連言を否定する条件のための Rete アルゴリズムの拡張

高野 啓[†] 吉良賢治[†] 澤本 潤[†]

プロダクションシステムの高速実行手法である Rete アルゴリズムは，現在までに多くのプロダクションシステム処理系に実装されてきた．しかし，Rete は元来 OPS 言語を対象に開発されたため，その記述力は暗に OPS 言語の言語仕様から制約を受けているとわれわれは考える．したがって，Rete を実装する処理系では，ルールの条件部の言語仕様には OPS 言語に由来する構文上の制約を受けることになる．われわれは，Rete を採用した処理系では「単独のルールでは連言を否定する条件パタンを記述できない．」ことを確認し連言の否定条件を解釈するための機構を開発，Rete アルゴリズムの拡張として実装した．Rete には，これまでに各種の改造アルゴリズムが提案されてきたが，それら Rete の改良アルゴリズムは，その目的を Rete のさらなる高速化としており，記述力の強化を狙うものはなかった．本論文では，Rete アルゴリズムに上記の制約がある理由，およびわれわれが開発した Rete アルゴリズムの拡張としての連言の否定条件の解釈方法，およびその評価について述べる．

Extension of the Rete Algorithm for Interpreting Negated Conjunctional Condition Patterns

Akira Takano,[†] Kenji Kira[†] and Jun Sawamoto[†]

The Rete algorithm has been applied to many production system interpreters. On the other hand, Rete was originally developed for OPS language, and Rete's describability is implicitly restricted by OPS's language specification. Therefore Rete-based production system interpreters also inherit OPS's syntactic restrictions. We found that Rete-based PS interpreters can not describe negated conjunctional condition patterns. We extended the Rete algorithm to eliminate that restriction. So far, many improved algorithms based on the Rete algorithm have been proposed. But all of Rete-based algorithms appear to have been developed for improving efficiency, and no one seems to have extended the description power. This paper describes the reason why the Rete algorithm includes the restriction and our extension of the Rete algorithm for interpreting negated conjunctional condition patterns.

1. はじめに

Rete アルゴリズム[1]は，プロダクションシステム(以下 PS) の高速実行手法として，多くの PS 処理系に適用され，標準的アルゴリズムとしての地位を占めている．また，Rete を基盤として，さらに高速化するアルゴリズム[2),3)]，並列処理計算機への適用[4)]，Rete の問題点を明らかにし改善したアルゴリズム[5),6)] などが提案されている．これらの提案は，すべて Rete を上回る高速化を目的としている．

ところで，Rete は元来 OPS 言語に実装されたものであり，Rete で扱える文法は OPS 言語の制約を受けている[7)]が，これまでにこの問題に言及した報告はされていないようである．

本論文では，2章で Rete アルゴリズムが PS 処理系の文法に及ぼす制約を明らかにする．この「制約」とは，「単独のルール中では連言を否定する条件が書けない．」というものである．この制約は Rete を使用する処理系に特有のものであり，Rete を使用しない処理系で同様の問題があるとは限らない．よって，Rete を使用しない処理系では実現可能な文法が，Rete を使用する処理系で実現できない場合がある．また，連言の否定を記述するには，複数のルールで書き換える方法があるが，記述ルール数が増えるのでメンテナンス性，可読性が落ちると考える．したがって Rete を用いることで PS の記述力が制限されうるが，性能的には一般に Rete を用いることが有利である．

PS をプログラミング言語の一種とすれば，アプリケーションを作る側からは PS の標準的な言語仕様が

† 三菱電機(株)情報システム研究所
Computer & Information Systems Laboratory, Mitsubishi Electric Corp.

存在することが望ましく，「PS の標準仕様」を求める上で，実装手段から受ける制約をできるだけ明確にし，解消することが重要である．

3章では，上記の制約の解消方法を示す．その手法は，単独のルールで連言否定の記述を可能にするものである．その目的は，Rete ベースの処理系で扱える文法を拡張すること，その結果としてルールの可読性とメンテナンス性を向上することにある．なお，本論文中の用語およびルールの文法は OPS5[8] に準ずる．

OPS5 は，その実装内容が公開されており，他の商用の PS 処理系と比べて論じやすく，また PS に関する多くの論文で参照されており，この用語と文法に準ずることが適切である．

OPS5 自身は，実用されているケースはすでに少ないと思えるが，われわれは，現在でも Rete ベースの処理系では上述の問題は解決されていないと考える．たとえば，OPS83[9] は，OPS5 の後継といえる処理系であり，手続き処理の記述性や性能が向上したとされているが，否定条件に関する文法については，上述の問題が OPS5 と同様に存在する．

