知識ベースシステムにおける仮説推論の高速化に関する研究

1998年

加藤昇平

目 次

第1章
序論

1.1 研究の背景と目的

人間が持つ知能活動や高度な知識処理を計算機上で実現しようという計算機科学の研究分野に人工知能研究があるが，人工知能の実用化や知識情報処理システム構築のための要素技術として不可欠な研究課題として知識ベースシステムが挙げられる．

同システムにおいて現在のところ実現されている推論機能は，プロダクションシステム等に見られるように比較的単純な演繹的推論が中心である．しかるに，知識ベースシステムの高機能化のためには，演繹的推論以外にも，仮説に基づく推論，帰納的推論，デフォルト推論，不確かな知識の利用など，より高度で複雑な推論機能を用いて，柔軟で知的な処理を行なうことが期待されている．本研究では高次推論の一形式である仮説推論に着目する．

仮説推論は，真か偽か不明な事柄を取り敢えず真と考えて(仮説を立てて)推論を進め，矛盾なくうまく問題が解決できれば立てた仮説は正しかったと考えるという推論である[石塚88][井上92]．仮説という真か偽か不明で他の知識と矛盾する可能性を有する不完全な知識も扱うということで，知識ベースの能力の幅を広げることができる．

仮説推論が扱う知識ベースでは，事実の知識（対象世界でつねに成り立ち矛盾の可能性がない知識）の集合Fと仮説の知識（対象世界でつねに成り立つとは限らず矛盾を引き起こす可能性のある知識）の集合Hが分けられて保持される．Hの部分集合をhとする．

Pooleの提唱する論理的枠組[Poo88]によれば，仮説推論の基本的な推論動作は次のように実現される．観測（ゴール）$\mathcal{O}$が与えられたとき，まずFから$\mathcal{O}$が演繹的に証明

できるか確かめる．Fからだけでは証明できないとき，次の条件を満たすhを求める．

$F \cup h$から$\mathcal{O}$が説明できて（$F \cup h \vdash \mathcal{O}$），かつ，

$F \cup h$は無矛盾である（$F \cup h \not\vdash \square$）．

すなわち，事実の知識Fだけで$\mathcal{O}$を証明できればそれでよいが，事実の知識Fだけでは$\mathcal{O}$を証明できないとき，仮説の集合Hから無矛盾な部分集合hを切り出しFと合わせて$\mathcal{O}$を証明するようにする．多くの場合にはhは最小性（hの部分集合が上記の条件を満たさないこと）も要請される．

人間の頭脳の中にも互いに矛盾を引き起こす可能性のあるような知識が多数あるが，与えられた一つの問題に対する解を求める場合には，人間はある無矛盾な知識の範囲を切り出してきて問題に対処する．仮説推論はこのような人間の思考法に対応しているといえる．

通常の演繹推論では，知識ベースからゴール$\mathcal{O}$が証明できるとき成功（$\mathcal{O}$は真），証明できないとき失敗（$\mathcal{O}$は偽）となる．これに対して仮説推論では$\mathcal{O}$は証明されることが条件となり，この証明に必要な無矛盾な仮説hを生成する．すなわち，演繹推論を逆向きに利用して解を求める構造になっている．

上記のように仮説推論は，不完全な知識の下で適切な推論を行なう高次推論の一形式であり，柔軟で知的な知識処理を行なうことができ，人工知能の実用化の一例として現在広く使われている診断や設計･計画のためのエキスパートシステムの能力向上のための鍵となる推論処理として，注目を浴びている．しかしながら，仮説推論システム実現上の問題点として，仮説推論の計算量が極めて大きいことが知られているので，いかにして仮説推論の探索空間を絞り込む工夫を行なうかということが重要な課題となっている．

そこで本研究では，特に知識ベースシステムの推論処理の高機能化として仮説推論に着目し，その推論処理の高速化技法について検討を行なう．

1.2 論文の構成

プログラム解析に基づく仮説推論の高速化

第2章では仮説推論もつ推論処理の特性に着目し,「無矛盾性の検査の効率化」の側面から推論処理の高速化を試みる.

仮説推論を前節のような論理的枠組で捉えると，論理プログラミングや演繹データベースの分野においてこれまでに研究されてきた問合せ処理における様々な最適化技術が適用できることは，容易に予想される. 実際, [Sti91], [OI92],[KMI91] などの研究においては，演繹データベースの分野で提案された上昇型計算法（例えばマジックセット法[BMSU86]）や下降型計算法（例えばQSQ法[Vie86]など）を仮説推論に適用し，それらの問合せ処理が持つ，(i) 同一のサブゴールを繰り返し解く重複計算を排除し，(ii) 与えられたゴールに無関係な推論を回避するという「ゴール指向」(goal-directed）の推論を行なう性質を利用して，効率の良い仮説推論を提案している.

しかるに，仮説推論と演繹ベータベースにおける通常の問合せ処理とが異なる点としては,与えられた観測を説明する仮説集合が，知識ベース内の無矛盾性制約と矛盾しないかという無矛盾性の検査を受けなければならないことである. この無矛盾性検査のための推論は与えられたゴールとは無関係に大域的に行なわれるので，マジックセット法やQSQ法の持つ「ゴール指向性」の利点が失われることになる.

従って本論文では，まず，この問題を解決するために知識ベース（プログラム）に出現する述語の依存関係を解析することによる仮説推論の効率化技法を提案する [加藤 94]. 本プログラム解析法は，以下の2つの処理からなる.

(1) 仮説依存関係の解析

与えられた観測に対する説明がどのような仮説に依存するかを命題論理のレベルで近似的に解析する. 同様に，知識ベース内の無矛盾性制約節についても，それがどのような仮説に依存するかを近似的に解析する.

