

論文

E 同一化を用いたマイクロプログラム自動合成手続き

非会員 木下 貴史[†] 正 員 直井 徹^{††} 正 員 今井 正治[†]

A Procedure for Automatic Microprogram Synthesis Using E-Unification

Takafumi KINOSHITA[†], *Nonmember*, Tohru NAOI^{††} and Masaharu IMAI[†], *Members*

あらまし Zhu, Johnson は抽象データ型に基づくハードウェアアーキテクチャの仕様記述法を提案し、そのもとでマイクロプログラムを自動合成する問題を逐次化問題として定式化した。本論文では、逐次化問題を解く非決定的な準手続きを提案する。この手続きは E 同一化手続きを用いて探索空間の枝刈りを行う。また、基本的な構成をもつ CPU の仕様記述と逐次化問題の例を与え、手続きの実行例を示す。最後に、この実行例により E 同一化手続きを用いた枝刈りの効果が極めて高いことをみる。

キーワード 抽象データ型, E 同一化, マイクロプログラム, ナローイング, 逐次化問題

1. ま え が き

近年、抽象度の高い仕様記述からハードウェアシステムを自動合成する技術が注目されている⁽¹⁾。このような設計技術では、要求された演算に必要なデータパス構成を決定する処理と、その演算を実行するマイクロ操作列を求める処理が中心的な役割を果たす。本論文では、後者に関連する問題、すなわち、要求された計算がデータパスに含まれる機能モジュールを使用して有限クロックサイクル内で実行可能であるか否かを判定し、可能な場合にはその計算のためのマイクロ操作列を求める問題を考える。

Zhu, Johnson は抽象データ型の概念に基づくハードウェアアーキテクチャの仕様記述法を提案し、この枠組みの中で、先の問題を逐次化問題 (serialization problem) と呼ばれる決定問題として定式化した⁽²⁾。文献(2)ではこの問題が一般に決定不能 (準決定可能) であること、また仕様に含まれる等式集合が空のとき決定可能であることが示されている。しかし、等式集合が空でない場合に逐次化問題を効率的に解く方法につ

いては議論されていない。

本論文では、等式集合が完備な項書換え系とみなせると仮定し、逐次化問題の準手続きを提案する。この手続きはマイクロ操作列を非決定的に生成し、要求された計算が実現できたかどうかを検査する方式を用いるが、この基本的動作に加え、E 同一化手続きにより解が存在しない場合の探索の打ち切りを試みる。この操作により、探索空間の枝刈りや、解が存在しない場合の手続きの停止が期待できる。

本論文の構成は以下のとおりである。2. では抽象データ型や項書換え系などの基本概念について述べ、Zhu, Johnson による仕様記述法を紹介する。また、E 同一化について述べる。3. では逐次化問題の準決定手続きを示す。4. ではまず、Zhu, Johnson の仕様記述法に基づき、レジスタをビット列として扱う方法を提案する。次に、算術論理演算に必要な基本演算器 (ALU, シフタ) を備えた CPU の仕様記述を与える。最後に、逐次化問題の例に本手続きを適用し、枝刈りの効果について調べる。この結果、E 同一化手続きを用いた枝刈りにより、手続きの実行時間および探索空間の爆発が大幅に縮小されることが知られた。

2. 準 備

本章では、抽象データ型、項書換え系などの基本概念、Zhu, Johnson⁽²⁾ によるアーキテクチャ仕様記述法、

[†] 豊橋技術科学大学情報工学系, 豊橋市
Faculty of Engineering, Toyohashi University of Technology,
Toyohashi-shi, 441 Japan

^{††} 岐阜大学工学部, 岐阜市
Faculty of Engineering, Gifu University, Gifu-shi, 501-11 Japan

および E 同一化手続きについて述べる^{(3),(4)}.

2.1 基本概念

\mathcal{S} をソートの集合とする. ソート $s \in \mathcal{S}$ の関数記号集合と変数記号集合をそれぞれ $\mathcal{F}_s, \mathcal{X}_s$ と表す. 任意の $s, s' \in \mathcal{S}$ について $\mathcal{F}_s \cap \mathcal{F}_{s'} = \emptyset$ とし, $s \neq s'$ なる $s, s' \in \mathcal{S}$ について $\mathcal{F}_s \cap \mathcal{F}_{s'} = \emptyset, \mathcal{X}_s \cap \mathcal{X}_{s'} = \emptyset$ とする. \mathcal{F}_s の要素 f にはランクと呼ばれるソートの列 $\langle s_1, s_2, \dots, s_n, s \rangle$ ($s_i \in \mathcal{S}$) が一意に対応する. $\langle s_1, s_2, \dots, s_n \rangle$ を f のアリティ, s を f のソートと呼ぶ. f が定数記号の場合, そのランクは $\langle s \rangle$ である.

$\mathcal{F} = \bigcup_{s \in \mathcal{S}} \mathcal{F}_s, \mathcal{X} = \bigcup_{s \in \mathcal{S}} \mathcal{X}_s$ とする. \mathcal{F} と \mathcal{X} から生成されるソート $s \in \mathcal{S}$ の項全体の集合を $\mathcal{T}_s(\mathcal{F}, \mathcal{X})$ と表し, $\mathcal{T}(\mathcal{F}, \mathcal{X}) = \bigcup_{s \in \mathcal{S}} \mathcal{T}_s(\mathcal{F}, \mathcal{X})$ とする.

等式は項の対 $t = s$ である. E を等式の集合とする. 二つの項 t, s が等式理論 E を法として同値であることを $t =_E s$ と表し, t と s は E 同値であると言う.

4 項組 $\langle \mathcal{S}, \mathcal{F}, \mathcal{X}, E \rangle$ を抽象データ型仕様と言う.

代入は \mathcal{X} から $\mathcal{T}(\mathcal{F}, \mathcal{X})$ への写像である. 次式

$$\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n)) \quad (1)$$

により, 代入の定義域を $\mathcal{T}(\mathcal{F}, \mathcal{X})$ 上へ拡張する. 代入 σ に対し, $\mathcal{D}(\sigma) = \{x \mid \sigma(x) \neq x, x \in \mathcal{X}\}$ とする. $\mathcal{D}(\sigma) = \{x_1, \dots, x_n\}$ で $\sigma(x_i) = t_i$ であるとき, σ を $\langle x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n \rangle$ と表す.

