人工知能 72－6
（1990．9．12）

オブジェクト指向推論方式

濱口　敏英　　渡守武　和記
松本　典子　　西山　保
松下電器産業（株）　半導体研究センター

推論機構における照合処理高速化手法の中でも特にＲｅｔｅアルゴリズムは有名である。しかしながらデータ量の増加に対して処理速度が極端に遅くなってしまう。本稿では、データ量の増加に対して効率のよい照合処理を行うために、Ｒｅｔｅアルゴリズムにオブジェクト指向を取り入れた照合方法を提案する。具体的には、従来どうりＲｅｔｅネット上にルールの部分的な照合結果を記憶しておくが、データオブジェクトに照合処理メソッドを持たせることでデータオブジェクト自身が照合処理を行なう。また、論理合成システムの回路変換処理にこの方式を適用しその効果についても報告する。

An Object Oriented Inference Method

Toshihide HAMAGUTI　Kazunori TOMOTAKE
Noriko MATSUMOTO　Tamotsu NISHIYAMA
Semiconductor Research Center, Matsushita Electric Industrial
3-15, Yagumo-nakamachi, Moriguchi, Osaka, 570, Japan

The Rete match algorithm is especially popular among many high-speed pattern matching algorithms in the inference mechanism. However, the increasing of data extremely lowers its speed. In this paper, we propose an object oriented pattern matching method in Rete match algorithm, so that an efficient pattern matching is achieved under the increasing of data. Its main feature is that the data objects on the Rete net have their own matching methods. We also report the result of application of it to the circuit transforming part in the logic synthesis system.

1. はじめに

最近、様々な分野でエキスパートシステムが開発されている。エキスパートシステムで扱う知識の表現方法として、事象の追加・変更などの動作を表す動的な知識をルールで表現し、事象の定義・内容を表す静的な知識をフレームやオブジェクト指向で表現する方法が主流となってきている。

そこで、オブジェクト指向の概念を導入して推論機構を作ることにより、ルールとフレームとを統一的に組み込んだ推論システムを容易に実現しようと試みられている。例えば、以下に示す推論システムはオブジェクト指向で実現されている。Ｏｐｕｓ［１］は、プロダクションシステムＯＰＳ５と同等な機能をオブジェクト指向言語Smalltalk-80環境下で実現した推論システムである。ＡＵＫ［２］は、知識オブジェクトに統一な外部インターフェースを持たせることにより知識のモジュール性・統合性を高め、さらにオブジェクト指向の特徴である継承利用により拡張を容易にした知的システム構築用シェルである。

しかしながら、これらのシステムでは静的な知識である対象モデルの構造や属性のみを表すためだけにオブジェクト指向を用いている。そのため、推論機構がルールを起動して推論を進めていく上での属性値の参照の負担が大きい。そこで、推論機構の負担を軽減し推論処理を高速化する方法が必要となる。

そのためここでは、単に対象モデルの構造や属性だけを表すためにオブジェクト指向を用いるのではなく、対象モデル自身に効率のよい照合処理のメソッドを持たせることによって、推論機構の負担を軽減させ処理の高速化をはかる方法を提案する。また、そのために必要なクラス構成について述べる。

本稿では、まず２章でReteアルゴリズム［３］とその問題点について述べる。３章でオブジェクト指向の推論方式の特徴について述べる。４章では、推論機構の構成と処理内容について述べる。５章では、以上の考えに基づいてシステムの試作を行ない論理合成システムへ適用したことについて報告する。

2. 解決する課題

推論機構の照合処理高速化手法の１つとして特にReteアルゴリズムが有名である。Reteアルゴリズムは以下の２点の特徴を持つ。

（１）１回のルールの実行においてＷＭのわずかな部分のみが変化することに着目して、前回までの照合の状態を保持し、変化したワーキングメモリだけに照合処理を限定することによって効率化をはかる。

（２）ルール条件部の共通部分の照合を１度に行ない、同一の照合処理を避けることによって効率化をはかる。

具体的には、１データ内の属性値の一致や比較のための１入力１出力の１入力ノードと、データ間での属性値の一致や比較のための２入力１出力の２入力ノードと、条件がすべて成立したデータの組を候補に登録するterminalノードと、データをはじめに流すrootノードとの４種類のノードを用いて全ルールをネットワーク状に展開する。そして、rootノードにデータを流すことにより照合処理が進む。