(2) 仮説依存関係による制約節の検査

仮説依存関係の解析結果から，無矛盾性制約節の評価時における無駄な探索を検出する．

本プログラム解析によって，従来の効率化法では解析不可能だった仮説の依存関係が事前に解析できるようになり，その結果無矛盾性の検査時における不要な探索が検出できるようになる．

コストに基づく仮説推論における推論制御の最適化

第3章では,「最適な説明の効率的な探索方法」の側面から仮説推論の高速化手法を提案する．

仮説推論では一般に，観測を説明する仮説集合は複数存在する．しかるに，計画·診断問題等において見られるように，要求される解はすべての説明ではなく，最も好ましい説明であることが多い．そこで仮説推論の課題として，与えられた観測をより適切に説明するような仮説を推論の過程においてどのように選択するかという「合理的な仮説の選択」の問題が認識されている [井上 92]．

従って本研究では，仮説の選択の基準として各仮説に重み（コスト）を与え，与えられた観測に対して最良の説明を求める仮説推論を考える．各仮説に与える重みは正の数とし，説明が最良であるとは，説明のコスト，すなわち，観測の説明に現れる仮説の重みの和が最小であることとする．

各仮説に重みが与えられることによって，最良の説明を求める問題は，ヒューリスティック探索の問題に帰着される．従って本研究では，論理プログラミングや演繹データベースの分野における問合せ処理技術に対してヒューリスティック探索の持つ探索制御技術を導入し，与えられた観測を説明する最小コストの仮説集合を効率的に求める仮説推論システムを提案する．

まず，推論の課程における探索木を形成するゴール節 gの評価値 $h(g)$ を定義し，ヒューリスティックな評価関数を最も効率的に利用できる最適解探索法のひとつである A^*アルゴリズム[Nil80]が持つ探索制御技術に着目し，A*アルゴリズムを用いた仮説

推論アルゴリズムを提案する．さらに，本研究で提案する推論方法の探索制御能力を決定づけるヒューリスティック評価関数の導出方法を提案する．本導出法を用いることでより効果的なヒューリスティクスが得られ，その結果，従来の推論システムよりも効率的に最適な説明が得られることを例題を用いて確認する．

PARCAR: コストに基づく並列仮説推論システム

第 4章では，2章 3章で述べる仮説推論の高速化手法に加えて，仮説推論の探索空間を複数のプロセッサへ分散し「推論処理を並列化」することにより，大規模な知識データベースにおいて高速な仮説推論処理を実現する [KSI96].

仮説推論の推論処理技術に並列ヒューリスティック探索が持つ探索制御技術を導入し，与えられた観測を説明する最小コストの仮説集合を効率的に求めることにより最適な説明を得る並列仮説推論システムを提案する．本システムでは「進行フェイズ」および「確認フェイズ」の 2 段階の状態を経て推論が実行される．

「進行フェイズ」では，各プロセッサは，与えられた観測に対する説明が求まるまで以下の 3 つの手続きを繰り返す．

(1) ゴール節展開手続き 割り当てられた仮説推論の探索空間について局所的に推論を行う．生成されたサブゴール節の数が一定の量に達するまで，最良なゴール節に対する SLD 導出および生成されたサブゴール節に対する無矛盾性の検査を繰り返す．

(2) サブゴール節分散手続き ゴール展開手続きにおいて生成された探索空間の葉にあたるのゴール節をその評価値が小さい物から順にゴール節の数が均一となるように全プロセッサに分散する．

(3) サブゴール節受信手続き 全プロセッサから送信されたゴール節を受信し探索空間に追加する．

「進行フェイズ」においてあるプロセッサにて観測に対する説明が求まると，全プロセッサは「確認フェイズ」に移行し，求められた説明の最適性を検査する．

仮説推論のノード展開は述語のユニファイや変数の束縛情報の伝搬等をともなうSLD導出による．したがってノード展開にかかる計算量は大きく，生成される子ノードの数によってその計算量は大きく異なる．そこで本並列仮説推論システムでは，各プロセッサで一回の反復で行なう推論の処理量は展開されるノード数ではなく，生成されるノード数に基づくものと考え，全プロセッサ間でそのノード数が一定になるように負荷分散を行っている．また，仮説推論では展開されるノードは述語論理ホーン節で表現される．したがってノードのデータ構造は複雑でありノード分散にかかる通信コストも大きくなる．そこで本並列仮説推論システムでは各プロセッサが1反復に処理する処理量をある程度大きくし，プロセッサ間の通信回数を抑え，通信コストを抑えている．

本研究で提案する並列仮説推論システムはMIMD型分散メモリ並列計算機である富士通AP1000上実装され，実験により計算パフォーマンスの高い仮説推論処理を確認する．

第 2 章
プログラム解析に基づく仮説推論の高速化

2.1 はじめに

仮説推論の研究に関しては，その理論的枠組の提案，効率的な推論システムの実現方法，並びに診断·設計問題等への応用など，多くの研究結果が報告されている（例えば，文献 [Poo88],[井上 92]）．序論でも述べたように，Poole の定式化によって，論理プログラミングや演繹データベースの分野においてこれまでに研究されてきた問合せ処理における様々な最適化技術が仮説推論に適用され，効率的な仮説推論システムが提案されている．実際, [Sti91],[OI92],[KMI91] などの研究においては，演繹データベースの分野で提案された上昇型計算法（例えばマジックセット法[BMSU86]）や下降型計算法（例えば QSQ 法[Vie86]など）を仮説推論に適用し，それらの問合せ処理が持つ，(i) 同一のサブゴールを繰り返し解く重複計算を排除し，(ii) 与えられたゴールに無関係な推論を回避するという「ゴール指向」（goal-directed）の推論を行なう性質を利用して，効率の良い仮説推論を提案している．しかるに，仮説推論と演繹ベータベースにおける通常の問合せ処理とが異なる点としては, 与えられた観測を説明する仮説集合が，知識ベース内の無矛盾性制約と矛盾しないかという無矛盾性の検査を受けなければならないことである．無矛盾性制約の知識は，通常負節 (negative clauses) によって知識ベースに与えられる．そのため，この無矛盾性検査のための推論は与えられたゴールとは無関係に大域的に行なわれ，その結果，マジックセット法や QSQ 法の持つ「ゴール指向性」の利点が失われることになる．

従って本章では，この問題を解決するために，知識ベース（プログラム）に出現する述語の依存関係を解析することによる仮説推論の効率化技法を提案する[1]．本効率

化技法の特徴は，与えられたゴールに無関係な無矛盾性制約を検出するのみならず，ゴールに関係がある無矛盾性制約式に対しても，それに対する無矛盾性検査時の不必要な探索空間の刈り込みを試みる点にある．本研究で述べる効率化法を計算機上に実装し，その実験結果についても報告する．

まず**2.2節**で従来の仮説推論の方式について述べ，その問題点を指摘し，**2.3節**でその問題点を解決するプログラム解析法を後向き推論の枠組で説明し，**2.4節**でこの解析法のマジックセット法に基づく前向き推論による実装化について述べる．そして，**2.5節**で実験結果について説明する．

2.2 仮説推論

2.2.1 仮説推論の定式化

ここでは，本章で用いるPooleにより提案された仮説推論の枠組[Poo88]について説明する．

定義 2.2.1 [Poo88] ホーン節集合F（「事実」と呼ぶ）と，単位節の集合H（「仮説集合」と呼ぶ）が与えられたとする．また，存在束縛されたアトムの連言O（「観測」，または単に「問合せ」と呼ぶ）が与えられたとする．このとき，Oの$F \cup H$による説明とは，以下の条件を満足するような，Hの要素の代入例から成る集合hを求めることである[2]．