2. Rete アルゴリズムの概要

Rete アルゴリズムは，ルールの各条件からネットワークを構成し，条件中，同じ内容の部分を共通のノードとすることでパタンマッチを共通化し，また，パタンマッチの結果を保存して各実行サイクルに用いるようにすることで，PS の計算量を減少させたアルゴリズムである．

ここでは，Rete ネットワークの基本構成とアルゴリズムの基本的な動作を述べる．

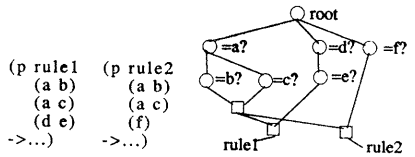
2.1 Rete ネットワークの基本的な構成

図1で，(1)の2つのルールから，(2)のような Rete ネットが構成される．

図1のルールの意味を簡単に述べる．ルール冒頭の 'p' は，ルールを示す指示子，'rule 1' 'rule 2' は，ルール名である．rule 1 の意味は，「作業記憶要素 '(a b)' '(a c)' '(d e)' が作業記憶に同時に存在すれば，rule 1 は真である．」ことを示す．

ルールの条件パタン中の1つの要素が，1つのノードに対応する．これを「1入力ノード」と呼ぶ．ここで，条件パタン中に共通の部分（rule 1 の中の 'a'）および，ルール間で共通の条件パタン（'(a b)' と '(a c)'）が共有される．

各パタンを表すアークの終端には，α メモリと呼ばれるノードがつく．各パタンは，図1(2)で□印のノードに1つずつ結合され，ここで複数パタンにまたがるパタンマッチを実施する．この種のノードを「2入力ノード」と呼ぶ．2入力ノードには，「β メモリ」と呼ばれるメモリノードがつく．



(1) ルール例 (2) Rete ネットの例

図1 ルールと Rete ネットの例
Fig. 1 An example of rules and Rete networks.

2.2 Rete アルゴリズムの基本的な動作

パタンマッチの対象となる作業記憶要素（WME: Working Memory Element）は，作業記憶に追加するときは＋トークンとして，削除されるときは－トークンとして，Rete ネットへ送られ，ルートノードからパタンマッチを開始する．

今，WME として '(a b)' を＋トークンとして図1(2)の Rete ネットに追加する．このとき，トークンはルート直下の各ノードでパタンマッチされ，'=a' のノードでのマッチだけ成功する．成功した場合だけトークンは次のノードへ送られる．以下，'=b' のノードのマッチに成功し，α メモリに格納され，2入力ノードに至る．このとき，2入力ノードの反対側のアークからもトークンが送られており，追加したトークンとのパタンマッチに成功すれば，2つのトークンを合成して β メモリに格納し，後続のノードへ送る．

このようにして，終端のノードに合成の＋トークンが到達すれば，あるルールが真であることになり，合成トークンとルール名を実行可能ルールの集合（「競合集合」と呼ばれる）に追加する．

逆にある WME が削除されるときは，その WME が－トークンとして Rete ネットに送られる．－トークンが到達したメモリノードからはトークンが削除され，パタンマッチに成功した2入力ノードでは，合成トークンを－トークンとして後続ノードに送る．以上の結果，終端ノードに－トークンが到達すると，あるルールを真とした WME の組合せが失われたことになり，ルールと WME の組合せを競合集合から削除する．

2.3 Rete アルゴリズムによる否定条件の解釈

Rete ネットワーク内で，ルール左辺中の否定条件

が結合する位置の2入力ノードは，同様の肯定条件が結合する2入力ノードとは別のノードとして作成される．否定条件が結合するノードは，肯定条件が結合する2入力ノードと動作が異なる．肯定条件にかかる2入力ノードの動作は，次のようである．

1）＋トークンの到達時には，トークンの進入と反対側のメモリ中の各トークンと照合を行い，成功したものとの組合せをβメモリに格納し，＋トークンとして次のノードへ流す．

2）－トークンの到達時は，トークンの進入と反対側のメモリ中の各トークンと照合を行い，成功したものは，その組合せをβメモリから除き，－トークンとして次のノードへ流す．

これに対し，否定条件にかかる2入力ノードの動作は，次のようになる．

1）左側（ルール中否定条件の上側）からの＋トークン到達時には，右側の各トークンと照合し，すべて失敗した場合，βメモリに格納して，次ノードへ流す．成功した組合せがある場合は，その数をトークンに記録する．

