TM-0817

並列動作系推論システムANDOR-IIによる学習システムの構築

坂本忠昭，高橋和子，
竹内彰一(三菱電機)

October, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列動作系推論システムＡＮＤＯＲ－Ⅱによる学習システムの構築

Implementation of Learning System by ANDOR II : An Inference System for Concurrent Systems

坂本　忠昭　　高橋　和子　　竹内　彰一
三菱電機㈱　中央研究所

並列動作系推論システムＡＮＤＯＲ－Ⅱの応用の１つとして、帰納的学習システムＣＩＧＯＬを取り上げ、その学習アルゴリズムの持つ並列性について検討し、ＡＮＤＯＲ－Ⅱによる実現方法について述べる。さらに、学習アルゴリズムの並列化に伴う問題点をあげ、それに対するＡＮＤＯＲ－Ⅱの拡張について検討する。

１．はじめに

複数の構成要素がお互いに作用を及ぼしながら並列に動作する系においては、その動作に非決定性が含まれる場合が多い。ところが、従来のCommitted Choice型の並列論理型言語は、論理積の関係にあるゴールリテラルを並列に実行するＡＮＤ並列性を有するものの、一度ある節が選択されると他の節の選択の可能性が棄却されるため、非決定的要素を含む動作系の記述に適しているとは言えない。このような並列動作系を記述する言語には、従来のＡＮＤ並列性の記述・推論能力に加えて、非決定性を含む問題を簡潔に記述でき、全ての可能世界に対する推論を効率よく行いながら全解を探索するＯＲ並列性が必要であり、近年その手法が幾つか報告されている。[2][4][6] 本稿で紹介する並列動作系推論システムＡＮＤＯＲ　Ⅱ[5] もこのような背景のもとで設計されたシステムである。ＡＮＤＯＲ－Ⅱは、ＡＮＤ・ＯＲ両並列性を持ち、非決定性を含む並列動作系の記述言語を提供すると共に、それをコンパイルすることによって、全解探索機能を持つＫＬ１[1] プログラムを生成する。

本稿では、ＡＮＤＯＲ－Ⅱの応用問題の１つとして帰納的学習システムの構築について述べる。帰納的学習システムは、外部から与えられた例をもとに、帰納推論によって知識を獲得するシステムである。ＡＮＤＯＲ－Ⅱの応用問題として帰納的学習システムを取り上げたのは以下の理由による。

①帰納的学習のプロセスは、入力された例をもとに、これらに一般化・特殊化等のオペレーションを施すことによって、入力された全ての例を満足し、新しい入力例の予測が可能な一般的記述を得ることである。オペレーションの適用は試行錯誤的であるため、これは、基本的に非決定的な探索問題である。従って、問題にＡＮＤ・ＯＲ両並列性が内在しており、まさにＡＮＤＯＲ－Ⅱが目的としている問題である。

②現在ＡＮＤＯＲ－Ⅱは、並列推論マシンMulti-PSI 上に移植中であるが、帰納的学習における探索問題はその規模が大きく、並列推論マシンを用いて解く応用問題として十分な計算量を持つ。

以下、第２章でＡＮＤＯＲ　Ⅱの概略を示し、第３章で応用問題として用いた帰納的学習システムの説明をする。続いて、第４章でＡＮＤＯＲ－Ⅱによる帰納的学習システムの構築における学習アルゴリズムの並列化及びその際の問題点について述べる。さらに、その問題に対するＡＮＤＯＲ－Ⅱの拡張について述べる。

２．ＡＮＤＯＲ　Ⅱ

ＡＮＤＯＲ－Ⅱは、並列動作系を対象とする問題解決システムとして開発された。ＡＮＤＯＲ－ⅡはＡＮＤ・ＯＲ両並列性を持ち、並列事象の簡潔な記述言語及び、全ての可能解の探索・推論機能を提供する。

ＡＮＤＯＲ－Ⅱが提供する記述言語は論理型言語であり、その述語はＡＮＤ関係とＯＲ関係に分類される。ＡＮＤ関係はガード付き節のみから定義され、そのリゾリューションは従来のCommitted Choice型言語と同様に、ある節が選択されると他の節の選択の可能性は棄却されるという形で実行される。一方ＯＲ関係はガードを持たない節のみから定義される。そのゴールリテラルのリゾリューションは、ヘッド部とユニファイ可能な全ての節の数だけ分岐し、互いに論理和の関係にある複数世界で並列に実行される。１つの述語はＡＮＤ関係かＯＲ関係のいずれかであり、それは予め宣言しなければならない。ボディ部には混在を許している。また、各述語の引数の入出力モードも予め宣言しなければならない。入力引数はゴールの計算中に参照しかされない引数であり、出力引数はそれ以外のケースである。