$F \cup h \vdash O$ （$F \cup E$からOが証明される） (AR1)

$F \cup h \not\vdash \square$ （$F \cup E$は無矛盾である） (AR2)

■

[1] 本稿は文献 [KSI93b], [KSI93a] の報告をもとにして本論文にまとめたものである．

[2] 一般に，条件 (AR1),(AR2) を満たすようなhは複数存在するが，本章では，このようなhをすべて求める全解探索について考える．

ここで，Fは確定節と負節からなる集合であり常に成り立つ知識として扱われる．一方，Hの要素の代入例からなる集合はFと矛盾する可能性がある．

定義 2.2.2 Fの中に存在する確定節を「ルール」と呼び，Fの中に存在する負節を「無矛盾性制約節」(あるいは単に「制約節」) と呼ぶ．制約節は，論理式 $false \leftarrow A_1, \cdots, A_n.$ で表現される．ただし，A_k $(1 \leq k \leq n;\ n \geq 1)$ はアトムであり，$false$ は,「矛盾」を表す．また, 制約節がF内に複数個ある場合は, $false_i$ $(i \geq 1)$ のように添字をつけて互いに区別する． ■

例 2.2.1 図 2.1に示す簡単な例題[太田 91]を考える[3]．部門 s1 の一人 X と部門 s2 の一人 Y が会議室 Z で会議を行なう (述語 m(X,Y,Z) で表す)，あるいは，ラウンジ Z で談合を行なう (d(X,Y,Z) で表す) 場合のスケジューリング問題である．

ルール			
m(X,Y,Z)	←	hp(X,s1),hp(Y,s2),v(Z).	(1)
d(X,Y,Z)	←	hp(X,s1),hp(Y,s2),q(Z).	(2)
v(Z)	←	r(Z),hv(Z).	(3)
q(Z)	←	l(Z),hq(Z).	(4)
a(Z)	←	r(Z),hv(Z).	(5)
a(Z)	←	l(Z),hq(Z).	(6)
r(101). r(102). l(201). l(202). l(203). l(204). na(101). na(204).			(DB)
仮説集合			
hv(Z). hq(Z). hp(b,s1). hp(c,s1). hp(e,s2). hp(f,s2).			(H)
無矛盾性制約			
$false_7$	←	hp(X,D),nhp(X,D).	(7)
$false_8$	←	a(Z),na(Z).	(8)

図 2.1: 例題知識ベース P_{ex}

ルール:

- 会議を行なうには部門 s1 の X と s2 の Y が出席すると仮定でき (それぞれ hp(X,s1) と hp(Y,s2) で表す)，かつ会議室 Z が空いている (v(Z) で表す) ことが条件であり，談合を行なうには s1 の X と s2 の Y が出席すると仮定でき，かつラウンジが

[3] この例は非再帰的なルールで表現されているが，本研究で提案する仮説推論システムは再帰的な知識ベースにも対応できる．再帰的な知識ベースに対する適用例としては文献 [KSI93a] を参照されたい．

静かである（q(Z) で表す）ことが条件である．図 2.1では (1),(3),(2),(4) のルールで表現されている（以下同様)。

- 会議室は 101,102 の 2 室，ラウンジは 201 から 204 の 4 室が用意されている．(DB)
- 会議室 101 及びラウンジ 204 は利用できないことがわかっている．(DB)

仮説集合：

- 通常会議室は空いており，ラウンジは静かである．また，部門 s1 の b は通常，会議及び談合に出席でき，これを hp(b,s1) で表す（c,e,f についても同様である）．(H)

無矛盾性制約：

- 部門 D の X が，会議もしくは談合に出席できるであろうと仮説が立てられているのに出席できない時，知識ベースは矛盾する．(7)
- 「利用できる」と仮定された会議室またはラウンジが利用できない時，知識ベースは矛盾する．(8),(5),(6)

ここで,「部門 s1 の b は誰とどこで会議を行なうことができるか」というゴール "$\leftarrow m(b,Y,Z)$" が与えられたとする．なお，単位節以外の節を識別するために，識別番号として各節に自然数を割り当てる．この番号を「節番号」と呼ぶ．図 2.1においては（ ）内の数字が，対応する節の節番号であるとする．

図 2.1の知識ベースが事実と仮説の和集合 $F \cup H$であり，ゴール "$\leftarrow m(b,Y,Z)$" が観測 $\mathcal{O}$である． □

図 2.2に例 2.2.1に対する OLDT 反駁[TS86]を用いた仮説推論の実行例を示す．OLDT 反駁とは，QSQ 法等と同様に通常の SLD 反駁にループチェック機能とレンマの利用による解の再利用の機能を導入したものである[4]．また，同図においてゴール節に現れる M なるオペレータが付いたアトム "MA" は，論理的には $true$ を表し，記号 M は単に，サブゴール$\leftarrow A_0$（ここでアトム A は A_0のある代入例）を解く際に,仮説を入力節として導出を行なったこと（すなわち，A に対して仮説が立てられたこと）を

記録するためのマークである.