2）右側（否定条件を示すリンク）からの＋トークン到達時には，左側の各トークンと照合し，すべて失敗した場合には，何もしない．照合に成功した左側トークンで，他の右側トークンと照合に成功していなかったものは，これを－トークンとし，βメモリから除き，次ノードへ流す．成功していたものは，その数をインクリメントする．

3）左側からの－トークン到達時には，これがβメモリにあれば除き，次のノードへ送る．

4）右側からの－トークン到達時には，左側の各トークンとの照合において，成功したものの数をデクリメントする．その結果，0になった左側トークンは＋トークンとしてβメモリに格納し，次ノードへ送る．

先に述べたように，Rete では，条件パタンを1つずつネットワークに結合する．したがって，上記の個々の動作では，右側のトークンは常にルール中の単独のパタンであることが想定されている．これは，OPS5 の否定条件が単一パタンの否定しか扱わないことによることに由来していると考える．つまり Rete では，複数の条件パタンを一度に結合する枠組がなく，結果として，単独のルール中では，肯定条件の連言を否定するような条件記述ができない．

以上が，否定条件部分の Rete の動作概要と，そこに生じる制約である．

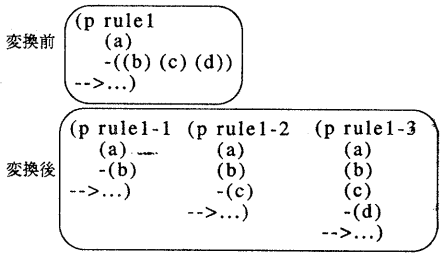


図 2　連言否定のためのルールの書換え
Fig. 2　Rewriting rules for a negated conjunction.

Rete アルゴリズムを使用する PS 処理系において連言の否定を扱うには，**図2**（以下，便宜的に連言否定は条件パタンを括弧内に入れた形式で示す）のように連言の否定条件を含むルールをいくつかのルールに書き換えて，動作を近似する方法がある．

この変換は，「元のルールの否定条件が偽になるのは，条件中のすべてのパタンが真であるときに限る．」ことを利用している．書換え後のルールは，元の否定条件中の各パタンを順に1つずつ追加し，それぞれ最後のものだけ否定条件にしたものになっている．この場合，書き換えたルール群全体では，'b' 'c' 'd' の各パタンすべてが真の場合にだけすべての書換えルールが偽になり，それ以外の場合は（'a' は常に真と仮定する．）元のルールは常に真，書換え後のルールはいずれか1つが真になる．よって，この方式により，元のルールの動作を置き換えることができる．

この方法の問題点は，ルール数が増えることにより，ルールベースのメンテナンス性が落ちること，また，ルールの書換え操作をパーザで行うとなれば，パーザの構造が複雑になること，などを挙げられる．

3. 連言否定条件の解釈方法とその評価

3.1 拡張方法

否定条件中に複数のパタンを記述した条件記述（これを「連言否定」と呼ぶ）を可能にするため，まず，連言否定の Rete 向けの解釈方法について考察し，続いて，われわれが採用した解釈方法に基づいて Rete アルゴリズムを拡張する方法を述べる．

われわれは，連言否定の解釈を，次のように行うことにした．ここで，連言否定に先行する条件群を「先行パタン」としてまとめ，連言否定内部の各パタンを「パタン1」～「パタン N」で表す．

連言否定全体が偽になるのは，パタン 1～パタン N

がすべて真のときだから，各パタンについて，偽であるパタン i を発見するまで，各パタンを評価するような枠組を Rete に追加し，その結果を後続ノードでのパタンマッチに用いる．この解釈にしたがって拡張した Rete ネットワークの概要を**図3**に示す．以下，図3のネットワークの動作を述べる．図3で，太線のリンクは，従来の2入力ノード間のリンク（「通常リンク」と呼ぶ）とは別にここで新たに追加するものである．これを「否定リンク」と呼ぶ．