また、Reteアルゴリズムの改良による高速化の研究も行われている。例えば、条件間の排他性を考慮してネットワークを展開することによって必ず失敗する照合を避ける方法［４］や、実行中における各々のノードでの統計データを基にして２入力テストの順番や組合せを動的に変更することによって２入力ノードでの計算量を削減する方法［５］などが研究されている。

処理速度は、ルール数の増加に対しては、以上の研究などにより効率のよい順番で照合処理を行うようにネットワークを展開することができるため、それほど低下しない。

しかしながらデータ量の増加に対しては、データ量の増加にともなって各ノードに記憶される照合結果も増加するために低下する。通常、２入力ノードでの処理に要する時間はデータ量の２乗に比例して増加する。特に１入力ノードの制限が

弱く２入力ノードに到達するデータが多い場合には全体の処理速度がかなり低下してしまう。

また別の知識表現を用いた照合処理方法として、動的な知識をルールで表し静的な知識をフレームで表すのではなく、オブジェクトそれ自身に静的な知識と動的な知識とを持たせ、オブジェクトがメッセージを交換することにより推論を進めていくことも可能である。しかし、この方法ではReteアルゴリズムを活用できない。

3. オブジェクト指向推論方式の特徴

今回提案するオブジェクト指向の推論方法を説明する。本推論方法はReteアルゴリズムの照合処理にオブジェクト指向を取り入れることによってさらに高速化を行なう方法である。

本方式の特徴を以下に示す。

（１）知識をオブジェクト指向で表現することによって知識のモジュール性がよい。

（２）継承の利用によってシステムの拡張が容易である。

（３）静的な知識にはデータの属性値参照以外に推論の照合処理の手続きも備えている。

（４）全ての処理をメッセージセンディングで行う。

特に本手法の特徴である（３）について以下で説明する。動的な知識であるルールと静的な知識であるデータの両方をオブジェクトとみなし、データオブジェクトが照合処理を行ない、ルールの集まりであるルールセットオブジェクトが照合に成功しているルールとデータオブジェクト組とを記憶する。

具体的には、ルールセットオブジェクトはReteアルゴリズムに従って以下の処理を行なう。ルールの実行によって変化したデータをメッセージで受け取り、ネット上の部分的に照合に成功したデータの組合せのメモリの修正を行いつつ、実行可能なルールの候補の変化分を求める。その際照合処理は、ルールセットオブジェクトからデータオブジェクトにメッセージを送ることで行われる。つまり，データのある属性の一致や比較などの条件を満足するかのテストや、データの指定した属性の値と一致する属性値を持つデータオブジェクトを検索するなどのメッセージが送られる。

また、データオブジェクトの照合処理メソッドは高速実行を実現しているが、それについては4.3節で説明する。

4. 推論機構

4. 1 クラス構成

今回提案する推論システムのクラス構成を図１に示す。推論クラスとデータクラスに分かれる。

推論クラスは推論方法、競合解消戦略、推論の途中結果や候補の記憶などを記述したクラスである。主なクラスの概要は以下の通りである。

・Candidate（候補クラス）
　推論の照合に成功した情報（ルールとデータの組等）を記憶するクラス
・InferenceStertegy（推論制御クラス）
　推論方法を記述したクラス
・Candidates（候補集合クラス）
　候補インスンタンスを要素に持ち、かつ競合解消戦略を記述したクラス
・RuleSet（ルールセットクラス）
　メッセージで送られてきたデータによって実行可能となる候補を求めるクラス

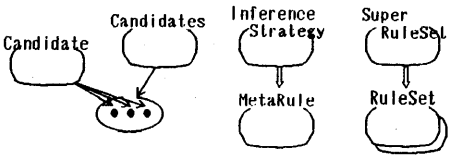
他に、RuleSetクラスのスーパークラスであるSuperRuleSetクラスや、どのRuleSetクラスをどの推論方法で処理するか、およびその順序、競合解消方法等の戦略を記述したMetaRuleクラスなどがある。

データクラスはデータを表すためのクラスである。静的な知識とそれに関連したメソッドと照合処理メソッドとを持つ。また、階層化により静的な知識とメソッドの継承を有効に使っている。

4. 2 推論の流れ

図２を用いて主要なクラス間のメッセージの流れを説明する。推論制御クラスは以下の処理手順が記述されている。　変化したデータをルールセットクラスに知らせる［データの登録］メッセージを送り、照合処理が行われる。　候補集合クラ

