$	loop
	`push(`x`,`x`)` $\mid$ `pop(`x`,`x`)` $\mid$	stack modification
	`local` $t\ x$ `=` $e\ \ s\ \$ `delocal` $t\ x$ `=` $e \mid$	local variable block
	`call` $q\texttt{(}x\texttt{,}\ldots\texttt{,}x\texttt{)} \mid$ `uncall` $q\texttt{(}x\texttt{,}\ldots\texttt{,}x\texttt{)} \mid$	procedure invocation
	`skip` $\mid s\ s$	statement sequence
e ::=	$c \mid x \mid x\texttt{[}e\texttt{]} \mid e \otimes e \mid$ `empty(`x`)` $\mid$ `top(`x`)` $\mid$ `nil`	expression
c ::=	`-2147483648` $\mid \cdots \mid$ `0` $\mid$ `1` $\mid \cdots \mid$ `2147483647`	integer constant $(-2^{31}$ to $2^{31}-1)$
$\odot$ ::=	`+` $\mid$ `-` $\mid$ `^`	operator
$\otimes$ ::=	$\odot \mid$ `*` $\mid$ `/` $\mid$ `%` $\mid$ `&` $\mid$ `|` $\mid$ `&&` $\mid$ `||` $\mid$	operator
	`<` $\mid$ `>` $\mid$ `=` $\mid$ `!=` $\mid$ `<=` $\mid$ `>=`	

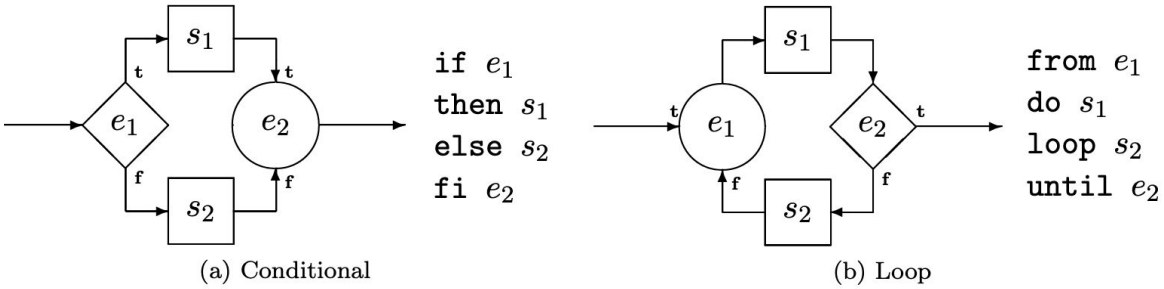
Fig. 1. Syntax of Janus

式と代入

- 可逆代入によって整数変数と配列要素を更新できる
- 可逆代入が変数の値を変更する唯一の方法
- 可逆代入では，代入の左側に右側の式に現れる変数があってはいけない
- 式は定数，変数，配列，バイナリ式，is-empty，スタック
- 算術演算子(+, -, *, /, %)
- ビット演算子(&, |, ^)
- 論理演算子(&&, ||)
- 関係演算子(<, >, =, !, =, <=, >=)

制御構造

- if, loopには事前条件と事後条件が必要



(a) Conditional (b) Loop

Fig. 2. Reversible structured control flow

Janusについて

- 無制限のメモリを持つJanusはr-Turing完全である
- r-Turing完全はどのような可逆チューリング機械も無関係なゴミ情報を出力せずにシミュレーションできるという性質
- 可逆チューリング機械がチューリング機械をシミュレートできるように，Janusもゴミ情報を含み，非可逆関数をプログラム内に表現することができる.
- 無制限のメモリを持つJanusは，その後方決定論的性質から，必ず停止する.

5. 結果

可逆高級言語Janusは，可逆的なプリミティブを高次元で可逆的に構成することで，Janusにより記述されたプログラムの可逆性を保証するものであった.

6. 有用性

低消費エネルギーコンピュータ，量子コンピュータ，プログラム反転や双方向変換など.

7. 限界・短所

8. 次に何を読めばいいか？

Janus(オリジナル)

Lutz, C., Janus: a time-reversible language, Letter to R. Landauer (1986).
http://www.cise.ufl.edu/~ mpf/rc/janus.html

r-Turing完全について

Yokoyama, T., H. Axelsen and R. Glu ¨ck, Reversible flowchart languages and the structured reversible program theorem, in: L. A. I. Damg˚ard, L. A. Goldberg, M. M. Halld´orsson and A. I. I. Walukiewicz, editors, International Colloquium on Automata, Languages and Programming, Proceedings, LNCS 5126, 2008, pp. 258–270.

プログラム反転

Glu ¨ck, R. and M. Kawabe, Derivation of deterministic inverse programs based on LR parsing, in: Y. Kameyama and P. J. Stuckey, editors, Functional and Logic Programming, Proceedings, LNCS 2998 (2004), pp. 291–306.

双方向変換

Foster, J. N., M. B. Greenwald, J. T. Moore, B. C. Pierce and A. Schmitt, Combinators for bi-directional tree transformations: A linguistic approach to the view update problem, ACM Trans. Prog. Lang. Syst. 29 (2007), Article 17, pp. 1–65.

Mu, S.-C., Z. Hu and M. Takeichi, An injective language for reversible computation, in: D. Kozen, editor, Mathematics of Program Construction, Proceedings, LNCS 3125 (2004), pp. 289–313.