否定リンクの導入にともない，2入力ノードを次の3種類に分類する．

1）否定開始ノード
「通常リンク」「否定リンク」の両方を持つ2入力ノードで，連言否定が始まる位置を示す．

2）否定終了ノード
「否定リンク」だけを持つ2入力ノードで，連言否定が終わる位置を示す．

3）通常ノード
「通常リンク」だけを持つ2入力ノードである．連言否定の内部に置く．

上記のうち，「通常ノード」の動作は，従来の2入力ノードと同様である．以下，「否定開始ノード」「否定終了ノード」の動作を解説する．

1）否定開始ノードの動作

ⅰ．＋トークンが，左側（先行パタン）から来たとき
＋トークンに ID 番号を付加し，後続ノードへ合成トークンを送る．また右側の各トークンとのパタンマッチを行い，成功した組合せの合成トークンを否定リンク側のノードへ送る．否定リンク側に続くノードでは，通常の2入力ノードと同様の動作になり，パタンマッチに成功する限り，前述の論理式のように偽となるノードに到達するまで評価する．

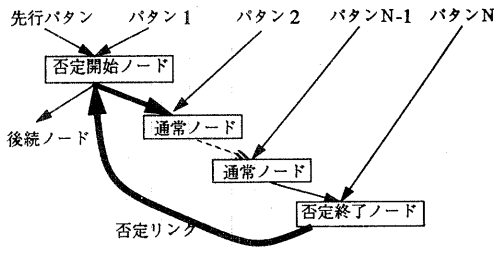


図3 拡張した Rete ネット表現
Fig. 3 An outline of the extended Rete network.

ⅱ．＋トークンが，右側から来たとき
左側の各トークンとのパタンマッチで，成功したときは合成トークンを否定リンクに送り，前述と同様連言否定内部でのパタンマッチを行う．左側にあったトークンは，すでに後続ノードへ送られている．

ⅲ．－トークンが，左側から来たとき
引続き後続ノードへ－トークンを送るとともに，否定リンク側に合成トークンがある場合には，引続き否定リンク側にも－トークンを送る．

ⅳ．－トークンが，右側から来たとき
(ⅲ)と同様，否定リンク側に－トークンを送る．

2）否定終了ノード

ⅰ．＋トークンが来たとき
否定終了ノードにおいて，左右どちらかの方向から＋トークンが来て，パタンマッチに成功したときは，連言否定内のすべての条件項が真になったことを意味する．このとき，パタンマッチ対象になっている左側トークンは，前述の1)ⅲ)で付加した ID 番号を持つ．この ID 番号をキーとして，否定リンクを経由して否定開始ノード中にある同一 ID 番号のトークンを－トークンとして後続ノードへ送る．これにより，連言否定が真から偽に転じたことを表す．なお，否定終了ノードでは，従来の Rete と同様，左側の各トークンに対しパタンマッチに成功している右側トークンの数を記録する．この数が正のときに右側からのトークンとのパタンマッチに成功したときは，－トークンは送らず，単にマッチした数をインクリメントする．

ⅱ．－トークンが左側から来たとき
β メモリ中にこれに該当する合成トークンがあれば，これを削除する．さらに，否定リンクが示す否定開始ノードに，この－トークンと同じ ID 番号を持つトークンがある場合は，－トークンは連言否定内部で生じた（連言否定中のどこかのパタンが偽になり，したがって全体が真に転じた）ことを意味する．このときは，その－トークンと同じ ID 番号を持つトークンを否定リンクが示す否定開始ノードから後続ノードへ送る．これにより，ルール中の連言否定以降のパタンとのマッチを行う．

ⅲ．－トークンが右側から来たとき

その－トークンにマッチしていた左側トークンがあり，その左側トークンにマッチしていた右側トークンは他にない場合は，連言否定が偽から真に転じたことを意味する．このときは，その左側トークンと同じ ID 番号を持つトークンを，ⅱ）と同様に対応する否定開始ノードの後続ノードへ送る．

以上のようにして，Rete アルゴリズムの拡張として連言否定の解釈実行を行う．

3.1 節で示した拡張方法において，後続ノードに合成トークンが送られるときは，連言否定中の少なくとも1つのパタンが偽のときであり，後続ノードに合成トークンが送られることで連言否定条件の評価が真であることが示される．逆に，すべてのパタンが真のときは，後続ノードへ合成トークンは送付されないか，すでに送られているときは削除され，連言否定条件の評価値が偽であることが示される．

3.2 考　　察

ここでは，前述の方法を採用した理由を述べる．3.1 節以外の連言否定の解釈としては，連言否定の部分を副木として解釈し，その結果を先行パタンとのパタンマッチに用いる（**図 4**）Rete ネットを構築することが考えられる．しかし，この方法は，3.1 節と比較して2つの問題点がある．以下に述べる．