\mathcal{D}^* を正整数列の集合とし, その要素を出現と呼ぶ. Λ を空列とする. 項 t の, 出現 $u \in \mathcal{D}^*$ における部分項 t/u を次のように定める. すなわち, $t/\Lambda = t$ とし, また $t/u = f(s_1, \dots, s_m)$ のとき, $t/(u \cdot k) = s_k$ ($1 \leq k \leq m$) とする. 項 t に対し, 集合 $\{u \in \mathcal{D}^* \mid t/u \text{ は } t \text{ の部分項}\}$ を $\mathcal{O}(t)$ と表し, また $\bar{\mathcal{O}}(t) = \{u \in \mathcal{O}(t) \mid t/u \in \mathcal{X}\}$ とする. 項 t と s について, t の部分項 t/u を s で置き換えた項を $t[u \leftarrow s]$ と表す.

項 t に含まれる変数記号の集合を $\mathcal{V}(t)$ と表す. 項書換え系 R は, 書換え規則と呼ばれる項の順序対 $l \rightarrow r$ の集合である. 但し書換え規則 $l \rightarrow r$ は $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ を満たすとする. 項 t の部分項 t/u と書換え規則 $l \rightarrow r$ について $\sigma(l) = t/u$ なる代入 σ が存在し, $t' = t[u \leftarrow \sigma(r)]$ であるとき, $t \rightarrow_{Rt'} t'$ と表し, \rightarrow_R を R による書換え関係と言う. \rightarrow_R の反射推移閉包, 反射対称推移閉包をそれぞれ, $\xrightarrow{*}_R, \xleftarrow{*}_R$ と表す. 項 t に対し, $t \xrightarrow{*}_R s$ となる項 s が存在しないとき, t を R 正規形と言う. $t \xrightarrow{*}_R s$ で s が R 正規形であるとき, s を $t \downarrow_R$ と表す. どのような項 t に対しても, 無限の書換え列 $t \rightarrow_R t_1 \rightarrow_R t_2 \rightarrow_R \dots$ が存在しないとき, 項書換え系 R は停止性をもつと言われる. また R が合流性をもつとは, $t \xrightarrow{*}_R s_1$ かつ $t \xrightarrow{*}_R s_2$

であるすべての項 t, s_1, s_2 について, $s_1 \xrightarrow{*}_R t'$ かつ $s_2 \xrightarrow{*}_R t'$ なる項 t' が存在することを言う. 停止性と合流性をもつ項書換え系は完備であると言われる. 項書換え系 R が完備であるとき, 項 t の R 正規形は一意的に定まる⁽⁵⁾. 二つの書換え規則 $l_1 \rightarrow r_1, l_2 \rightarrow r_2$ と $u \in \bar{\mathcal{O}}(l_1)$ について, $\sigma(l_1/u) = \sigma'(l_2)$ となる最も一般的な代入 σ, σ' が存在するとき, 項の対 $\langle \sigma(l_1)[u \leftarrow \sigma'(r_2)], \sigma(r_1) \rangle$ を危険対と呼ぶ. 但し, 二つの書換え規則が同一で[†], かつ $u = \Lambda$ である自明な場合を除く. 次の二つの結果は項書換え系の性質としてよく知られている⁽⁵⁾.

[命題 1] 項書換え系 R が停止性をもつとする. R が合流性をもつための必要十分条件は, すべての危険対 $\langle t_1, t_2 \rangle$ に対し, $t_1 \xrightarrow{*}_R s$ かつ $t_2 \xrightarrow{*}_R s$ となる s が存在することである. \square

[命題 2] 等式理論 E と完備な項書換え系 R_E において $=_E$ と $\xrightarrow{*}_{R_E}$ が一致するとき, $t =_E t'$ であるための必要十分条件は $t \downarrow_{R_E} = t' \downarrow_{R_E}$ が成り立つことである. \square

2.2 アーキテクチャ仕様記述法と逐次化問題

本節では, Zhu, Johnson⁽²⁾ による抽象データ型を用いたアーキテクチャ仕様記述法と逐次化問題を, 若干の改良と共に与える. Zhu, Johnson のアーキテクチャ仕様記述法は, 抽象データ型仕様にデータパスを特定するための記述を付加し, ハードウェアシステムの仕様を代数的に記述する.

$\langle \mathcal{S}, \mathcal{F}, \mathcal{X}, E \rangle$ を抽象データ型仕様とする. \mathcal{X} と素な変数記号集合 $\mathcal{R} = \{r_1, r_2, \dots, r_N\}$ をレジスタ変数記号集合と呼ぶ. 但し, \mathcal{R} の各要素のソートは \mathcal{S} の要素とする. 代入 $c: \mathcal{R} \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{R})$ をデータパス代入と呼ぶ. データパス代入は 1 クロックで実行可能なレジスタ間データ転送, すなわちマイクロ操作を表現している. データパス代入の任意の有限集合を制約集合と呼ぶ. 制約集合は, 仕様により特定するハードウェアで実行できるマイクロ操作の全体を表す^{††}.

[例 1] レジスタ $\mathcal{R} = \{r\}$ と機能モジュール f_1, f_2 からなる図 1 のデータパスを考える. f_1, f_2 は 1 引数の関数

† ある書換え規則の変数を名前替えることにより, それが他の書換え規則に一致するとき, これら二つの規則は同一であるものとみなす.

†† Zhu, Johnson による定義では, データパス代入 c が制約集合の要素であれば, 任意の $\mathcal{O} \subseteq \mathcal{R}$ 上への c の制限 $c|_{\mathcal{O}}$ も制約集合に含まれた. これは, 1 クロックで実行可能なデータ転送 c の, 任意の部分的なデータ転送 $c|_{\mathcal{O}}$ もまた 1 クロックで実行可能であることを意味するが, このことは一般に成り立つとは限らないため, 本論文ではこの条件を課さない.

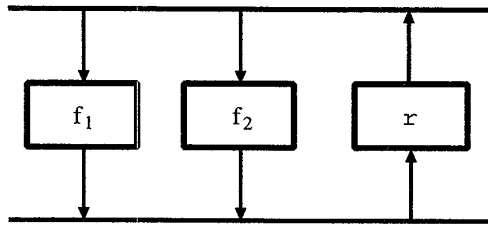


図1 例1のデータパス
Fig. 1 Datapath in Example 1.