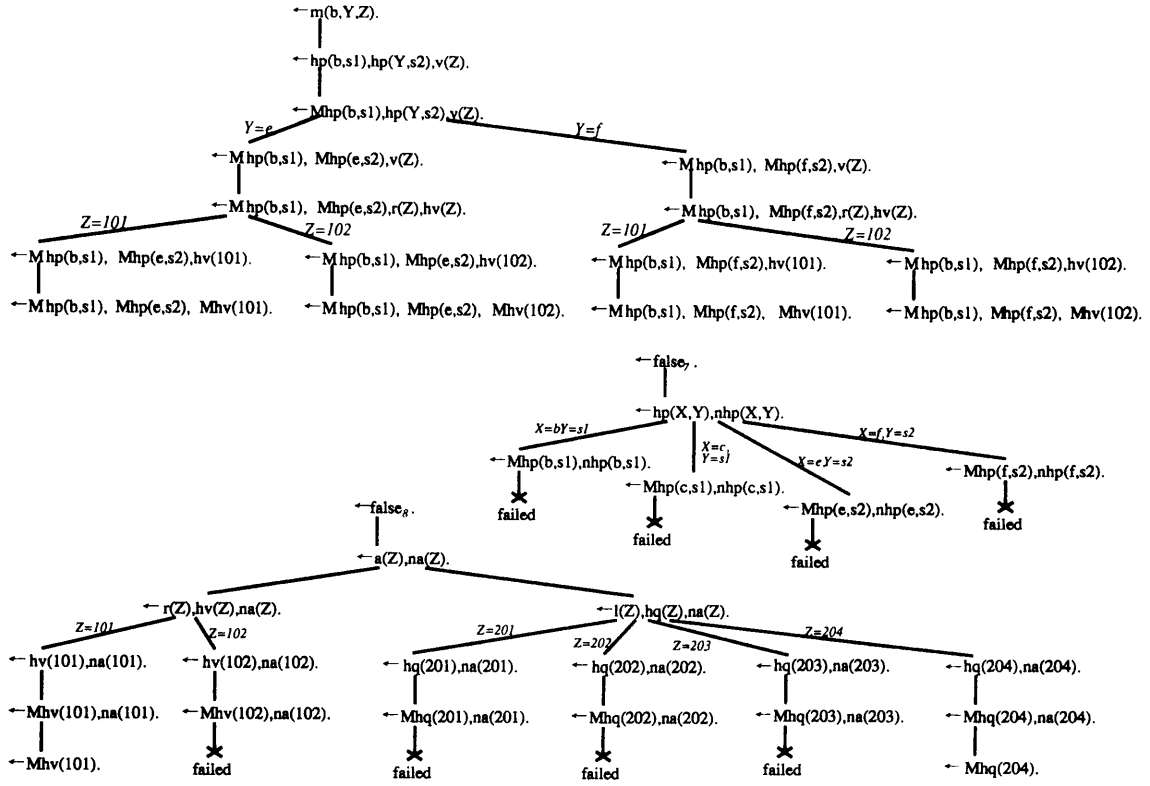


図 2.2: 仮説推論の実行例

図 2.2において，ゴール$\leftarrow m(b,Y,Z)$に対する反駁木の一番左側の枝から，仮説集合$\{hp(b,s1), hp(e,s2), hv(101)\}$の説明の下で解 $m(b,e,101)$ が得られる．しかしながら，ゴール$\leftarrow false_8$に対する反駁木から，仮説 $hv(101)$ を立てる，または，仮説 $hq(204)$ を立てることは知識ベースに対し矛盾することが判る．従って，解 $m(b,e,101)$ に対する説明は矛盾するのでこの解は却下される．

一方で，ゴール$\leftarrow m(b,Y,Z)$に対する反駁木の左から二番目の枝からは，仮説集合 $\{hp(b,s1), hp(e,s2), hv(102)\}$ が無矛盾であるので，この仮説集合を説明として解 $m(b,e,102)$ が得られる．

このように仮説推論では，与えられた観測を説明する仮説集合は，知識ベース内の無矛盾性制約と矛盾しないかという無矛盾性の検査を受けなければならない．しかし，

[4] ここでは説明の簡単のため，解テーブルは省略する．

この無矛盾性の検査のための推論は，必ずしも効率的ではなく，時には無駄な処理を伴う．

2.2.2 関連研究とその問題点

これまでに提案されている仮説推論システムとしては，下降型（後向き）推論に基づくもの[NM91][KMI91]や，マジックセット法などの上昇型（前向き）計算を利用したもの[Sti91][OI92]などがある．これらのシステムでは，観測に対する推論において真であると仮定された仮説の集合に対する無矛盾性の検査は，制約節（負節）に対する反駁木を形成して矛盾を引き起こす仮説集合を求め，これらの集合と仮定された仮説の集合との包含関係を調べることによって行なわれる[5]．その結果，制約節の評価時には，与えられたゴールとは無関係な部分の反駁木をも計算してしまう可能性がある．

例えば，図 2.2においてゴール$\leftarrow false_8$に対する反駁木の右側の部分木より，仮説$hq(204)$を立てることは知識ベースに対し矛盾することが判るが，ゴール$\leftarrow m(b,Y,Z)$に対する反駁木において，仮説$hq(204)$は全く現れない．従って，この部分木はゴール$\leftarrow m(b,Y,Z)$には無関係であり，このような部分木に対し推論を行なうことは無駄である．同様にして，ゴール$\leftarrow false_7$については，もはや反駁を行なう必要がないことが判る．すなわち，仮説推論においては無矛盾性の検査が要求されるために，問合せ処理における「ゴール指向性」を失っていることになる．

このような問題点に関しては，文献 [OI92] を除き，ほとんど研究がなされていない．文献 [OI92] では，与えられたゴールに無関係な制約節を検出する方法が提案されている．例えば図 2.2において，ゴール$\leftarrow false_7$に対する反駁を回避することができるが，ゴール$\leftarrow false_8$に対しては全計算をしてしまい，仮説推論時において「ゴール指向性」を失っている．

[5]無矛盾性の検査については，問合せ処理の後に，観測に対する推論において仮定された仮説の集合と事実を用いて前向き推論を行ない，矛盾 (false) が導出されるかどうかを調べる方法があるが（例えば，文献 [NM91]），本章では，後向き推論を用いて問合せ処理と同時に制約節を評価する方法を取ることにより，問合せ処理における反駁木の形成途中でも，あるゴールに矛盾が生じた場合はこれを早めに (成功葉に至る以前に) 刈り込むことを可能にしている．

そこで本研究では，上記の問題点を解決する一方法について以下の節で説明する．

2.3 プログラム解析を用いた探索空間の枝刈り

本節では，無矛盾性制約節の評価時における「ゴール指向性」を実現するためのプログラム解析法を提案する．

ここでは，説明のため OLDT 反駁を用いて本解析法を提案し，次節において，マジックセット法に基づいた前向き推論を用いた本解析の実装について述べる．

本研究で提案するプログラム解析法は，以下の 2 つの処理からなる．

(1) 仮説依存関係の解析 与えられた観測に対する説明がどのような仮説に依存するかを近似的に解析する．同様に，知識ベース内の無矛盾性制約節についても，それがどのような仮説に依存するかを近似的に解析する．

(2) 仮説依存関係による制約節の検査 仮説依存関係の解析結果から，無矛盾性制約節の評価時における無駄な探索を検出する．

与えられた問合せと無矛盾性制約節との論理的な依存関係を近似的に解析するために，本研究では述語論理式で表現された知識ベース並びに問合せに対して，それらの述語の引数を無視した命題論理版を対象として，事前解析を行なう．事前解析の精度を上げるためにはより細かいレベルまで（例えばアトムの第 1 引数まで考慮するなど）考えて解析を行なえば良いが，これは事前解析にかかるコストとのトレードオフの問題となる．

例 2.3.1 例 2.2.1の知識ベース P_{ex}（図 2.1参照）を命題論理へ抽象化した知識ベース $\bar{P}_{ex}$を，図 2.3に示す． □