1）構造が複雑になる

元来の Rete ネットでは，右側のノードは常に α メモリにつながり，3.1 節でも同様である．しかし，副木構造の場合は，右側にも β メモリが出現する．このことは，2入力ノードテストにおいて，右側トークンも合成トークンになることを想定した拡張を必要とする．われわれは，このためのインタプリタとパーザの改造は，3.1 節の方式によるよりも難しいと考えた．

2）変数の制約を表現しにくい

連言否定全体と先行パタンのパタンマッチを行う際，先行パタン中の変数の制約を連言否定内部に伝えにくい．例えば，次のルールでは，連言否定中の手続き的条件が，先行パタンの変数〈x〉を参照するが，副木構造ではこの変数情報を得られない．

```
(p rule 1
  (a ^a 1 〈x〉 ^a 2 〈y〉)
  －((b ^b 1 〈y〉 〈z〉) (><x><z>))
…>右辺)
```

この点，3.1 節では，連言否定の各パタンは逐次先行パタンに連結され，変数の制約を伝えられる．

3.3 評　　価

本方式により，ルールの書換えは不要になり，ルールのメンテナンス性は向上する．一例を図2のルールの連言否定内部の条件パタンを変更する場合について示す．

条件パタン‘(b)’を‘(b′)’のように変更するとき，本方式では図の「変換前」のルールを1つだけ変更するが，書換え式による場合は「変換後」すべてのルールを書き換える必要がある．

次に，本方式と，連言否定を含むルールを複数のルールで書き換える方式との性能比較を行う．

左辺途中に N 個の条件パタンからなる連言否定を持ち，それに続く P 個の条件パタンを持つ次のようなルールを，両方式で表現した場合について考える．ここで $N=3$，$P=3$ とした場合のルール書換え式による Rete ネットを**図 5** に示す．

```
(p rule 1
  (先行パタン)
  －((パタン1)…(パタン N))
  (後続パタン1)…(後続パタンP)
 －>右辺)
```

ルール書換え方式では，図5のように，否定条件中のパタンの数だけノードの分岐が起こり，書換えルール1つにつき $1+P$ 個のノードが増え，一般に（$P+$

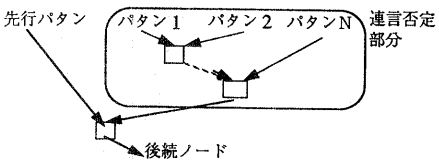


図 4　連言否定を副木にした Rete ネット表現
Fig. 4　Using sub-tree for a negated conjunction.

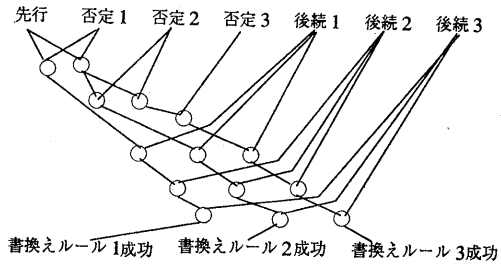


図 5　ルール書換え式による Rete ネット
Fig. 5　A Rete network of rewrited rules.

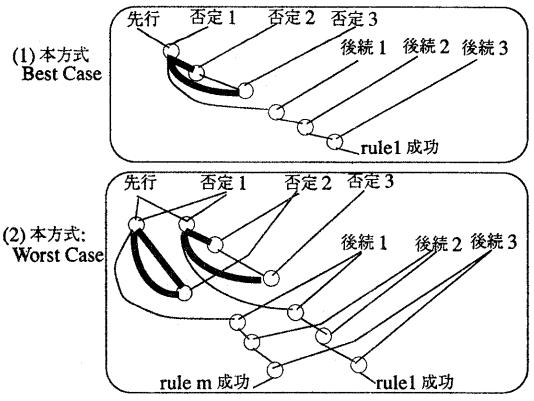


図 6 本方式による Rete ネット
Fig. 6 Extended Rete networks.

表 1 ノード数比較
Table 1 Node size comparison.

	ルール書換え	本方式
Best	θ (NP)	θ (N+P)
Worst	θ (NP)	θ ($N^2/2$+NP)

2) $N-1$ 個のノードが必要になる.

同じく, $N=3$, $P=3$ について本方式での Rete ネットを**図 6** に示す. 本方式では, 図 6 (1) のようにノード数は $N+P$ 個のノードで済む. したがって, この場合, 本方式によるノード数は書換え式よりも少なく, これにともなって速度向上も期待できる.