ＡＮＤＯＲ－Ⅱで記述されたプログラムは、ＡＮＤ・ＯＲ両並列性を実現するように、色付きストリームによるコンパイル方式に基づいてＫＬ１プログラムに変換される。ＯＲ関係によって世界が分岐した場合には、出力変数には各世界におけるリゾリューションの結果とそれが定義されている世界に付随する色のペアがストリームとして流される。このストリームを受け取ったゴールは、ストリーム中のデータの色をチェックし、同色のデータすなわち同一世界に属するデータ同士に対してのみリゾリューションを行う。この方式により、複数世界の処理を各世界ごとに並列に実行することができるわけである。

３．帰納的学習システム

ここでは、ＡＮＤＯＲ－Ⅱの応用問題として取り上げた帰納的学習システムＣＩＧＯＬ[3] を概説する。ＣＩＧＯＬは一階述語を獲得対象とした帰納的学習システムであり、リゾリューションを逆方向に行うinverse resolutionという方法を用いて、与えられた例から新しい述語を生成・獲得していくシステムである。我々がＡＮＤＯＲ－Ⅱの応用問題としてＣＩＧＯＬを取り上げたのは以下の理由による。まず、一般的記述を得るためのオペレーションのアルゴリズムが明確に定義されていること。そして、これらのオペレーションの適

用を行う制御アルゴリズムには並列性が含まれていることである。

まず、ＣＩＧＯＬの学習に用いられている３つのオペレーションについて述べる。

(1) truncation

２つのUnit節C1, C2が与えられた時、それらの両方を満足する一般的記述C を求めるオペレーションであり、教師から与えられた例の一般化に用いられる。これは、C1, C2に対するleast-general-generalizationである。図3-1 にtruncationの例を示す。

```
C1 = member(1,[j])
C2 = member(a,[a,b,c])
        ↓
C  = member(A,[A|B])
```

図3-1 truncationの例

(2) absorption

節C が２つの節C1, C2から導出される節であるとき、C1とC が与えられ、これらからC2を導出するオペレーションをabsorptionと呼ぶ。ただし、C1はUnit節に制限されている。absorptionによって節集合から節C が除かれ、節C2が追加される。図3-2 にabsorptionの例を示す。

```
C1 = append([A],B,[A|B])
C  = append([A,B],C,[A,B|C])
        ↓
C2 = append([A|B],C,[A|D]) :- append(B,C,D)
```

図3-2 absorptionの例

absorptionのアルゴリズムには非決定的要素が含まれているため、同じC1, C から複数のC2が生成される可能性がある。ただし、生成されるC2がすべて正しい節とは限らず、過度の一般化により意味を持たない節が生成されることもある。

(3) intra-construction

intra-constructionは複数個の節C1, C2, …, Cn が与えられたとき、新しい述語を導入してそれらを一般化するオペレーションである。図3-3 にintra-constructionの例を示す。

```
train(loco,[])
train(loco,[first,first])
train(loco,[first,second])
train(loco,[first,second,second])
        ↓
train(loco,A) :- p1(A)
p1([])
p1([first,first])
p1([first,second])
p1([first,second,second])
```

図3-3 intra-constructionの例

図中のp1はシステムが導入した新しい述語であり、その命名はユーザが行う。オペレーション後の述語train はオペレーション前の述語train の一般的記述になっているが、一般化の度合いによって複数の一般的記述が生成されることがある。

absorption, intra-constructionによる節の導出は、通常のリゾリューションとは逆方向であり、inverse resolutionと呼ばれる。

次にＣＩＧＯＬの学習アルゴリズムについて述べる。まず教師から対話的に例を１つ入力し、その時点における節集合内の節を用いてtruncationを行い例を一般化する。truncationが失敗した場合にはintra-constructionによって一般化する。一般化された節は節集合に追加される。そして、それらに対してabsorptionやintra-constructionを行いさらに一般化を進める。このとき、生成された節を教師に提示し、その節の採用・棄却を判断させる。このようにしてこれ以上新しい例の入力がなくなり、オペレーションの適用もできなくなった時点で終了する。