記号とし、 r のソートと f_1, f_2 のソート、アリティは同じとする。 f_1, f_2 は一つのバスに接続されているため、並列に動作させることはできない。従ってこのデータパスに対する制約集合は $\{\langle r \leftarrow f_1(r) \rangle, \langle r \leftarrow f_2(r) \rangle\}$ となる。 \square

二つのデータパス代入、 c_1, c_2 に対応するマイクロ操作を、この順序でレジスタ r に適用した結果は、 $c_1(c_2(r))$ と表されることに注意されたい。すなわち、 c_1, c_2 の代入としての適用順序と、マイクロ操作としての適用順序は逆になる。

\mathcal{R} をレジスタ変数記号集合、 \mathcal{C} を制約集合とする。6項組 $\langle \mathcal{S}, \mathcal{T}, \mathcal{X}, E, \mathcal{R}, \mathcal{C} \rangle$ をアーキテクチャ仕様と言う。

v を \mathcal{T} に含まれない関数記号とする。 $v(t_1, \dots, t_n)$ ($t_i \in \mathcal{T}(\mathcal{T}, \mathcal{R})$; $i=1, \dots, n$)をベクトル項と呼び[†]、以後 $v(t_1, \dots, t_n)$ を $[t_1, \dots, t_n]$ と略記する。ベクトル項に対する代入を式(1)と同様に定義する。

[定義1] 仕様 $\langle \mathcal{S}, \mathcal{T}, \mathcal{X}, E, \mathcal{R}, \mathcal{C} \rangle$ のもとで、与えられたベクトル項 $\bar{w} = [w_1, \dots, w_N]$ に対し、

$$c_1(c_2(\dots c_n(\bar{r}) \dots)) =_E \bar{w} \quad (\text{但し } \bar{r} = [r_1, \dots, r_N]; r_i \in \mathcal{R}) \quad (2)$$

を満たすデータパス代入の列 $c_1, c_2, \dots, c_n \in \mathcal{C}$ が存在するかどうかを決定する問題を逐次化問題と言う。 \square

逐次化問題は、 \bar{w} で表される演算を行うマイクロプログラムを生成せよ、という問題である。 \bar{w} に対する逐次化問題の解が存在するとき、 \bar{w} は逐次化可能であると言われる。Zhu, Johnsonは、逐次化問題が一般に決定不能(準決定可能)であり、また、等式集合 E が空のとき決定可能であることを示した⁽²⁾。

2.3 E 同一化手続き

E を等式集合とする。二つの項 t, s に対し、代入 σ が存在して $\sigma(t) =_E \sigma(s)$ となるとき、 t と s は E 同一化すると言われ、 σ は t と s の E 同一化子と呼ばれる。また、 t, s に対し代入 σ が存在して $\sigma(t) =_E s$ となるとき、 t は s へ E 照合すると言われ、 σ は t から s への

E 照合子と呼ばれる。 t と s が共通の変数記号をもたない場合、 t から s への E 照合は s に含まれる変数記号を定数記号とみなした E 同一化である⁽⁴⁾。 E 同一化可能性は一般に決定不能である⁽⁴⁾。

E 同一化手続きとは、 E 同一化可能な二つの項に対し、少なくとも一つの E 同一化子を求める手続きである。 $=_E =^*_{R_E}$ であるような完備な項書換え系 R_E が存在する場合、基底ナローイング⁽³⁾を利用した E 同一化手続きが構成できる^{††}。手続きの詳細については、文献(3)、(6)および(7)を参照されたい。この手続きによる E 同一化の決定可能性に関し、次の性質が導かれる。

項書換え系 R について、関数記号 f が、 R のいずれの書換え規則の左辺にも最外側の記号として現れない場合、 f を構成子記号と呼ぶ。

[命題3] R_E が完備であり、 R_E に含まれるすべての書換え規則の右辺が、変数記号、または構成子定数記号であるとき、 E 同一化可能性は決定可能である。

(略証) 文献(3)の命題1と命題2から直ちに導かれる。 \square

例えば $\text{and}(x, 0) \rightarrow 0$, $\text{and}(x, 1) \rightarrow x$ といった形の論理演算の定義について、命題3が成り立つ。4.では、このような論理演算の記述をもつ仕様のもとでの逐次化問題の例をみる。 E 同一化が決定可能な等式集合により、どのような仕様が記述できるか、また、どのような逐次化問題が扱えるかについて、その見通しをむすびで述べる。

3. 逐次化手続き

本章では、逐次化問題を解くための非決定的な準手続きを示す。この手続きは項列

$$\bar{r}, c_1(\bar{r}), c_2(c_1(\bar{r})), \dots, c_k(\dots c_2(c_1(\bar{r})) \dots), \dots$$

を順次生成してゴール \bar{w} と E 同値な項が現れるかどうか検査し、更に各項から \bar{w} への E 照合可能性の検査により解が存在しない場合の手続きの打ち切りを試みる。項列におけるデータパス代入の適用順から知られるように、この手続きはマイクロ操作列を後ろ向きに推定する。はじめに逐次化可能性と E 同一化可能性の間の

[†] 厳密には次のような議論になる。 \mathcal{S} に含まれないソート x と、 \mathcal{T} と素な関数記号集合 $V = \{v_{\langle s_1, \dots, s_n, x \rangle} \mid s_1, \dots, s_n \in \mathcal{S}, n=1, 2, \dots\}$ が与えられているとする。 $v_{\langle s_1, \dots, s_n, x \rangle}$ のランクは $\langle s_1, \dots, s_n, x \rangle$ である。 t_1, \dots, t_n のソートをそれぞれ s_1, \dots, s_n とすると、 $v_{\langle s_1, \dots, s_n, x \rangle}(t_1, \dots, t_n)$ をベクトル項と呼ぶ。

^{††} 基底ナローイングを用いた方法は単に(本論文の意味での) E 同一化手続きであるだけでなく、 E 同一化子の完備集合⁽³⁾を枚举するという、より強い性質をもっているが、本論文ではこの性質を用いない。

関係について述べる。

[補題 1] 項 \bar{w} が式 (2) を満たす (逐次化可能である) とき, 任意の k ($1 \leq k \leq n$) について $c_k(c_{k+1}(\dots c_n(\bar{r}) \dots))$ は \bar{w} に E 照合する。

(証明) E 照合の定義より明らか。 \square

従って, データパス代入の列 c_k, c_{k+1}, \dots, c_n について $c_k(c_{k+1}(\dots c_n(\bar{r}) \dots))$ が \bar{w} へ E 照合しないなら, これを延長した列 $c_1, c_2, \dots, c_k, \dots, c_n$ により $c_1(c_2(\dots c_k(\dots c_n(\bar{r}) \dots)) \dots) =_E \bar{w}$ とすることはできない。この性質から, 次の手続きが導かれる。仕様の等式集合 E に対応する完備な項書換え系 R_E が存在し, E 照合可能性は決定可能であることを仮定する。

[逐次化手続き]

●入力:

○アーキテクチャ仕様 $\langle \mathcal{A}, \mathcal{F}, \mathcal{X}, E, \mathcal{R}, \mathcal{C} \rangle$