（1）推論クラスとルールセットクラス



（2）データクラス

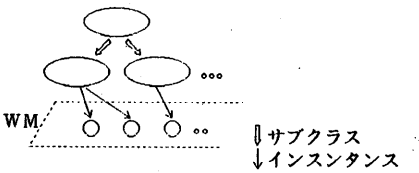


図1　クラス構成

スに［候補の選択］メッセージを送り、競合解消が行われる。　選択された候補に［ルールの発火］メッセージを送り、ルールの実行が行なわれる。また、ルールセットクラスは送られてきた変化データから実行可能な候補集合の変化を求め、候補集合クラスの更新を行なうために［候補ルールの登録］メッセージを候補集合に送る。候補集合クラスは候補選択メッセージに対して、競合解消戦略に従って候補を選択する。

以上示した様に各クラスの機能が明確に分離されており、それらのクラス間をメッセージが流れることでワーキングメモリの更新を行っていく。

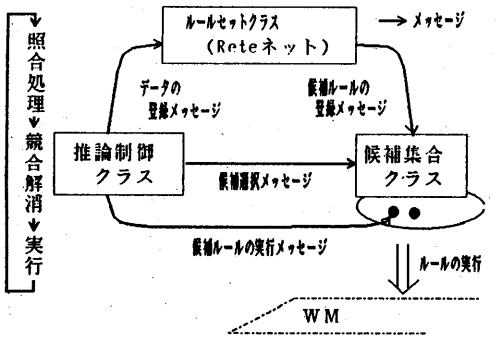
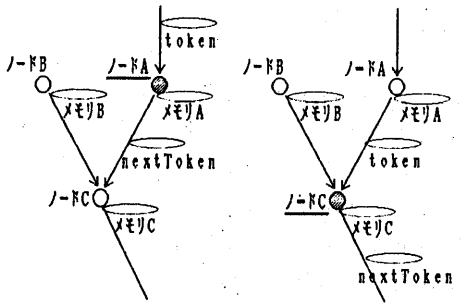


図2　メッセージの流れ

4．3　照合処理

Reteネットワークは先に述べた4種類のノードから構成される。照合処理はその内の1入力ノードと2入力ノードとで行われる。以下では、この2種類のノードでの従来方法の処理手順と提案する方法の処理手順とについて述べた後、その違いについて説明する。

まず、従来の追加、消去の照合手順を図3を用いて説明する。



（1,2）1入力ノード　（3,4）2入力ノード

図3　Reteネット上のtokenの流れ

・1入力ノードでの追加処理

step1 流れてきた＋token（プラストークン）中の指定位置のデータを取り出す。

step2 取り出したデータのある属性値がこのノードでの条件を満足しているかを調べる。

step3 もし満足しているならば、次のノードに＋tokenを流して処理を進めていく。
もし満足していなければ、終了する。

・1入力ノードでの消去処理

－token（マイナストークン）が流れる以外は追加と同じ処理である。

・2入力ノードでの追加処理

step1 流れてきた＋token側のメモリAに流れてきたtokenを追加する。

step2 流れてきた＋token中の指定位置のデータと流れてこなかったノード側の途中結果メモリB中の指定位置のデータとの関係がこのノードでの条件を満足しているかを調べる。

step3 もし満足していれば、新しいnextTokenを合成し、さらに次のノードに＋nextTokenを流して処理を進めていく。

もし満足していなければ、終了する。

・2入力ノードでの消去処理

step1 流れてきた－token側のメモリAから流れてきたtokenを消去する。

step2,step3

－tokenが流れる以外は追加と同じ処理である。

次に、今回提案するオブジェクト指向の照合手順を次に示す。

・1入力ノードでの追加、消去処理

step1で途中メモリへの追加、消去処理を行なうことと、step2でデータオブジェクトにメッセージを送って照合に成功したかを調べることとが違う以外は、従来の処理と同じである。

・2入力ノードでの追加処理

step1 流れてきた＋token中の指定したデータオブジェクトを取り出す。

step2 取り出したデータオブジェクトに対し、このノードでの条件を満足するデータを探せとメッセージを送り、条件を満たすデータを求める。求めたデータが、反対側の途中結果メモリB中の指定位置にあるかを調る。

step3 もし存在すれば、照合が成功したとして新しくnextTokenを生成し、nextTokenを途中結果メモリCに追加し、さらに次のノードに＋nextTokenを流して処理を進めていく。