4．帰納的学習システムの構築

4．1　学習アルゴリズムの並列化

帰納的学習システムの構築に先立ち、ＣＩＧＯＬの学習アルゴリズムに内在する並列性について考える。まずＯＲ並列性であるが、ある時点での節集合（世界）に対し適用可能なオペレーションが複数存在する場合には、その数だけ新しい世界が生成される。加えて、absorptionやintra-constructionはそのオペレーション自身に非決定的要素を持つため、１回の適用によって複数の新しい世界が生成されることもある。これらの新しい世界について考えれば、これ以降の処理は他の世界に関係なく個々の世界独自に進めることができる。つまり、各世界はＯＲ並列に処理可能であり、このアルゴリズムはＯＲ並列性を持つことがわかる。次にＡＮＤ並列性であるが、あるオペレーションに着目した場合、その副処理の中には、お互いに独立に実行可能な処理や、リストの要素処理のようにパイプライン的実行が可能な処理が含まれている。これらの処理はＡＮＤ並列処理系のもとで実現されるものであり、従ってＡＮＤ並列性を持つと言える。このように、この学習アルゴリズムはＡＮＤ・ＯＲ両並列性を内在しているが、ＣＩＧＯＬは逐次処理系上のシステムゆえ、アルゴリズムも逐次型である。すなわち、ＡＮＤ並列は通常の逐次処理とし、ＯＲ並列はバックトラックによる線型探索として実現している。そして、教師との対話から得られる情報を探索制御に用いている。これに対し、ＡＮＤＯＲ－Ⅱは学習アルゴリズムが本来持っている並列性をそのままの形で実現できる。

4．2　ＡＮＤＯＲ－Ⅱによる実現と問題点

学習アルゴリズムの並列性を考慮してＡＮＤＯＲ－Ⅱで記述した帰納的学習システムのトップレベルのプログラム例を図4-1 に示す。図中の第１節はモード宣言部であり、各述語の引数が入力引数(+) か出力引数(-) かの宣言を行う。また、or_relation によって宣言されたものがＯＲ関係、その他がＡＮＤ関係の述語である。ＯＲ関係の述語inverse_prove/2, absorption/3はいずれも世界を分岐させるための述語である。以下、図中で定義されていない述語についてその処理内容を簡単に述べる。

①load_examples/1 : ファイルから例の読み込みを行う。
②generalize examples/2 : truncationあるいはintra-constructionによって例を１段階一般化する。
③create_c_c1_pairs/2 : 現在の節集合W からabsorptionの入力となるC, C1の組合せを作りリストにして返す。
④exec_abs/4 : C, C1に対してabsorptionを実行し、新しい節集合のリストWLを生成する。
⑤intra_check/2 : 現在の節集合W の中でintra constructionの適用可能な節の集合W1を求める。
⑥exec_intra/3 : intra constructionを実行し、新しい節集合のリストWLを生成する。

このプログラムではabsorptionを優先的に行い、intra-constructionが行えないかまたは失敗した場合にその世界を１つの組としている。

このプログラムの動作を検討した結果、実行効率の面で以下の問題点が明らかになった。すなわち、ＯＲ並列で新しい世界が次々と生成されていくが、その数が多くなると並列処理マシンと言えどもその負荷が大きくなりすぎ、実行効率が低下するというものである。従って、次のような制御を行う必要がある。

①新しい世界の生成を抑制する。
評価関数や制約を用いて、解に到達しないと予測される世界はできるだけ生成されないようにする。
②同じ内容を持つ世界は削除する。
ＯＲ並列時に個々の世界はお互いに独立に処理を進めるが、場合によっては幾つかの世界の内容が外見は異なっていても実質的には同じになることがある。この場合、それ以降の処理はすべて等しいため、どれか１つの世界の処理さえ実行されればよい。従って、他の世界の情報をモニタ- し、もし内容の等しいものがあれば、どちらかを削除する処理が必要である。

まず①に対しては、新しい世界を生成するオペレーションの前処理部であるcreate_c_c1_pairs/2及びintra_check/2において、absorptionの入力であるC, C1のペア及びintra-constructionの入力である節集合に対する評価を行い、条件を満たさないものや評価値の低いものは棄却する。但し、この評価は一種のヒューリスティクスであるため、容易にチューニングできるようにモジュール化しておくことが必要である。