○ $\bar{w} = [w_1, \dots, w_N]$

●出力: データパス代入の列

初期化: $\bar{r} = [r_1, \dots, r_N]$, $\bar{t} := \bar{r}$, $k := 0$, c_k を恒等代入とする。

以下を繰り返す:

(ステップ 1) $\bar{t} \downarrow_{R_E} = \bar{w} \downarrow_{R_E}$ か否かを調べる (命題 2)。

(i) $\bar{t} \downarrow_{R_E} = \bar{w} \downarrow_{R_E}$ の場合

データパス代入の列 c_k, c_{k-1}, \dots, c_1 を出力して停止 (成功)。

(ii) $\bar{t} \downarrow_{R_E} \neq \bar{w} \downarrow_{R_E}$ の場合

\bar{t} から \bar{w} への E 照合子を求める。 E 照合子が求められなければ停止 (失敗)。

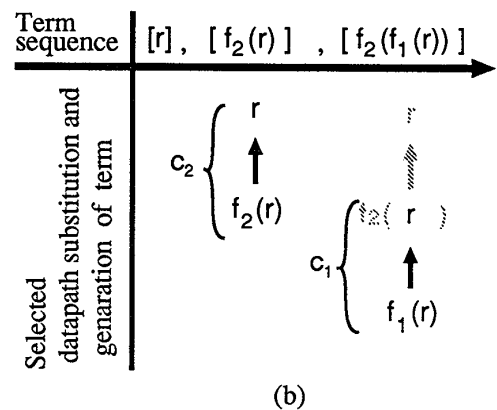
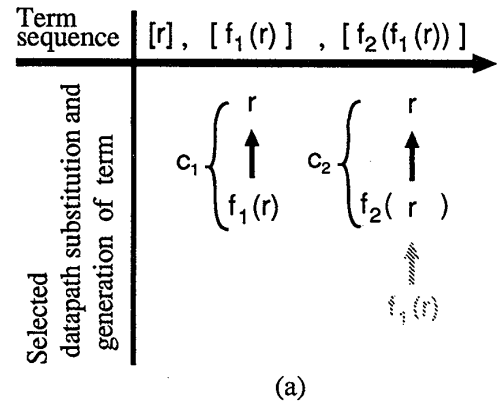
(ステップ 2) $k := k+1$ とする。 $c \in \mathcal{C}$ を非決定的に一つ選び $c_k := c$, $\bar{t} := c(\bar{t})$ とする。 \square

この手続きでは完備な項書換え系 R_E の存在を仮定し, 命題 2 による E 同値性の決定手続きを利用する。この場合, 上記の手続きのほかに, 次のような手続きも考えられる。すなわち, \bar{w} と E 同値な項が現れるまで, 項列

$\bar{r}, c_1(\bar{r}), c_1(c_2(\bar{r})), \dots, c_1(c_2(\dots c_k(\bar{r}) \dots)), \dots$

を順次生成する手続きである。項列を生成する際のデータパス代入の適用順から知られるように, この手続きはマイクロ操作列を前向きに推定する。逐次化手続きと前向き推定手続きの動作の違いを次の例で考える。

[例 2] 図 1 のデータパスについて, 例 1 の制約集合と, 等式集合 $\{f_2(f_1(x)) = h(x)\}$ をもつ仕様を考える。 $\langle r \leftarrow f_1(r) \rangle$ を c_1 , $\langle r \leftarrow f_2(r) \rangle$ を c_2 と表す。ゴール項 $[h(r)]$ の解 c_1, c_2 を求めるときの各手続きの様子を図 2 に示す。



(a) Forward procedure
(b) Serialization procedure

図 2 前向き推定手続きと逐次化手続きにおける項列生成
Fig. 2 Term sequence generation in forward procedure and serialization procedure.

前向き推定手続き (図 2 (a)) は c_1, c_2 の順にデータパス代入を選び, 項列 $[f_1(r)], [f_2(f_1(r))]$ を順次生成する。後者はゴール項と E 同値であるから手続きは解を出力し, 停止する。一方, 逐次化手続き (図 2 (b)) は, 前向き推定手続きとは逆に c_2, c_1 の順にデータパス代入を選び, 項列 $[f_2(r)], [f_2(f_1(r))]$ を順次生成する。従って同様の解を出力し, 停止する。 \square

次に, E 照合可能性の検査が行われる場合の逐次化手続きの振舞いを, 上述の例で考える。はじめに得た項 $[f_2(r)]$ は $[h(r)]$ へ E 照合するので手続きは続行され, 例と同様に $[f_2(f_1(r))]$ を得て解を求める。一方, はじめにデータパス代入 c_1 を選び項 $[f_1(r)]$ を得る場合, この項は $[h(r)]$ に E 照合しないので手続きは打ち切られる。 E 照合の定義から知られるように, E 照合可能性の検査による枝刈りは, 解を後ろ向きに推定する手続きにのみ利用可能であることを注意されたい。

[定理 1] 仕様に含まれる等式集合に対して完備な項書換え系が存在し, かつ E 照合可能性が決定可能である

とき、この仕様のもとで \bar{w} が逐次化可能ならば、逐次化手続きは解を出力して停止する。

(証明) 仮定と補題 1 より、データパス代入 $c_1, c_2, \dots, c_n \in \mathcal{C}$ が存在し、項列 $\bar{r}, c_1(\bar{r}), c_2(c_1(\bar{r})), \dots, c_k(c_{k-1}(\dots c_1(\bar{r})\dots))$ の各項は \bar{w} と E 照合する。従って逐次化手続きはこのデータパス代入列を出力して停止する。 \square

非決定的探索を幅優先探索で置き換えた逐次化手続きを、ワークステーション (SUN SPARC station 2) 上に C 言語により実現した。次章では逐次化問題の例を与え、本手続きの実行例を示す。

4. 逐次化手続きの実行例

本章ではまず、2.2 の仕様記述法に基づきレジスタをビット列として記述する方法を示す。次に、この方法を用いて基本的な CPU の仕様記述を与える。最後に逐次化問題の例を与えて前章の手続きにより解を求め、手続き実行中の枝刈りの効果について検討する。

4.1 レジスタのビット列表現

本節では、レジスタをビット列として記述する方法を示す。次節で見るように、この記述法に従うとレジスタ同士の演算が論理演算だけで記述されるため、逐次化手続き中の E 照合で使われる演算定義も論理演算の定義だけに限られる。従って、2.3 で述べたことから、逐次化手続き中の E 照合の停止性の観点で有利な記述法と考えられる[†]。