もし存在しなければ、終了する。

・2入力ノードでの消去処理

step1 流れてきた－token中の指定したデータオブジェクトを取り出す。

step2 取り出したデータオブジェクトが途中結果メモリCの指定位置に存在するか調べる。

step3 もし存在すれば、メモリCから消去し、さらに次のノードにtokenを流して処理を進めていく。

もし存在しなければ、終了する。

以上から、従来の照合処理との違いは、従来方法ではデータは単に属性値を参照されるだけであるが、今回の提案する方法ではデータをオブジェクトとみなし、条件を満たすデータオブジェクトを検索する機能をもたせており、Reteネット上には照合に成功した組を記憶しているところに特徴がある。

効率よく照合処理を行えるかは、データオブジェクトが条件を満たすデータをいかに早く検索し、また、検索で求まったデータが途中結果メモリ中の指定位置に記憶されているかをいかに早く検索するかによる。条件を満たすデータの検索を高速化するために属性値のインスンタンスを作り、各データインスンタンスに属性値を持たせるのではなく、属性値インスンタンスとポインタ接続させておくことで、同一の属性値を持つデータインスンタンスを速く検索できるようする。また、途中結果メモリにデータオブジェクトが含まれているかはハッシュテーブルを用いて高速化をはかる。

4．4　推論戦略

推論方法や競合解消戦略はメソッドの継承や抽象クラスを使って実現している。

具体的に推論方法については、すべてのRuleSetクラスのスーパークラスであるSuperRuleSetクラスと、抽象化クラスであるCandidatesクラスを用いてInferenceStertegyクラスに推論方法を記述することによって、RuleSetや競合解消方法に独立な形で推論方法を実現できる。競合解消戦略については、RuleSetクラスや推論制御クラスとは独立な形でCandidatesクラスに競合解消戦略が書かれているので、Candidatesクラスの候補ルール登録メソッドの処理内容だけを変更したサブクラスを作るだけで異なった競合解消戦略を容易に実現できる。

5．試作システム

以上の考えに基づいて、オブジェクト指向言語Ｋｏｏｌａ［６］を用いて論理合成エキスパートシステム［７］の回路変換・最適化処理［８－９］用に推論システムを試作した。回路変換・最適化とは、ある回路パターンを最適な回路パターンに変換したり、あるいは使用可能なライブラリのセルへ割り付けるものである。

4. 1 論理回路表現

図4に論理回路の表現と必要なクラス関係を示す。論理回路中の各ゲートはインスンタンスで表され、それらは属性値にあたるネットインスタンスを経由してポインタ接続で回路を実現している。また、クラスを階層化することで、インスンタンス変数、メソッドの継承を有効に用いている。例えば、照合を満足するゲートの検索、接続の修正やfanoutの計算などをするための共通なメソッドは上位に定義している。

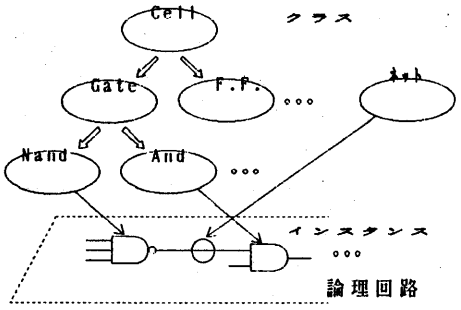


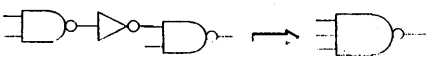
図4 論理回路の表現方法

5. 2 回路変換ルール記述

回路変換ルールの記述例を図5に示す。1行目では、ルールセット名を表しており、この名前のクラスが生成される。2行目は、ルール名を表している。3行目の#priorityは優先度を表すキーワードである。次の行に、このルールより優先度の低いルール名を記述する。4行目の#connectionは前提条件を表すキーワードである。回路パターンをフレーム形式で記述する。Y1, Y2などの一致が接続条件を表している。14行目の#conditionは制約条件を表すキーワードである。ゲートの出力の枝別れが1本であるかなどの接続以外の条件を記述する。17行目の#conclusionは結論部を表すキーワードである。回路最適化後の回路をフレーム形式と式とを混在して記述する。