以下では，論理式（あるいはその集合）Pに対し，これを命題論理に抽象化したものを$\bar{P}$で表記する．

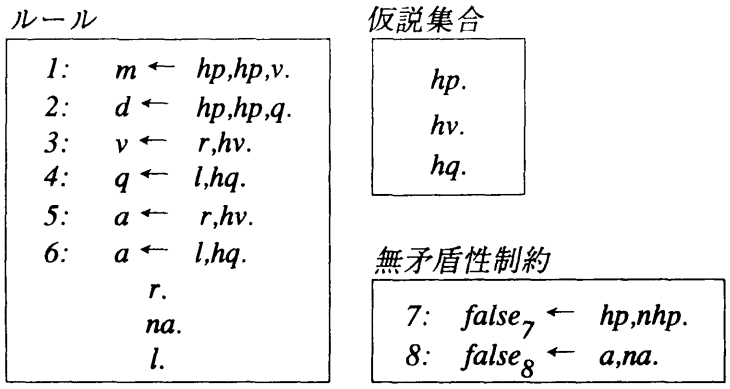


図 2.3: 命題論理に抽象化された知識ベース$\bar{P}_{ex}$

2.3.1 仮説依存関係の解析

本研究で提案する解析を行なうために，OLDT 反駁に以下のような変更を加える．

入力節情報付 OLDT 反駁

- ゴール G の反駁とは，G の反駁木における根から，空節もしくは M 演算子付アトムのみから成るゴール節$\leftarrow \mathsf{M}h_1, \cdots, \mathsf{M}h_k (k \geq 1, h_i$はアトム) でラベル付けられたノード (成功葉と呼ぶ) までの経路であり，成功葉に現れる仮説 h_iの集合 $\{h_1, \cdots, h_k\}$を G の「候補仮説」，この経路において選択された入力節番号の集合を G の「入力節集合」と呼ぶ．
- 解テーブルはエントリの集合で．各エントリは，キーと解リストの組からなる．キーはアトムである．解リストは，対応するキーの候補仮説と，入力節集合の組から成るリストである．

図 2.4に抽象化された知識ベース$\bar{P}_{ex}$におけるゴール$\leftarrow m$ 及び$\leftarrow false_i\ (i = 7, 8)$ に対する OLDT 反駁木を示す．

2.3.2 仮説依存関係による制約節の検査

事実 F及び仮説集合 Hの和集合であるプログラム Pとゴール$\leftarrow O$に対して，それを命題論理に抽象化したものをそれぞれ$\bar{P}$, $\leftarrow \bar{O}$とする．また, Pにおけるゴール$\leftarrow O$に対する OLDT 反駁木（$P \cup \{\leftarrow O\}$の OLDT 反駁木と呼ぶ）において，Oのある候補

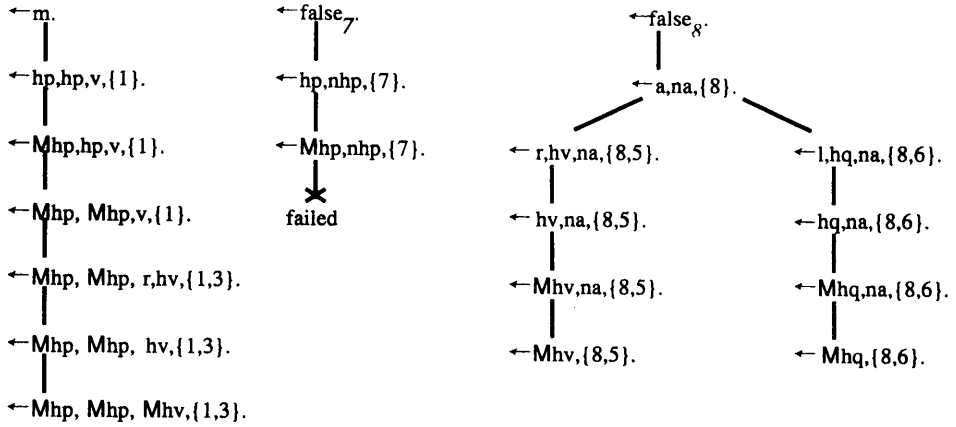


図 2.4: 知識ベース$\bar{P}_{ex}$の解析結果

仮説をH_Oとする（つまり$F \cup H_O \vdash O$である）．同様にゴール$\leftarrow false$に対しても，Pにおける$false$のある候補仮説をH_{false}とする．この時，組(H_O, H_{false})で，H_{false}がH_Oの部分集合となるようなものを，Oに対する「矛盾仮説対」と呼ぶことにする（$(\bar{H}_{\bar{O}}, \bar{H}_{false})$に対しても同様に定義する）．このように定義すると，次の命題が成り立つことが分かる（証明は付録に示す）．

命題 2.3.1 Pをプログラム，Pに含まれる事実をF，仮説集合をHとし，Oを観測とする．Oに対する候補仮説H_Oが条件(AR2)を満たしていないとする．このとき，(i) $P \cup \{\leftarrow false\}$の反駁木において，組$(H_O, H_{false})$が$O$に対する矛盾仮説対となるような$H_{false}$を$false$の候補仮説として持つ成功葉が必ず存在し，(ii) (i)が成り立つならば，$(\bar{H}_{\bar{O}}, \bar{H}_{false})$も$\bar{O}$に対する矛盾仮説対となる． ■

言い換えると，$P \cup \{\leftarrow O\}$の仮説推論では，候補仮説H_Oに対する無矛盾性の検査においては，まず命題論理版において，すべての$\bar{H}_{\bar{O}}$に対して$(\bar{H}_{\bar{O}}, \bar{H}_{false})$が矛盾仮説対であるか否かを調べ，もしそれが矛盾仮説対でないことが事前にわかるならば，このような候補仮説H_{false}を仮定する導出は行なう必要がないことになる．

例 2.3.2 $\bar{P}_{ex} \cup \{\leftarrow false_8\}$のOLDT反駁木の右側の枝から候補仮説{hq}を得る（図2.4参照）．また，$\bar{P}_{ex} \cup \{\leftarrow m\}$のOLDT反駁木から得られる$\leftarrow m$の候補仮説は{hp,hv}である．従って，組合せ({hp,hv},{hq})は矛盾仮説対ではないので，この枝は，問合

せ "$\leftarrow m$" に関しては無関係な枝である.