仕様記述で用いる抽象データ型 \mathcal{B} を

$\langle \{bit, bitlist\}, \mathcal{F}, E, \mathcal{X} \rangle$

とする。但し $\mathcal{X} = \mathcal{X}_{bit} \cup \mathcal{X}_{bitlist}$, $\{d, 0, 1, nil\} \subseteq \mathcal{F}$ とする。 $d, 0, 1, nil$ のランクはそれぞれ $\langle bit, bitlist, bitlist \rangle, \langle bit \rangle, \langle bit \rangle, \langle bitlist \rangle$ である。 \mathcal{B}

は 2 進数表現を扱うことができる。例えば 4 ビットの 2 進数 0011 は $d(1, d(1, d(0, d(0, nil))))$ で表現される。以後、 $d(t_1, d(t_2, d(\dots, d(t_n, nil)\dots)))$ を $(t_n, t_{n-1}, \dots, t_1)_{\mathcal{B}}$ と略記する。

N 個のレジスタに自然数 $1, 2, \dots, N$ を対応させ、レジスタ i のビット長を $\beta(i)$ と表す。 \mathcal{B} 上のレジスタ変数記号集合 \mathcal{R}_{bit} はソート bit の変数記号集合 $\mathcal{R}_{bit} = \{r_1^1, \dots, r_1^{\beta(1)}, \dots, r_N^1, \dots, r_N^{\beta(N)}\}$ である。代入 $c: \mathcal{R}_{bit} \rightarrow \mathcal{F}_{bit}(\mathcal{F}, \mathcal{R}_{bit})$ の有限集合を \mathcal{B} 上の制約集合と呼び、 $\mathcal{C}_{\mathcal{B}}$ のように表す。項 $(r_i^{\beta(i)}, r_i^{\beta(i)-1}, \dots, r_i^1)_{\mathcal{B}}$ を \tilde{r}_i , 代入 $\langle r_i^1 \leftarrow t_1, \dots, r_i^{\beta(i)} \leftarrow t_{\beta(i)} \rangle$ を $\tilde{r}_i \leftarrow (t_{\beta(i)}, \dots, t_1)_{\mathcal{B}}$ と略記する。 \mathcal{B} 上のアーキテクチャ仕様は、6 項組 $\langle \{bit, bitlist\}, \mathcal{F}, \mathcal{X}, E, \mathcal{R}_{bit}, \mathcal{C}_{\mathcal{B}} \rangle$ である。

ビット列表現されたレジスタに対する逐次化問題を、次のように定義する。

[定義 2] \mathcal{B} 上の仕様 $\langle \{bit, bitlist\}, \mathcal{F}, \mathcal{X}, E, \mathcal{R}_{bit}, \mathcal{C}_{\mathcal{B}} \rangle$ のもとで、与えられた項 $\bar{w} = [w_1, \dots, w_N]$ ($w_i \in \mathcal{F}(\mathcal{F}, \mathcal{R}_{bit})$; $i=1, \dots, N$) に対し

$$c_1(c_2(\dots c_n(\bar{r})\dots)) =_E \bar{w} \quad (\bar{r} = [\tilde{r}_1, \dots, \tilde{r}_N]) \quad (3)$$

を満たす列 $c_1, c_2, \dots, c_n \in \mathcal{C}_{\mathcal{B}}$ が存在するかどうかを決定する問題を \mathcal{B} 上の逐次化問題と言う。 \square

\mathcal{B} 上のデータパス代入の定義から知られるように、 \mathcal{B} 上の逐次化問題は、1 ビットレジスタを対象とした逐次化問題と等価である。従って容易に示されるように、 \mathcal{B} 上の逐次化問題も決定不能 (準決定可能) である。また逐次化手続きは、初期化で $\bar{r} = [r_1, \dots, r_N]$ とする代わりに $\bar{r} = [\tilde{r}_1, \dots, \tilde{r}_N]$ とすることにより \mathcal{B} 上の逐次化問題に適用できる。このとき定理 1 と同様の

[†] 基底ナローイングによる E 照合は、例えば $add(s(x), y) = s(add(x, y))$ のような典型的な自然数上の加算の記述に対しても停止しない。

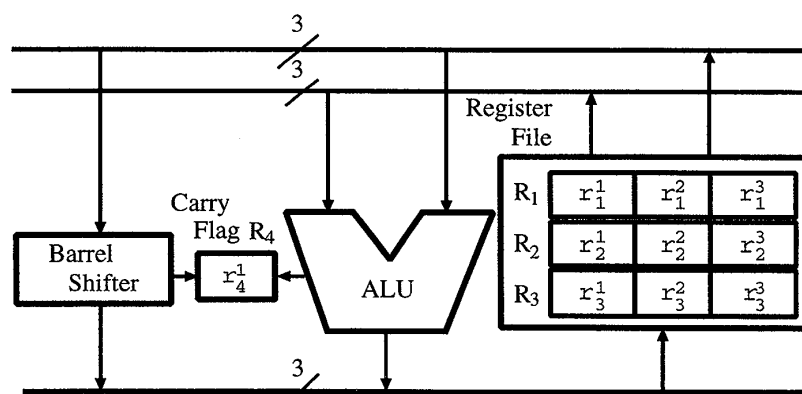


図 3 CPU のデータパス
Fig. 3 Datapath of a CPU.

性質が成り立つ。

4.2 CPU の仕様記述

本節では、前節の議論に基づき基本的な CPU のデータパスに対する仕様記述を与える。

図 3 に本例で扱う CPU のデータパスを示す。この CPU は ALU, バレルシフト, キャリーフラグのみをもつ 1 ビット長のフラグレジスタ R_4 , および三つの 3 ビット長レジスタ R_1, R_2, R_3 をもつレジスタファイルからなる。ALU は二つのレジスタの加算や論理演算が可能で、バレルシフトは左右への複数ビットシフトを行う。また、特に、適当な制御部が存在し、レジスタ R_1 の各ビットの値に従ったレジスタ選択や、レジスタ選択とシフトが 1 クロックで実行できるとする。

このデータパスの、 \mathcal{B} 上のハードウェアアーキテクチャ仕様を \mathcal{A} と表す。各レジスタのビット長は、それぞれ $\beta(1)=3, \beta(2)=3, \beta(3)=3, \beta(4)=1$ である。関数記号集合、レジスタ変数記号集合をそれぞれ

$$\mathcal{F}_{\mathcal{A}} = \{add, mult, sftl, sftl2, sftr, sftr2, msk, xor, and, or, not, d, 0, 1, nil\}$$

$$\mathcal{R}_{\mathcal{A}} = \{r_1^1, r_1^2, r_1^3, r_2^1, r_2^2, r_2^3, r_3^1, r_3^2, r_3^3, r_4^1\}$$