5. 3 ルールセットクラスの構造

図6は図5のルール記述の#connectionにおけるデータの接続関係（回路パターン）を解析して生成されるクラス定義のReteネット部分を図式化し



```
RuleSet ReduceRule
| nandinverternand |
    #priority
    #connection
        gate1 a_kind_of: nand
              input    : [ X1IL1 ]
              output   : Y1 ;
        gate2 a_kind_of: inverter
              input    : [ Y1IL2 ]
              output   : Y2 ;
        gate3 a_kind_of: nand
              input    : [ Y2IL3 ]
              output   : Y3 ;
    #condition
        [ gate1 fanput ] == 1 ;
        [ gate2 fanout ] == 1 ;
    #conclusion
        [ IX4 ] = ( [ X1IL1 ] + [ IL3 ] ) ;
        gate4 a_kind_of: nand
              input    : [ IX4 ]
              output   : Y3 ;
```

図5 回路変換ルール記述例

たものである。各ノードはメソッドとして実現されており、その処理について説明する。

矢印は次のノードへのメッセージの流れを示している。各ノードはルールセットクラスのメソッドに変換されており、その処理内容は照合メッセージや次のノードへのメッセージ、途中結果修正メッセージなどからなる。具体的には、rootノードでは、受け取ったデータからtokenを生成し、次のノードに生成したtokenをメッセージで送る。1入力ノードであるinverterノードでは送られてきたtokenがinvertereであるかの照合を行い、成功したtokenを次のノードにメッセージで送り、途中結果メモリinvetereCellsにtokenを加える。2入力ノードであるnand_inverterノードでは、もし左からtokenが流れてきた時、流れてきたtokenと右のメモリnandCellsとがこのノードでのテストである接続を満足しているかを調べ、成功したtokenを次のノードにメッセージで送り、途中結果メモリnandinverterCellsにtokenを加える。terminalノードでは、candidatesにここまでの照合に成功したtokenを送り、候補集合の更新を行う。

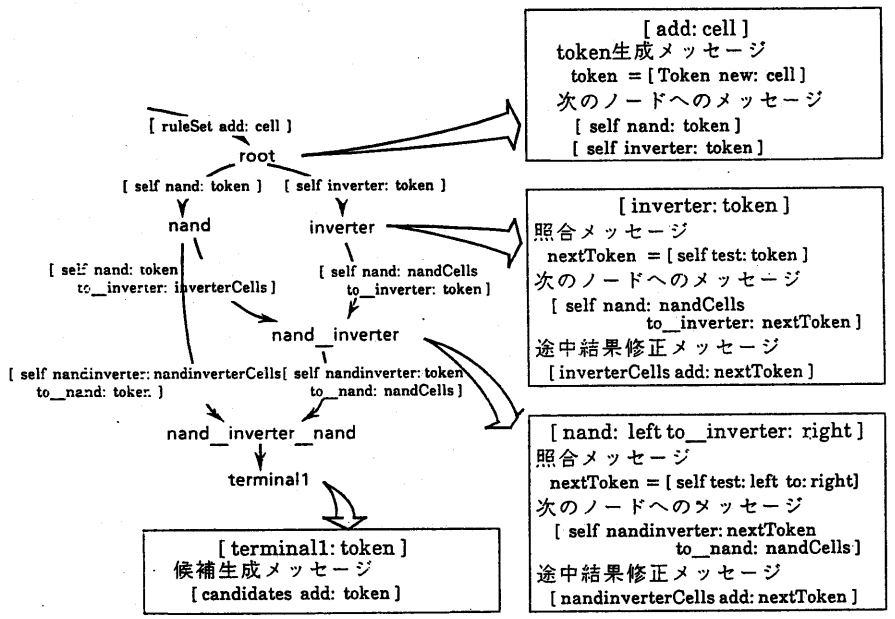


図6　Reteネットワークとメッセージ

5. 4　推論処理

具体的な推論処理はInferenceStrategyクラスに記述した戦略にしたがって行われる。例えば、変化したデータとしてinverterゲートをrootノードに送ると、rootノードでは、まずtokenを生成し、それをnandノードとinverterノードとにメッセージで送る。inverterノードでは照合に成功し、途中結果メモリinverterCellsにtokenを追加し、セルフメッセージ[self nand: nandCells to_inverter: token]を送る。このメッセージをnand_inverterノードが受け取り、流れてきたtokenに含まれているinveterゲートに対して、入力ネットと同じ出力ネットを持つゲートを求めよとメッセージを送り、求められたゲートがnandCellsに含まれているかを調べる。もし含まれていれば照合に成功したとしてtokenを合成し、次のノードにセルフメッセージを送る。さらに途中結果メモリnandinverterCellsに追加もする。terminalノードに到達すると、そのtokenからcandidateを生成してcandidatesの修正を行う。修正がすべて終わると競合解消戦略にしたがってcandidatesから選択されたルールを起動することによって推論が進む。