②に対しては、ＯＲ並列問題について考えると、本システムに限らず、ＯＲ並列で分岐した各世界が全く独立に処理を進めるのではなく、世界間で何等かの情報交換を行いながら処理を進めるという場合は多いと考えられる。そこで、この世界間の通信を簡単に記述できるように、ＡＮＤＯＲ－Ⅱを拡張することを考える。

```
:- mode main(-), load_examples(-),
        generalize_examples(+,-), inverse_prove(+,-),
        create_c_c1_pairs(+,-),
        inverse_porove_sub(+,+,-), absorption(+,+,-),
        exec abs(+,+,+,-), intra_construction(+,-),
        intra_check(+,-), intra_sub(+,+,-),
        exec_intra(+,-), intra_check2(+,+,-).

main(Ws) :- true |
        load_examples(E),
        generalize_examples(E,W),
        inverse_prove([W],Ws).

:- or_relation inverse_prove/2.
inverse_prove([W|T],Ws) :- create_c_c1_pairs(W,L),
                           inverse_prove_sub(L,W,Ws).
inverse_prove([_|T],Ws) :- inverse_prove(T,Ws).

inverse_prove_sub(L,W,Ws) :- L \= [] |
        absorption(L,W,Ws).
inverse_prove_sub([],W,Ws) :- true |
        intra construction(W,Ws).

:- or_relation absorption/3.
absorption([[C,C1]|T],W,Ws) :- exec abs(C,C1,W,WL),
                               inverse_prove(WL,Ws).
absorption([_|T],W,Ws) :- absorption(T,W,Ws).

intra_construction(W,Ws) :- true |
        intra check(W,W1),
        intra sub(W1,W,Ws).

intra_sub([],W,Ws) :- true | Ws = W.
intra_sub(W1,W,Ws) :- W1 \= [] |
        exec_intra(W1,W,WL),
        intra_check2(WL,W,Ws).

intra_check2([],W,Ws) :- true | Ws = W.
intra check2(WL,_,Ws) :- WL \= [] |
        inverse_prove(WL,Ws).
```

図4-1 帰納的学習システムのトップレベルのプログラム例

4．3　ＡＮＤＯＲ－Ⅱの拡張

まず、ＯＲ並列時の世界間の通信機能の実現方法について考える。この場合、世界が動的に変化するため、一つの世界から他の世界に直接通信路を開くことは困難である。加えて、この方法では世界数n に対する通信路数は(n-1)!となるため、世界数の増加と共に通信路の数も著しく増加することが予想される。そこで、共通の黒板を１つ用意し、これを用いて情報交換を行う方法をとる。

図4-1 とよく似た動きをする簡単なプロダクションシステムを例にあげて、この通信機能をまずＫＬ１で実現する。図4-2 に、通信機能を持たないプロダクションシステムのプログラムを示す。このプログラムはＡＮＤ並列性しか持たないＫＬ１で書かれているため、ＯＲ並列は疑型探索に置き換えてられている。トップレベル述語main/3への入力引数は、次の通りである。

Rules : ルールのリスト。ルールの形式は[<IF>,<THEN>]。
但し、<IF>, <THEN>はアトムである。
WM : ワーキングメモリ。形式は[<STATE1>,…,<STATEn>]。
但し、<STATEi>はアトムである。

プログラム中の述語select_rules/3は、現時点のワーキングメモリに適用可能なルールを全て求める処理を行い、述語apply_rules/3は、ワーキングメモリに対してselect_rules/3で得られたルールを適用し、ルール数に等しい数の新しいワーキングメモリを生成する処理を行う。このプログラムでは、内容の等しいワーキングメモリが生成されても、システムはそれに気づかず同じ処理を行う。

次に、黒板による通信機能を付加したプログラムを図4-3

```
main(Rules,WM,WMs) :- true |
        merge(WMs0,WMs),
        ps([WM],Rules,WMs0).

ps([],_,WMs) :- true | WMs = [].
ps([WM|T],Rules,WMs) :- true |
        select_rules(Rules,WM,SRules),
        ps_sub(SRules,WM,WMs1,Rules),
        WMs = {WMs1,WMs2},
        ps(T,Rules,WMs2).

ps_sub([],WM,WMs1,_) :- true | WMs1 = [WM].
ps_sub(SRules,WM,WMs1,Rules) :- SRules \= [] |
        apply_rules(SRules,WM,WMs2),
        merge(WMs2,WMs3),
        ps(WMs3,Rules,WMs1).
```