とする。各関数記号のランクを、

$$add : \langle bitlist, bitlist, bitlist \rangle$$

$$mult : \langle bitlist, bitlist, bitlist \rangle$$

$$sftl : \langle bitlist, bitlist \rangle$$

$$sftr : \langle bitlist, bitlist \rangle$$

$$fadd : \langle bit, bit, bitlist, bitlist \rangle$$

$$sftl2 : \langle bitlist, bitlist \rangle$$

$$sftr2 : \langle bitlist, bitlist \rangle$$

$$msk : \langle bit, bitlist, bitlist \rangle$$

$$xor : \langle bit, bit, bit \rangle$$

$$and : \langle bit, bit, bit \rangle$$

$$or : \langle bit, bit, bit \rangle$$

$$not : \langle bit, bit \rangle$$

とする。等式集合 $E_{\mathcal{A}}$ を表 1 に示す。B, C はソート *bit* の変数記号, X, Y はソート *bitlist* の変数記号である。 $E_{\mathcal{A}}$ はビット列の加算, 左右シフト, 乗算, 論理演算を定義する。 $E_{\mathcal{A}}$ の各等式を、左辺から右辺へ方向づけて得られる項書換え系を $R_{E_{\mathcal{A}}}$ とする。 $R_{E_{\mathcal{A}}}$ は次の性質をもつ。

[命題 4] $R_{E_{\mathcal{A}}}$ は完備である。

(略証) 停止性は辞書式経路順序⁽⁸⁾を用いて証明される。また、合流性は命題 1 を用いて示される。□

[命題 5] \mathcal{B} 上の逐次化問題について、逐次化手続き中で検査される $E_{\mathcal{A}}$ 照合可能性は、基底ナローイングを

表 1 等式集合 $E_{\mathcal{A}}$

$$add(d(B_1, X_1), d(B_2, X_2)) = d(xor(B_1, B_2), fadd(X_1, X_2, and(B_1, B_2))) \quad (1)$$

$$fadd(d(B_1, X_1), d(B_2, X_2), C) = d(xor(xor(B_1, B_2), C), fadd(X_1, X_2, or(and(B_1, B_2), and(or(B_1, B_2), C)))) \quad (2)$$

$$fadd(d(B_1, X_1), nil, C) = d(xor(B_1, C), fadd(X_1, nil, and(B_1, C))) \quad (3)$$

$$fadd(nil, d(B_1, X_1), C) = d(xor(B_1, C), fadd(nil, X_1, and(B_1, C))) \quad (4)$$

$$fadd(nil, nil, C) = nil \quad (5)$$

$$mult(d(B_1, X_1), d(B_2, X_2)) = add(msk(B_1, d(B_2, X_2)), mult(X_1, sftl(d(B_2, X_2)))) \quad (6)$$

$$mult(nil, X) = X \quad (7)$$

$$msk(B_1, d(B_2, X)) = d(and(B_1, B_2), msk(B_1, X)) \quad (8)$$

$$msk(B, nil) = nil \quad (9)$$

$$sftl(d(B, Y)) = d(0, sftl2(d(B, Y))) \quad (10)$$

$$sftl2(d(B_1, d(B_2, X))) = d(B_1, sftl2(d(B_2, X))) \quad (11)$$

$$sftl2(d(B, nil)) = nil \quad (12)$$

$$sftr(d(B_1, d(B_2, X))) = d(B_1, sftr2(X)) \quad (13)$$

$$sftr2(d(B, X)) = d(B, sftr2(X)) \quad (14)$$

$$sftr2(nil) = d(0, nil) \quad (15)$$

$$xor(B, 0) = B \quad (16)$$

$$xor(0, B) = B \quad (17)$$

$$and(B, 0) = 0 \quad (18)$$

$$and(0, B) = 0 \quad (19)$$

$$or(B, 0) = B \quad (20)$$

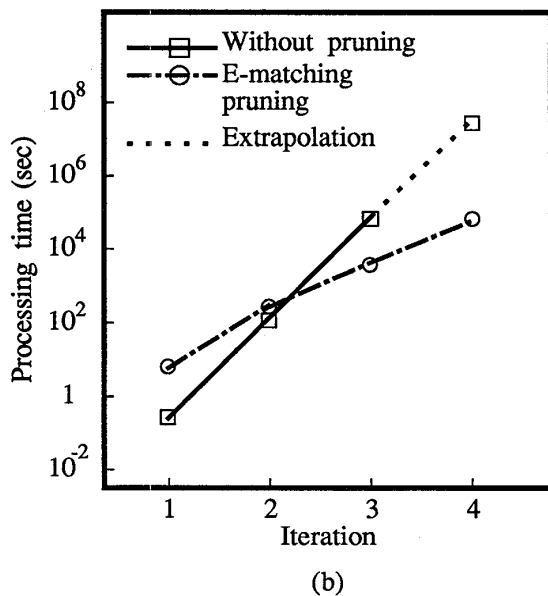
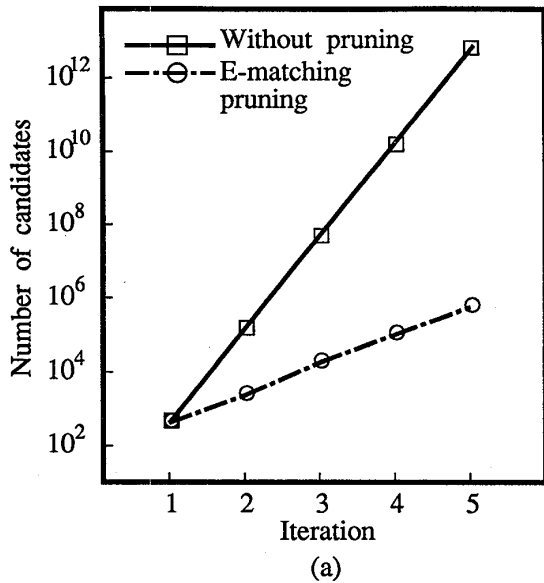
$$or(0, B) = B \quad (21)$$

$$not(not(B)) = B \quad (22)$$

用いた E 照合手続きにより決定可能である。

(略証) \mathcal{B} 上の制約集合の定義より、逐次化手続き中で生成される項列の各項は、

$[(t_i^{\beta(i)}, t_i^{\beta(i)-1}, \dots, t_i^1)_{\mathcal{B}}, \dots, (t_N^{\beta(N)}, t_N^{\beta(N)-1}, \dots, t_N^1)_{\mathcal{B}}]$
 $(t_i^j \in \mathcal{T}_{bit}(\mathcal{F}, \mathcal{R}_{bit}); i=1, \dots, N, j=1, \dots, \beta(i))$ の形をしている。また、各関数記号のアリティより、ソート *bit* の項の部分項もまたソート *bit* である。従って、逐次化手続き中で $R_{E_{\mathcal{A}}}$ により実行されるすべての基底ナローイングはソート *bit* の部分項から始まり、使用される書換え規則は(16)–(22)のみである。これらは右辺が変数記号、または構成子定数記号であるから、命題



(a) Number of generated candidates
(b) Processing time

図4 各繰返しにおける解候補の数と処理時間

Fig. 4 Number of generated candidates and processing time in each iteration of procedures.