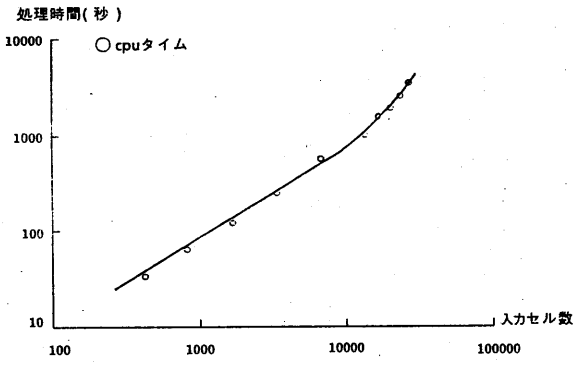
なお、MetaRuleには、どのRuleSetクラスをどの推論方法で処理するか、およびその順序等の戦略が記述されている。

6. 考察

提案する本手法の効果を調べるために、論理合成エキスパートシステムの約120の回路変換・最適化ルールを用いて、試作した推論システムの計測を行った。第7図に処理速度と回路規模との関係を示す。約1万ゲート程度まではほぼ比例関係が保たれている。しかし、1万ゲート以上では処理速度が極端に遅くなっている。

ほぼ比例関係が保たれている原因は、回路規模が大きくなっても回路の接続関係がそれほど複雑にはならないためにゲートがネット名の一致するゲートを検索する照合メソッドの処理速度が変わらないためであると考えられ、その様なデータ処理に対しては本手法は効果があると思われる。

また、１万ゲート以上で極端に遅くなった原因は、照合処理によるのではなく、競合解消に従った候補の追加・消去・選択の高速化を怠っていたために、実行可能な候補の増加に対して競合解消処理に時間がかかってしまったからである。



第７図　処理速度と回路規模の関係

さらにシステム構築において、ルールの優先度の高いルールを選択する戦略や、論理回路の出力側に近いデータとの照合に成功したルールを選択する戦略など回路変換・最適化に用いる競合解消戦略をオブジェクト指向の特徴である差分プログラミングによって容易に実現できた。

7. あとがき

本稿では、知識をオブジェクト指向で表現した時のクラス構成とその間のメッセージの流れについて述べた。さらに、ルールセットオブジェクト側では部分的な照合処理結果をReteネットを用いて記憶し、データオブジェクト側では照合処理の機能を持たせることによって、Reteアルゴリズムにオブジェクト指向を取り入れた照合処理方法についても述べた。

また試作システムを開発し、論理合成システムに適用し、その効果を確認した。

しかし今回の試作システムでは、静的知識の属性値の一致関係（ゲートのネット名の一致関係）を検索する照合処理機能を持たせたのみである。今後は、静的知識に一致関係以外の照合処理機能を持たせることにより汎用的なシェルへの拡張を行ない、本手法の有効性を確認していきたい。

参考文献

[1] Laursen,J. and Atkinson,R.:"Opus: A Smalltalk Production System", OOPSLA, pp337-387, (1982).

[2] 藤村,他:"オブジェクト指向知識表現ａｕｋを用いた知的システム構築用シェルＡＵＫ", 情報処理学会論文誌, Vol.31, No.1, pp.76-86 (1990)

[3] Forgy,C,L.:"Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem", Artificial Intelligence, Vol.19, pp17-37, (1982).

[4] 荒屋,他:"プロダクションシステムのための高速パターン照合アルゴリズム", 情報処理学会論文誌, Vol.27, No.7, pp.768-755 (1987)

[5] 石田:"プロダクションシステムにおける条件記述の最適化", 情報処理学会論文誌, Vol.29, No.12, pp.1158-1169 (1988)

[6]渡守武,他:"オブジェクト指向言語koola", 情報処理学会第４０回全国大会, pp.704-707

[7]西山,他:"論理合成エキスパートシステムＬＯＤＥＳ", 計測と制御, Vol.27, No.10, pp.923-924 (1988)

[8]松本;他:"論理合成エキスパートシステムＬＯＤＥＳ -回路変換における知識表現- ", 情報処理学会第３６回全国大会, pp.1975-1976 (1988)

[9]松本,他:"論理合成システムにおける高速な回路変換手法", 電子情報通信学会技術研究報告, Vol.89 No.93 pp.7-14 (1989)