また$\bar{P}_{ex} \cup \{\leftarrow false_7\}$の OLDT 反駁木からは，実際の仮説推論において，ゴール $\leftarrow false_7$の OLDT 反駁は行なう必要がないことがわかる．これは，文献 [OI92] の効率化に対応しており，我々の方法は，文献 [OI92] を特別な場合として含んでいる．■

この例から分かるように，命題論理に抽象化した知識ベースに対する OLDT 反駁木から，問合せに無関係な探索を検出することができる．述語論理版の実際の仮説推論において不必要な探索を回避するために，候補仮説に付随する入力節集合を用いる．

例 2.3.3 図 2.4において$\bar{P}_{ex} \cup \{\leftarrow false_8\}$ の OLDT 反駁木の左側の成功葉から得られる入力節集合は$\{8,5\}$である．実際の$P_{ex} \cup \{\leftarrow false_8\}$の OLDT 反駁において，使用する入力節をその節番号が$\{8,5\}$で与えられる節に限定すると，図 2.5に示すように探索空間を絞り込むことができる． □

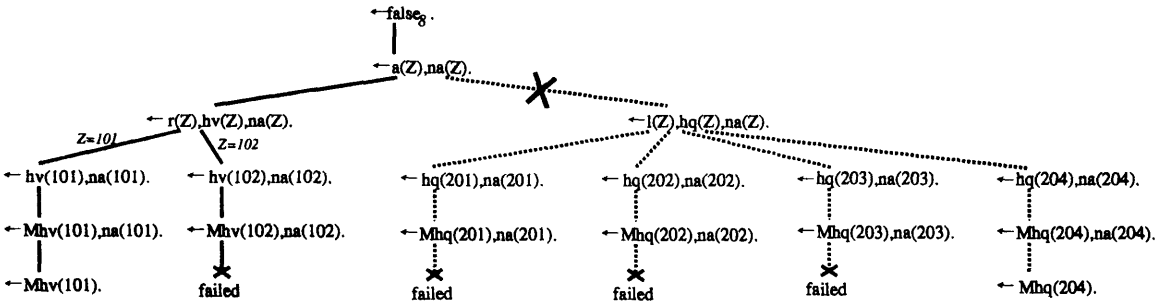


図 2.5: 制約節の評価時における枝刈りの例

2.4 マジックセット法による枝刈りの実現

この節では，2.3節で述べた仮説依存関係の解析と，その解析に基づいた仮説推論を実現するための方法を述べる．本研究では，OLDT 反駁と本質的に等価な計算を実行するマジックセット法[BMSU86]に基づいた前向き推論を用いてこれらを実現する[6]．その理由は，マジックセット法に基づいた前向き推論は，OLDT 反駁などの後向き推論に比べ，大量のデータを対象とする計算（大規模な知識ベースを対象とした問合せに

対する全解探索など）に適しているからである．

通常のマジックセット法は，与えられたホーン節プログラム及び問合せにプログラム変換を適用し，その変換されたプログラムに対する前向き推論が与えられたプログラムに対する後向き推論をシミュレートできるようなプログラムを導出する．変換されたプログラム及び問合せに対して不動点を求める前向き推論は，問合せに対して全解探索を行なう後向き推論に相当する．また，前向き推論の終了条件を変更する[7]ことにより，問合せに対する単解探索にも対応できる．

2.4.1 マジックセット法を用いた仮説依存関係の解析法

まず，OLDT 反駁によるプログラム解析法は，その解析用に変更されたマジックセット法を用いることによって単純な前向き推論で実現できることを例題を用いて説明する．

例 2.4.1 マジックセット法の考え方に基づき変換された例題知識ベース$\bar{P}_{ex}^{magic}$（図2.6参照）を用いれば，単純な前向き推論を行なうだけで問合せと無矛盾性制約との仮説依存関係を求めることができる．

知識ベース$\bar{P_{ex}}$（図2.3参照）において，例えば節番号1のルール$C_1 = m \leftarrow hp, hp, v.$は，知識ベース$\bar{P}_{ex}^{magic}$における3つのルール (1.1), (1.2), (1.3) に変換される．これらのルールの前向き推論による直観的意味は次の通りである：

ルール (1.1), (1.2)： ゴール$\leftarrow m$に対応するアトム$magic_m$が与えられ，かつ，仮説hp, hpを仮定する時，サブゴール$\leftarrow v$を解くために$magic_v$が得られる．この時，サブゴールの導出において得られた候補仮説及び，その導出時に用いられた入力節集合をゴールへ伝播するための述語$cont_{11}(\{hp\}, \{\})$も導かれる．ここで，述語$cont_{11}(\{hp\}, \{\})$の第一引数は，mの候補仮説の部分集合を表し，第二引数は，mの入力節集合の部分集合を表している．

[6] OLDT 反駁とマジックセット法等の対応については，例えば [Sek89],[宮崎 90] を参照されたい．

[7] Oを問合せ，T_Pを直接帰結オペレータ，Iをエルブラン解釈とする．T_Pの不動点計算における終了条件$I = Tp(I)$を$O' \in Tp(I)$へ変更する（ただし，O'はOの例 (instance) とする）．

ルール			
magic_m,hp,hp	→	magic_v.	(1.1)
magic_m,hp,hp	→	cont_{11}({hp},{}).	(1.2)
cont_{11}(H,Cls),v(Hv,Cls_v)	→		
		m(H∪Hv,{1}∪Cls∪Cls_v).	(1.3)
magic_d,hp,hp	→	magic_q.	(2.1)
magic_d,hp,hp	→	cont_{21}({hp},{}).	(2.2)
cont_{21}(H,Cls),q(Hq,Cls_q)	→		
		d(H∪Hq,{2}∪Cls∪Cls_q).	(2.3)
magic_v,r,hv	→	v({hv},{3}).	(3.1)
magic_q,l,hq	→	q({hq},{4}).	(4.1)
magic_s,r,hv	→	s({hv},{5}).	(5.1)
magic_s,l,hq	→	s({hq},{6}).	(6.1)
r. ns. l.			
仮説集合			
hp. hv. hq.			
無矛盾性制約			
$\mathrm{magic\_false}_7$,hp,nhp	→	false_p({hp},{7}).	(7.1)
$\mathrm{magic\_false}_8$	→	magic_s.	(8.1)
$\mathrm{magic\_false}_8$	→	cont_{81}({ }{}).	(8.2)
cont_{81}(H,Cls),s(Hs,Cls_s),ns	→		
		false_8(H∪Hq,{8}∪Cls∪Cls_s).	(8.3)
ゴール節			
	→	magic_m.	(9)
	→	$\mathrm{magic\_false}_7$.	(10)
	→	$\mathrm{magic\_false}_8$.	(11)