3より逐次化手続き中の $E_{\mathcal{A}}$ 照合可能性は決定可能である。□

制約集合 $\mathcal{C}_{\mathcal{A}}$ は、このハードウェアで実行可能なマイクロ操作330種類に対応するデータパス代入からなる。その一部を次に示す。

$$\begin{aligned} c_1 &= \langle \tilde{r}_1 \leftarrow (ADD_3(1,2), ADD_2(1,2), ADD_1(1,2))_{\mathcal{S}}, \\ &\quad r_4 \leftarrow CAR(1,2) \rangle \\ c_2 &= \langle \tilde{r}_2 \leftarrow (ADD_3(2,3), ADD_2(2,3), ADD_1(2,3))_{\mathcal{S}}, \\ &\quad r_4 \leftarrow CAR(2,3) \rangle \\ c_3 &= \langle \tilde{r}_1 \leftarrow (and(r_1^3, r_1^3), and(r_1^1, r_1^2), and(r_1^1, r_1^1))_{\mathcal{S}} \rangle \end{aligned}$$

$$\begin{aligned} c_4 &= \langle \tilde{r}_2 \leftarrow (and(r_1^2, r_1^2), and(r_1^2, r_1^1), 0)_{\mathcal{S}}, \\ &\quad r_4 \leftarrow and(r_1^2, r_1^3) \rangle \end{aligned}$$

$$c_5 = \langle \tilde{r}_3 \leftarrow (and(r_1^3, r_1^1), 0, 0)_{\mathcal{S}}, r_4 \leftarrow and(r_1^3, r_1^2) \rangle$$

ここで、読みやすさのため

$$ADD_1(i, j) = xor(r_i^1, r_j^1)$$

$$ADD_2(i, j) = xor(xor(r_i^2, r_j^2), and(r_i^1, r_j^1))$$

$$ADD_3(i, j) = xor(xor(r_i^3, r_j^3), or(and(r_i^2, r_j^2), and(or(r_i^2, r_j^2), and(r_i^1, r_j^1))))$$

$$\begin{aligned} CAR(i, j) &= or(and(r_i^3, r_j^3), and(or(r_i^3, r_j^3), \\ &\quad or(and(r_i^2, r_j^2), and(or(r_i^2, r_j^2), \\ &\quad and(r_i^1, r_j^1)))))) \end{aligned}$$

とおいている。 c_1 は R_1 と R_2 を加算した数を R_1 に格納するマイクロ操作、 c_2 は R_2 と R_3 を加算した数を R_2 に格納するマイクロ操作を表す。また、 c_3, c_4, c_5 は、それぞれ R_1 の最下位ビット r_1^1 、第2ビット r_1^2 、最上位ビット r_1^3 の値について、1なら R_1 を各ビット位置まで左シフトした数を、0ならすべてのビットが0からなる数を、それぞれ R_1, R_2, R_3 へ格納するマイクロ操作を表す。

4.3 手続き実行例と枝刈りの効果

前節の仕様のもとで、レジスタ R_1 の値を2乗するマイクロ操作列を生成する問題を考える。

[問題] \mathcal{A} のもとで $\bar{w} = [w_1, w_2, w_3, w_4]$ は逐次化可能か。但し、

$$w_1 = mult(\tilde{r}_1, \tilde{r}_1)$$

$$w_2 = (xor(and(r_1^2, r_1^2), and(r_1^3, r_1^1)), and(r_1^2, r_1^1), 0)_{\mathcal{S}}$$

$$w_3 = (and(r_1^3, r_1^1), 0, 0)_{\mathcal{S}}$$

$$\begin{aligned} w_4 &= (or(and(and(r_1^1, r_1^3), xor(and(r_1^2, r_1^2))), \\ &\quad and(or(and(r_1^1, r_1^3), xor(and(r_1^2, r_1^2), \\ &\quad and(r_1^3, r_1^1))), and(and(r_1^1, r_1^2), \\ &\quad and(r_1^2, r_1^1))))_{\mathcal{S}} \end{aligned}$$

である[†]。□

この逐次化問題に対し、逐次化手続きは5ステップのマイクロ操作列 c_5, c_4, c_3, c_2, c_1 を出力して停止した。実行時間は55,568秒であった。

枝刈りの効果をみるため、枝刈りなしの手続きを上述の問題に適用し、実行中に生成された解候補の数と実行時間について逐次化手続きと比較する。逐次化手続きからステップ1(ii)を取り外し、枝刈りなしの手続きとする。

図4に、両者の手続きが各繰返しで生成した解候補

[†] 本来は w_1 を求めることにのみ関心があるが、手続きを実行するためには w_2, w_3, w_4 も指定しなければならない。この問題については必ずびで検討する。

の数, および処理に要した時間を示す。但し, 手続きは解を求めた時点で停止するので, 最後の繰返しの処理時間は示していない(図4(b))。また, 枝刈りなしの手続きは, 繰返し3回以上になると処理時間が急激に増加するため, 候補数, および処理に要した時間の測定ができなかった。枝刈りなしの手続きでは候補数は330(可能なマイクロ操作の数)を底とする指数関数で増加するので, 実測しなかった部分はこれに従って算出した値を図に示した。また, 処理時間については, 候補当りの実行時間から推測値を求めて示した。

図4(a)より, 枝刈りなしの手続きの場合, 候補数は330を底とする指数関数に従い増加するが, 逐次化手続きでは底がおよそ6に改善されていることが知られる。このとき, 各繰返して枝刈りされた候補数を調べたところ, 生成された候補数の97%以上であることが確かめられた。

また, 枝刈りなしの手続きはE照合可能性検査のコストがかからないため1候補当りの処理時間が短く, 繰返し2回までの処理時間は逐次化手続きより優位である。しかし, 繰返し3回以上になると枝刈りのコストより候補数の爆発の影響が大きくなり, 逆に逐次化手続きが優位になることが図4(b)より知られる。

