引数渡し機構をもつ可逆プログラミング言語の可逆性

田中 秀明 † 新海 由侑 † 横山 哲郎 †
† 南山大学情報理工学部ソフトウェア工学科

1 はじめに

計算過程が可逆とは，任意の状態でその直前と直後の状態が存在すればそれぞれ必ず一意に定まることをいう．可逆プログラミング言語とは，そのプログラムの実行過程が必ず可逆になるような言語設計がなされているプログラミング言語であり，可逆論理回路の設計 [1] やプログラミング言語理論の研究対象 [2] として使われている．そうした言語のひとつである Janus では，構文規則や意味規則における制限によって可逆性を実現しており，その結果，可逆プログラムでのみ可能な逆実行をするための機構を備えている．

現在の Janus[3] は，引数機構が未整備のために可読性や簡潔性が欠如していると考えられる点がある．原因のひとつとして，状態モデルによって識別子のみを渡すことのできる引数渡し機構が用いられていることが挙げられる．このため，識別子以外の式の評価値や参照を渡すことが困難であった．そこで，我々は左辺値を意味領域に含むことのできる環境・記憶域モデルを採用し，可逆性をもつ値渡しと参照渡しの機構を実現する．

Janus では，複合代入演算の左辺に出現した識別子が右辺に出現してはならないという制約があるが，この拡張により，プログラム全域のプログラム解析をしなくてもその複合代入演算文の実行時にその文の情報だけからこの制約の確認ができるようになる．また，引数を伴うプロシージャ呼出しが，より読みやすく簡潔に記述できるようになる．

本稿では，引数渡し機構を拡張して表現力の増した Janus が可逆性をもつことの説明の概略を示す．

2 引数渡し機構をもつ可逆言語

提案する言語の構文規則の一部を図 1 に，意味領域を図 2 に示す．ここで，$x, y \in \mathrm{Vars}$, $q \in \mathrm{PIds}$, $p \in \mathrm{Procs}$ とする．Janus プログラムは変数宣言の列とプロシージャ定義の列からなる．プロシージャは引数渡しの方式が値渡しか参照渡しである一引数をとる．文と式は本稿で必要な部分のみ記述してある．

pp	::= $x^*\ p^*$	Janus プログラム
p	::= `procedure` q`(`$m\ y$`)` s	プロシージャ定義
m	::= `val` | `ref`	引数渡しの方式
s	::= $x\ \odot$`=` e	代入
	| `call` q`(`e`)`	プロシージャ呼出し
	| `uncall` q`(`e`)`	プロシージャ逆呼出し
	| $\cdots$	
e	::= $x \mid e \otimes e \mid \cdots$	式
$\odot$	::= `+` | `-` | `^`	演算子
$\otimes$	::= $\odot$ | `*` | `/` | $\cdots$	演算子

図 1: 構文規則の一部

$v \in \mathrm{Vals}$	=	$\mathbb{Z}_{32}$	値
$l \in \mathrm{Lvals}$	=	$\mathbb{N}$	左辺値
$\gamma \in \mathrm{Envs}$	=	$\mathrm{Vars} \cup \{next\} \rightharpoonup \mathrm{Lvals}$	環境
$\sigma \in \mathrm{Stores}$	=	$\mathrm{Lvals} \rightharpoonup \mathrm{Vals}$	記憶域
$\Gamma \in \mathrm{Pmaps}$	=	$\mathrm{PIds} \rightharpoonup \mathrm{Procs}$	プロシージャ環境

図 2: 意味領域

環境と記憶域は以下のように定義する．環境は識別子と左辺値とを対応付ける後置の部分関数である．環境は後置演算子 $[\cdot \mapsto \cdot]$ によって拡張される：

$$x(\gamma[x' \mapsto l']) = \begin{cases} l' & \textbf{if } x' = x \\ l\gamma & \textbf{otherwise} \end{cases}$$

ここで，$next$ は現在使用されておらず次に使用されることが意図された記憶域の場所を指す特別な識別子である．記憶域は左辺値と値を対応付ける部分関数である．後置演算子 $[l, v_1, v_2]$ は，記憶域の記憶場所 l にある値が v_1 である場合，v_2 に更新する．任意の演算子 $[l, v_1, v_2]$ に対して，その逆関数 $[l, v_2, v_1]$ が存在する．

プログラム実行の初期状態では，$next\,\gamma = 0$ であり，また任意の記憶場所 l に対して $l\sigma = undefined$ である．ここで $undefined$ は未定義を表す特別な値である．

判断

式の評価	$\gamma, \sigma \vdash_{expr} e \Rightarrow v$
文の実行	$\gamma \vdash^{\Gamma}_{stmt} \langle s, \sigma \rangle \Rightarrow \sigma'$

On Reversibility of a Reversible Programming Language with Parameter Passing Mechanisms

† Hideaki TANAKA
† Yoshiyuki SHINKAI
† Tetsuo YOKOYAMA
Department of Software Engineering, Faculty of Information Sciences and Engineering, Nanzan University (†)

$$\frac{x\gamma = l \quad l\sigma = v_1 \quad \gamma, \sigma[l, v_1, undef] \vdash_{expr} e \Rightarrow v \quad v_2 = [\![\odot]\!](v_1, v)}{\gamma \vdash^{\Gamma}_{stmt} \langle x \ \odot\texttt{=}\ e, \sigma\rangle \Rightarrow \sigma[l, v_1, v_2]} \ \textsc{AssVar}$$