図4-2 プロダクションシステムのプログラム例
（通信機能なし）

に示す。まずmain/3内のbb_manager/2によって黒板管理者との間に通信路を開く。そして、通信路は変数S によって保持され、ワーキングメモリの生成に合わせて分岐される。従って、１つのワーキングメモリは必ず１つの通信路を持つ。新しいワーキングメモリが生成された場合には、まずメッセージask(WM,Ans) を黒板管理者に送り、それが過去に現れているかを検索してもらう。既に現れている(Ans=old) 場合にはその先の処理は行わず、過去に現れていない(Ans=new) 場合には処理を続ける。黒板管理者は、通信路から送られてくるワーキングメモリが以前に現れていなければ黒板に書き込む処理も行う。このようにして、ＯＲ並列における同じ内容の世界の削除を行うことができる。

```
main(Rules,WM,WMs) :- true |
        bb_manager([],[open|S]),
        merge(WMs0,WMs),
        merge(S1,S),
        ps([WM],Rules,WMs0,S1).

ps([],_,WMs,S) :- true | WMs = [], S = [].
ps([WM|T],Rules,WMs,S) :- true |
        S = {[ask(WM,Ans)],S1},
        ps_check(Ans,WM,T,Rules,WMs,S1).

ps_check(new,WM,T,Rules,WMs,S) :- true |
        select_rules(Rules,WM,SRules),
        ps_sub(SRules,WM,WMs1,Rules,S1),
        WMs = {WMs1,WMs2},
        S = {S1,S2},
        ps(T,Rules,WMs2,S2).
ps_check(old,_,T,Rules,WMs,S) :- true |
        ps(T,Rules,WMs,S).

ps_sub([],WM,WMs1,_,S) :- true |
        WMs1 = [WM], S = [].
ps_sub(SRules,WM,WMs1,Rules,S) :- SRules \= [] |
        apply_rules(SRules,WM,WMs2),
        merge(WMs2,WMs3),
        ps(WMs3,Rules,WMs1,S).
```

図4-3 プロダクションシステムのプログラム例
（通信機能付き）

さて、ＡＮＤＯＲ－Ⅱの拡張についてであるが、ＡＮＤＯＲ－ⅡはＡＮＤ・ＯＲ並列を含む記述をＫＬ１にコンパイルするシステムである。従って、ＡＮＤＯＲ－Ⅱ上で図4-3 と同様の記述を行えば通信機能を実現することは可能である。しかしこの場合、ユーザ側の要求はある変数の値が以前出現したものか否かを知ることであるから、ＡＮＤＯＲ－Ⅱ上で組み込み述語風に例えば

```
check_value(WM,Ans)
```

と記述しておけば、ＫＬ１にコンパイルしたときに図4-3 のように通信路のオープン、問い合わせ、通信路の引き回し等の処理が自動的に付加されるのが理想である。また、このようなチェック以外にも通信路を用いた汎用的な処理があれば、同様の形式で実現することができる。現在、これらの実現に向けて検討中であるが、コンパイル時に、通信路を保持する変数を適切な節内に埋め込む処理が難しいと思われる。

5．おわりに

並列動作系推論システムＡＮＤＯＲ－Ⅱの応用として、帰納的学習システムの構築について述べた。まず、帰納的学習システムＣＩＧＯＬの学習アルゴリズムの並列性について検討し、ＡＮＤＯＲ－Ⅱがその並列性を直接記述できることを示した。また、学習アルゴリズムを並列化した場合に起こる実行効率面での問題点をあげ、特にＯＲ並列時に同じ内容の世界が出現する問題に対して、その削除を行うための世界間の通信機能の実現方法をＫＬ１上で示した。現在、帰納的学習システムのインプリメント及びＡＮＤＯＲ－Ⅱによる世界間の通信機能のコンパイル方法の検討を行っている。

謝　辞

本研究は第５世代コンピュータ・プロジェクトの一環として行われた。日頃御指導をいただいているＩＣＯＴ古川研究所次長及び長谷川第１研究室室長に感謝致します。

参考文献

[1] Chikayama,T.,Sato,H. and Miyazaki,T. : Overview of the Parallel Inference Machine Operating System (PIMOS), Proc. of Int. Conf. on FGCS'88(1988), pp.230-251.

[2] Clark,K.L. and Gregory,S. : PARLOG and Prolog United, Proc. of 4th Int. Conf. on Logic Programming(1987), pp.927-961.

[3] Muggleton,S. and Buntine,W. : Machine Invention of First-order Predicates by Inverting Resolution, Proc. of Machine Learning 88(1988), pp.1-14.

[4] Naish,L. : Parallelizing NU-Prolog, Proc. of Logic Programming(1988), pp.1546-1564.

[5] Takeuchi,A.,Takahashi,K. and Shimizu,H. : A Parallel Problem Solving Language for Concurrent Systems, ICOT TR-418, 1988.

[6] Yang,R. and H.Aiso : P-Prolog: A Parallel Logic Language Based Exclusive Relation, Proc. of 3rd Int. Conf. of Logic Programming(1986), pp.255-269.