5. む す び

Zhu, Johnsonが定式化した逐次化問題に対し, E同一化を利用して探索空間の枝刈りを行う準決定手続きを提案した。また, 算術演算の基本演算器(ALU, バレルシフタ)を備えたCPUのデータパスに対する仕様記述と逐次化問題の例を与え, 枝刈りの効果を調べた。この結果, E照合による枝刈りの効果は大きく, 枝刈りを行わない手続きに比べ, 本手続きは生成される候補数, および実行時間の点で極めて優位であることが知られた。

E同一化が決定可能な等式理論はいくつか知られているが⁽³⁾, ハードウェアの仕様記述で注目される理論にブール環がある⁽⁹⁾。ブール環はE同値性も決定可能であるから, 等式集合として逐次化手続きの利用条件を満たす。従って, 4.で述べた \mathcal{B} 上の仕様ブール環の公理を組み込むことにより, CPUなど実際のハードウェアに対する逐次化問題を扱うための有効な枠組みを与えることができると予想される。

ハードウェアの動作を書換えとしてとらえ, マイクロプログラムの自動合成問題を形式化する試みには文献(2)による以外にMahmoodら⁽¹⁰⁾がある。彼らは,

DT0Lを用いたデータパス動作の記述法のもとで同様な定式化を行い, ゴール項とE同値な項を順次生成し, これが逐次化可能であるか検査する準手続きを与えた。しかし, 演算公理の記述や, 探索空間の枝刈りに関する十分な議論はなされていない。

今後の課題として次の事柄がある。現在の枠組みではゴール項を与える際にすべてのレジスタに値を指定しなければならないため, 4.3の例題のように特定のレジスタに対するゴール値しか知られていない場合にはあらかじめ人手でマイクロプログラムを作成する方法により他のレジスタの値を求めなければならない。この場合, 逐次化手続きは人手で作成されたマイクロプログラムの検証や改良のためにしか利用できない。このような事態を避けるために, 値を指定しないレジスタを存在記号 \exists で修飾されたレジスタ変数で表すようにゴール項の定義を変更し, 逐次化問題を一般化することが考えられる。本論文の手続きを適当に改良することにより, この一般化された逐次化問題のための手続きを与えることができると思われる。

また, 任意の等式集合について決定可能な, 制限されたE同一化についての研究があるが⁽¹¹⁾, このようなE同一化が逐次化手続きに利用可能かどうか検討するべきである。更に, 実用上意義のある, 決定可能な逐次化問題の部分クラスを発見できれば本論文で提案した手続きの有効性が増すと思われる。

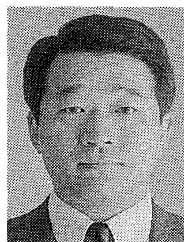
謝辞 日ごろ御指導頂く岐阜大学工学部後藤宗弘教授, 名古屋大学工学部稲垣康善教授に感謝致します。また, 本研究を進めるにあたり御討論頂いた豊橋技術科学大学VLSI設計研究室の諸兄, 鶴岡高専佐藤淳助手, ならびに三重大学工学部大山口通夫教授, 北陸先端科学技術大学院大学外山芳人教授, NTT基礎研究所小川瑞史氏, NTTコミュニケーション科学研究所真野健氏に感謝致します。また, 本論文に対し貴重な御意見を下さいました, 査読者の方々に感謝致します。

文 献

- (1) McFarland M. C., Parker A. C. and Camposano R.: "Tutorial on High-Level Synthesis", Proc. of the 25th Design Automation Conf. paper 23.1, pp. 330-336 (1988).
- (2) Zhu Z. and Johnson S. D.: "An Algebraic Characterization of Structural Synthesis for Hardware", Proc. of the IFIP Int. Workshop on Applied Formal Method for Correct VLSI Design, pp. 261-269, North-Holland (1989).
- (3) Hullot J. -M.: "Canonical Forms and Unification", Proc. of the 5th Conf. on Automated deduction, pp. 318-334 (1980).

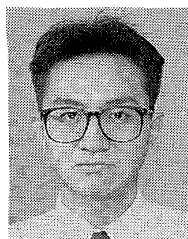
- (4) Fages F. and Huet G.: "Complete Set of Unifiers and Matchers in Equational Theories", Theoretical Computer Science, **43**, pp. 189-200 (1986).
- (5) Knuth D. E. and Bendix P. B.: "Simple Word Problems in Universal Algebras", Computational problems in abstract algebra, ed. J. Leech, Pergamon Press, Oxford, pp. 263-297 (1970).
- (6) Yamamoto A.: "Completeness of Extended Unification Based on Basic Narrowing", Proc. of the Logic Programming Conf., pp. 19-28 (1988).
- (7) Middeldorp A. and Hamoen E.: "Counter Examples to Completeness Results for Basic Narrowing", CWI report, CS-R9154 (1991).
- (8) Dershowitz N.: "Termination of Rewriting", J. Symbolic Computation, **3**, pp. 69-116 (1987).
- (9) Martin U. and Nipkow T.: "Boolean Unification—the story so far", J. Symbolic Computation, **7**, pp. 275-293 (1989).
- (10) Mahmood M., Mavaddat F., Elmasry M. I. and Cheng M. H. M.: "A Formal Language Model of Local Microcode Synthesis", Proc. of the IFIP Int. Workshop on Applied Formal Method for Correct VLSI Design, pp. 22-39, North-Holland (1989).
- (11) Gallier J., Narendran P., Raatz S. and Snyder W.: "Theorem Proving Using Equational Matings and Rigid E-Unification", J. ACM, **39**, 2, pp. 377-429 (1992).

(平成4年11月11日受付, 5月12日再受付)



今井 正治

昭49名大・工・電気工卒。昭54同大学院博士課程(情報工学専攻)了。工博。昭54より豊橋技術科学大学情報工学系勤務。現在、同助教授。昭59より60にかけて米国サウスカロライナ大学に文部省在外研究員(客員助教授)として滞在し教鞭をとる。平3より、日本電子機械工業会(EIAJ)VHDL小委員会仕様ワーキンググループ主査としてVHDL'92の標準化活動に従事。これまで、組合せ最適化問題、計算機アーキテクチャ、VLSIの設計自動化などの研究に従事。IEEE, ACM, 情報処理学会, 人工知能学会各会員。



木下 貴史

平1豊橋技科大・工・情報卒。平3同大学院博士前期課程了。現在、同大学院博士後期課程在学中。VLSIの設計自動化に関する研究に従事。情報処理学会会員。



直井 徹

昭58豊橋技科大・工・情報卒。昭60同大学院修士課程了。昭63名大大学院博士課程了。同大助手を経て、現在岐阜大・工・電子情報工学科助教授。工博。平1年度丹羽記念賞受賞。ソフトウェア基礎理論、特にプログラム意味論、項書換え系に関する研究に従事。