$$\frac{\Gamma(q) = \texttt{procedure}\ q\texttt{(val}\ y\texttt{)}\ s \qquad \gamma, \sigma \vdash_{expr} e \Rightarrow v \qquad \gamma, \sigma' \vdash_{expr} e \Rightarrow v \qquad \gamma[y \mapsto l][next \mapsto l+1] \vdash^{\Gamma}_{stmt} \langle s, \sigma[l, undef, v]\rangle \Rightarrow \sigma'}{\gamma[next \mapsto l] \vdash^{\Gamma}_{stmt} \langle \texttt{call}\ q\texttt{(}e\texttt{)}, \sigma\rangle \Rightarrow \sigma'[l, v, undef]} \ \textsc{CallVal}$$

$$\frac{\Gamma(q) = \texttt{procedure}\ q\texttt{(ref}\ y\texttt{)}\ s \qquad x\gamma = l' \qquad \gamma[y \mapsto l'][next \mapsto l] \vdash^{\Gamma}_{stmt} \langle s, \sigma\rangle \Rightarrow \sigma'}{\gamma[next \mapsto l] \vdash^{\Gamma}_{stmt} \langle \texttt{call}\ q\texttt{(}x\texttt{)}, \sigma\rangle \Rightarrow \sigma'} \ \textsc{CallRef} \qquad \frac{\gamma \vdash^{\Gamma}_{stmt} \langle \texttt{call}\ q\texttt{(}e\texttt{)}, \sigma'\rangle \Rightarrow \sigma}{\gamma \vdash^{\Gamma}_{stmt} \langle \texttt{uncall}\ q\texttt{(}e\texttt{)}, \sigma\rangle \Rightarrow \sigma'} \ \textsc{Uncall}$$

図 3: 意味規則の一部

を定義し，意味規則の一部を図 3 に与える．

3 引数渡し機構の可逆性

拡張した Janus の可逆性は次の定理から導かれる．

定理 1 (文の実行の判断の可逆性)．任意の文 s, プロシージャ環境 Γ, 環境 γ, 記憶域 σ, σ', σ'' に対して $\gamma \vdash^{\Gamma}_{stmt} \langle s, \sigma\rangle \Rightarrow \sigma'$ かつ $\gamma \vdash^{\Gamma}_{stmt} \langle s, \sigma\rangle \Rightarrow \sigma''$ ならば，$\sigma' = \sigma''$ である．また，任意の文 s, プロシージャ環境 Γ, 環境 γ, 記憶域 σ, σ', σ'' に対して $\gamma \vdash^{\Gamma}_{stmt} \langle s, \sigma'\rangle \Rightarrow \sigma$ かつ $\gamma \vdash^{\Gamma}_{stmt} \langle s, \sigma''\rangle \Rightarrow \sigma$ ならば，$\sigma' = \sigma''$ である．

この定理の証明は導出に関する帰納法で行われる．

最後に導出される規則が AssVar の場合の結論の可逆性は，式の評価の一意性，関数 $[\![\odot]\!]$ の第 1 引数についての単射性，および記憶域の更新の単射性から示される．式 e の評価は，x の記憶場所 l を未定義として行われるので，右辺に記憶場所 l の変数が現れた場合，計算が停止する．例えば，`x-=y` という文は，`x` と `y` が同じ記憶場所を指す場合，その両方の値を 0 にするのではなく，計算を停止させる．文献 [3] の Janus とは異なり，こうした実行時のエイリアシングの検査には，プログラム全域のプログラム解析は不要である．

プロシージャ呼出しでは，仮引数と実引数の結合によって環境が拡張される．こうした環境の拡張が可逆になるように，すべての意味規則は，結論の環境 γ の一番右側ある $[next \mapsto \cdot]$ をその前提の任意の環境でも一番右側にしている．最後に導出される規則がそれぞれのプロシージャ呼出しの場合の結論の可逆性は，Γ の単射性，式の評価の判断の一意性，記憶域の更新の単射性から示される．

値渡しの変数束縛は，規則 CallVal によって表現されている．通常の値渡しでは，プロシージャ呼出しから戻ってきたとき仮引数に割り当てられた記憶場所が解放されるが，その直後にはその仮引数値が何であったか再構築することができない．したがって，直前の記憶域が一意でないので可逆ではなくなる．そこで，プロシージャ呼出しから戻って仮引数を解放するとき，仮引数値が実引数値に一致しなければならないという制約を課した．このことは記憶域の更新 $[l, v, undef]$ によって表現されている．

参照渡しは実引数の左辺値を仮引数に渡す引数渡しの方法であり，規則 CallRef によって表現されている．参照渡しの場合，仮引数が実引数の値を間接参照するため，仮引数の値の変更が実引数の値にも反映されることになる．このため，値渡しとは異なり，プロシージャ呼出しから戻ってきた地点でも，仮引数の値は一意に再構築できる．

どちらの引数渡しの方式でもプロシージャ逆呼出しには規則 Uncall が使われる．プロシージャ逆呼出しの規則の前提にプロシージャ呼出しの判断を用いることは，プログラミング言語全体を注意深く可逆に設計して初めてできることである．

本稿では，1 引数のプロシージャのみを考えたが，複数引数で複数の引数渡し方式を用いたプロシージャについても，本稿の手法を拡張すれば実現できる．

謝辞 本研究の一部は JSPS 科研費 25730049 の助成を受けたものである．

参考文献

[1] Wille, R. and Drechsler, R.: *Towards a Design Flow for Reversible Logic*, Springer-Verlag (2010).

[2] Mogensen, T. Æ.: Partial Evaluation of the Reversible Language Janus, *Proc. Partial Evaluation and Program Manipulation*, ACM Press, pp. 23–32 (2011).

[3] Yokoyama, T., Axelsen, H. B. and Glück, R.: Principles of a Reversible Programming Language, *Proc. Computing Frontiers*, ACM Press, pp. 43–54 (2008).