ただし, ここで評価したルールと連言否定部分以外の構成が同じで, 連言否定内部は m ($m<N$) 番目までの条件を持つようなルールがあるとき, 本方式では, 連言否定内部のノード共有を行わず, 図 6 (2) ($m=2$ とした) のように, 連言否定の開始ノードからノードが分岐する. 一方, 書換え方式では, もともとそのようなルールを書換えルールに含むので, ノードは完全に共有される. このような場合, 本方式が不利となる. 以上から, ノード数比較を**表 1** にまとめる.

性能的には, ルールの書換えを行う場合との優劣はアプリケーションに依存するが, 先に述べたように, 本方式ではメンテナンス性, 可読性の向上を期待でき, また本論文で述べた制約をもたない PS 処理系との文法的な同等性も実現できる. これらの点において, 本方式の利点がある.

4. おわりに

否定条件の表現に関して Rete アルゴリズムが PS 処理系に及ぼす文法上の制約を述べ, それらの制約を解消する方法を提案した. 本方式により, Rete を使用する PS 処理系において, 連言否定を扱う際のルールの書換えは不要になる. 本方式には, ルールの書換えによる実行と比べて, 3.3 節に示した性能特性がある.

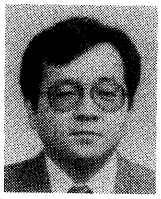
Rete 自体は, PS 処理系の実現手段として広く用いられてきたが, その文法的な特性を論じた報告はこれまで少なかったように思われる. 現在でも, PS はエキスパートシステム構築のための主要な手段である. PS 処理系に関する研究では, 従来はその速度性能を改良することに焦点が集中していた. 今後は, システム構築言語としての記述力を強化することも主要な課題であるといえる.

謝辞 Rete についてご指導いただいた福岡工業大学教授荒屋眞二先生, 本案の実現にご協力いただいた米国 Horizon Research Inc. の諸氏, 本研究に貴重な助言を頂いた ICOT 太田好彦氏に深く感謝いたします.

参考文献

1) Forgy, C. L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*, *Artificial Intelligence*, 19, pp. 17-37 (1982).
2) 荒屋眞二ほか: プロダクションシステムにおける効率的パタン照合のための連想 Rete ネットワーク表現, 情報処理学会論文誌, Vol. 29, No. 8, pp. 741-748 (1988).
3) 田野俊一ほか: ST-NET アルゴリズム: 双方向推論の高速処理方式, 情報処理学会論文誌, Vol. 29, No. 10, pp. 944-953 (1988).
4) Gupta, A.: Results of Parallel Implementation of OPS 5 on the Encore Multiprocessor, CMU-CS-87-146, Carnegie-Mellon Univ. (1987).
5) Miranker, D. P.: TREAT: A Better Match Algorithm for AI Production Systems, *AAAI '87*, pp. 42-47 (1987).
6) Miranker, D. P. et al.: On the Performance of Lazy Matching in Production Systems, *AAAI '90*, pp. 685-692 (1990).
7) 高野 啓ほか: 変数表現と否定条件のための Rete アルゴリズムの拡張, 情報処理学会研究会報告, 84-AI-1, pp. 1-9 (1992).
8) Forgy, C. L.: OPS 5 User's Manual, CMU-CS-81-135, Carnegie-Mellon Univ. (1981).
9) Forgy, C. L.: 人工知能用言語 OPS 83, p. 354, パーソナルメディア, 東京 (1986).

(平成 4 年 10 月 14 日受付)
(平成 5 年 12 月 9 日採録)

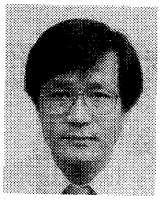


高野 啓（正会員）

昭和 63 年東京農工大学大学院工学研究科修士課程修了．同年三菱電機(株)入社．現在，同社情報システム研究所にて，ソフトウェア構築技術の研究開発に従事．日本ソフトウェア科学会会員．

吉良 賢治（正会員）

昭和 59 年京都大学理学部数学系卒業．同年三菱電機(株)入社．現在，同社情報システム研究所にて，知識ベース管理，データベース応用の研究開発に従事．平成 4 年イリノイ大学計算機科学科修士課程修了．日本ソフトウェア科学会，人工知能学会，AAAI 各会員．



澤本 潤（正会員）

昭和 50 年京都大学大学院工学研究科修士課程修了．同年三菱電機(株)入社．現在，同社情報システム研究所にて，システム構築技術の研究開発に従事．昭和 59 年スタンフォード大学修士課程修了．人工知能学会会員．