図 2.6: マジックセット法によって変換された例題知識ベース $\bar{P}_{ex}^{magic}$

ルール (1.3) : $cont_{11}(H, Cls)$ とともにゴール $\leftarrow v$の解 $v(Hv, Cls_v)$ が得られた時，ゴール $\leftarrow m$ の解 $m(H \cup Hv, \{1\} \cup Cls \cup Cls_v)$ が得られる．ここで，述語 $m(H \cup Hv, \{1\} \cup Cls \cup Cls_v)$ の第一引数は，m の候補仮説を表し，第二引数は，m の入力節集合を表している（述語 $v(Hv, Cls_v)$ についても同様である）．

ルール (1.1) ～ (1.3) に対する前向き推論は，節 C_1に対する後向き推論に対応している．

図 2.6の$\bar{P}_{ex}^{magic}$に対して，前向き推論を行ない，その不動点 $T_{\bar{P}_{ex}^{magic}}^{\uparrow\omega}$を求める．その結果，ゴール (9), (10), (11) に対して以下の結果を得る．

$$T_{\bar{P}_{ex}^{magic}}^{\uparrow\omega} \supseteq \left\{ \begin{array}{l} m(\{hp, hv\}, \{1, 3\}). \\ false_8(\{hv\}, \{8, 5\}). \\ false_8(\{hq\}, \{8, 6\}). \end{array} \right\}$$

例えば，m($\{hp, hv\}$,$\{1, 3\}$). の第一引数は，m の候補仮説，第二引数は，この候補仮説を導出するための入力節集合を示しており，この結果は，OLDT 反駁による$\bar{P}_{ex}$の解析 (図 2.4参照) から得られる結果と等しい． □

2.4.2 入力節制約付マジックセット法

次に，入力節が制限されたOLDT 反駁 (例 2.3.3参照) と同様な計算をマジックセット法に基づいた前向き推論で実現することを考える．このために，通常のマジックセット法を次のように拡張する．

- ゴールに対する解代入と同時に，ゴールを証明するのに必要な仮説の集合を求める．
- マジック変換されたルールを用いた前向き推論は，プログラム解析によって得られた入力節集合によって制御されることにより，無駄な探索空間の枝刈りを行う．

例 2.4.2 知識ベース P_{ex}（図 2.1参照）及び，ゴール$\leftarrow m(b, Y, Z)$, $\leftarrow false_8$ を図 2.7のようにマジック変換する．例えば述語 $magic\_m(X, Y, Z, Cls)$ の第 4 引数は，ゴール m の導出において選択すべき入力節集合を代入するための変数である．ここで，述語 $member(K, Cls)$ は，節番号 Kが集合 Cls の要素であるか否かをチェックする述語である．このようにマジック変換された知識ベース P_{ex}^{magic}に対して，ゴール$\leftarrow false_8$

に対応する節番号 (11) のゴール節を与えて前向き推論を行なえば，図 2.5と同様にしてゴール←$m(b,Y,Z)$の問合せに関する無駄な探索空間を絞り込むことができる．□

ルール	
$magic\_m(X,Y,Z,Cls), hp(X,s1), hp(Y,s2), member(1,Cls)$ $\rightarrow magic\_v(Z,Cls).$	(1.1)
$magic\_m(X,Y,Z,Cls), hp(X,s1), hp(Y,s2), member(1,Cls)$ $\rightarrow cont_{11}([X,Y,Z],\{hp(X,s1), hp(Y,s2)\},Cls).$	(1.2)
$cont_{11}([X,Y,Z],H,Cls), v(Z,Hv,Cv), Cls \supseteq Cv$ $\rightarrow m(X,Y,Z,H \cup Hv,Cls).$	(1.3)
$magic\_d(X,Y,Z,Cls), hp(X,s1), hp(Y,s2), member(2,Cls)$ $\rightarrow magic\_q(Z,Cls).$	(2.1)
$magic\_d(X,Y,Z,Cls), hp(X,s1), hp(Y,s2), member(2,Cls)$ $\rightarrow cont_{21}([X,Y,Z],\{hp(X,s1), hp(Y,s2)\},Cls).$	(2.2)
$cont_{21}([X,Y,Z],H,Cls), q(Z,Hq,Cq), Cls \supseteq Cq$ $\rightarrow d(X,Y,Z,H \cup Hq,Cls).$	(2.3)
$magic\_v(Z,Cls), r(Z), hv(Z), member(3,Cls) \rightarrow v(Z,\{hv(Z)\},Cls).$	(3.1)
$magic\_q(Z,Cls), l(Z), hq(Z), member(4,Cls) \rightarrow q(Z,\{hq(Z)\},Cls).$	(4.1)
$magic\_a(Z,Cls), r(Z), hv(Z), member(5,Cls) \rightarrow a(Z,\{hv(Z)\},Cls).$	(5.1)
$magic\_a(Z,Cls), l(Z), hq(Z), member(6,Cls) \rightarrow a(Z,\{hq(Z)\},Cls).$	(6.1)
r(101). r(102). l(201). l(202). l(203). l(204). na(101). na(204).	
仮説集合	
hp(b,s1). hp(c,s1). hp(e,s2). hp(f,s2). hv(Z). hq(Z).	
無矛盾性制約	
$magic\_false_7([],Cls), hp(X,Y), nhp(X,Y), member(7,Cls)$ $\rightarrow false_7([],\{hp(X,Y)\},Cls).$	(7.1)
$magic\_false_8([],Cls), member(8,Cls) \rightarrow magic\_a(Z,Cls).$	(8.1)
$magic\_false_8([],Cls), member(8,Cls) \rightarrow cont_{81},([],\{\},Cls).$	(8.2)
$cont_{81}([],H,Cls), a(Z,Ha,Ca), na(Z), Cls \supseteq Ca$ $\rightarrow false_8([],H \cup Ha,Cls)$	(8.3)
ゴール節	
$\rightarrow magic\_m(b,Y,Z,\{1,3\}).$	(9)
$\rightarrow magic\_false_8([],\{8,5\}).$	(11)

図 2.7: マジックセット法によって変換された例題知識ベース P_{ex}^{magic}

2.5 実験結果

本章で提案した効率化法の有効性を確認するために，推論時間の比較実験を行なった．例題としては，例 2.2.1の知識ベースを使用した．また，知識ベースに現れる事実の単位節の数を k とし，k をパラメータとして知識ベースの規模を変化させて実験

した.

実験結果を図 2.8に示す. 実験は計算機 SUN4/75 上で言語 SICStus Prolog を用いて行なった. 実線は本章で提案した手法, 破線は文献 [OI92] による手法の結果を示している. 知識ベースの規模に関わらず, 事前にプログラム解析を行なうことにより, 問合せ "$\leftarrow m(b,Y,Z)$" では約 45%, 問合せ "$\leftarrow d(b,Y,Z)$" では約 80%程度に, 推論時間がそれぞれ短縮されている.

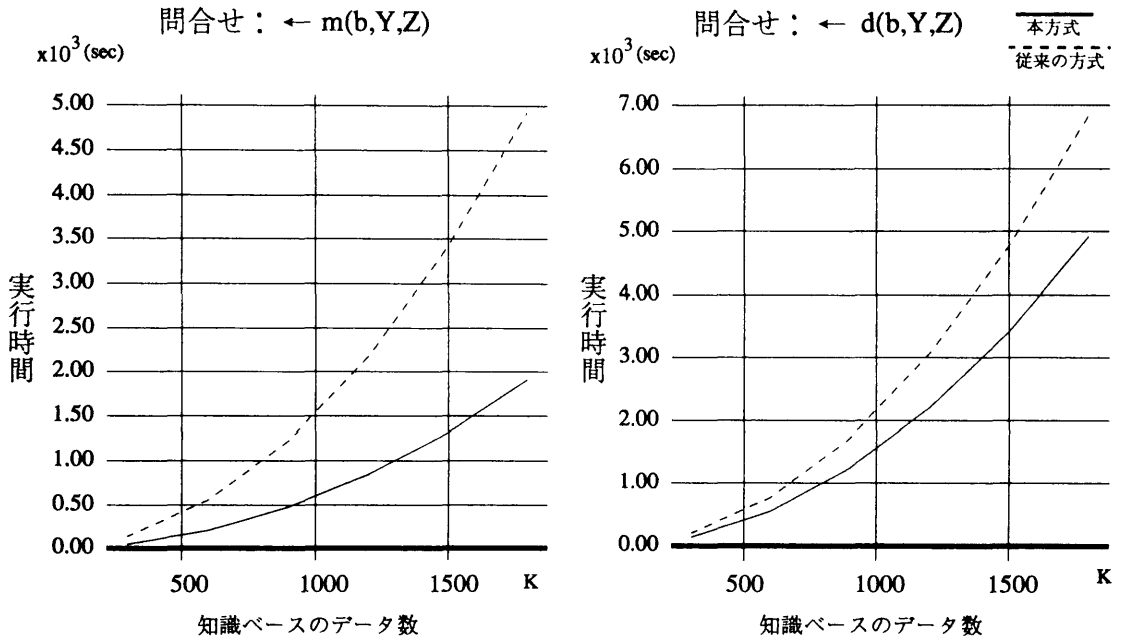


図 2.8: 実験結果

また, 事前解析自身に要する時間は, 命題論理のレベルに抽象化してプログラム解析を行なっているため, 知識ベースの規模 k が大きい場合には, 実際の仮説推論に要する時間の約 1%以下であり, 無視できる程度であった.

2.6 まとめ

一階述語論理を用いた仮説推論の高速化に対して, プログラム解析に基づく無矛盾性の検査の効率化の提案を行なった. 本章で提案したプログラム解析法は, ホーン節述語論理で表現された知識ベースを命題論理のレベルに抽象化することにより, 与え

られた問合せと知識ベースの論理的制約に対して，実際の推論において評価すべき探索空間を，近似的に求めるものである．この解析法を用いて，仮説推論における制約節の評価時において，無駄な探索空間の絞り込みを行なうような，また，マジックセット法に基づきつつ，それをプログラム解析結果を利用できるような形に拡張した方式により仮説推論システムを実装し，その有効性を確認した．本章で述べたマジックセット法に基づく推論システムの実装法は，それが前向き推論の一制御法を提案している点（例えば文献 [RSS92]）から，演繹データベースの分野においても興味深いものであると思われる．

第3章

コストに基づく仮説推論における推論制御の最適化

3.1 はじめに

仮説推論をPooleの定式化による論理的枠組で捉えることにより，論理プログラミングや演繹データベースの分野における様々な問合せ処理技術を用いて仮説推論が効率的に実現されている（例えば，[OI92], [Sti91]）．しかしながら，仮説推論が演繹ベータベースにおける問い合わせ処理と大きく異なる点として，推論によって求められた観測の説明が無矛盾性の検査を受けなければならないことがあげられる．そこで本研究では，無矛盾性の検査の手続きが仮説推論の計算量を増大させる要因となっていることに着目し，プログラム解析に基づく無矛盾性の検査の効率化を前章で提案し，一階述語論理を対象とした仮説推論の効率化を実現した [加藤 94], [KSI93b].

さらに，本章でも仮説推論の論理的枠組が持つ特性に着目し，その推論機能の高速化を試みる．

Pooleの定式化からもわかるように，仮説推論では一般に，観測を説明する仮説集合は複数存在する．しかしながら，計画·診断問題等において見られるように，ユーザが要求する解はすべての説明ではなく，むしろ最も好ましい説明であることが多い．例えば，計画問題においては，与えられた目標を達成するすべての手順よりも，それらの中で最も安価（あるいは最速な）手順がしばしば要求される．そこで文献 [Poo93],[CS90],[Poo92],[近藤 94] などの研究では，より適切な説明を得るために，仮説に好ましさの基準（確率，コスト等）を与える手法が報告されている．上記のよう

な要求は，与えられた観測をより適切に説明するような仮説をどのように選択するかという「合理的な仮説の選択」の問題として認識されている[井上92]．

そこで本章では，一階述語論理ホーン節で表現された知識ベースを対象に，与えられた観測を説明する最小コストの仮説集合を求めることにより最適な説明を得る仮説推論について考える[1]．各仮説に好ましさの基準が与えられることによって，最適な説明を求める問題は，ヒューリスティック探索の問題に帰着される．一方で，ヒューリスティックな評価関数を最も効率的に利用できる最適解探索法として広く知られている[DP85]ものに A^*アルゴリズム[Nil80]がある．従って本章では，仮説推論を実現する推論処理技術に対して A^*アルゴリズムの持つ探索制御技術を導入する．さらに，本研究で提案する推論方法の探索制御能力を決定づけるヒューリスティック評価関数の導出方法を提案する．

本章の構成は以下の通りである．まず **3.2節**でコストに基づく仮説推論の枠組について述べ，関連研究について言及する．次に **3.3節**で A* アルゴリズムに基づき最適な説明を求める効率的な推論方法を後向き推論の枠組で説明し，**3.4節**でこの推論方法の前向き推論による実装について述べる．そして，**3.5節**で実験結果について説明する．

3.2 コストに基づく仮説推論

この節では，Poole による定式化[Poo88]に基づきつつ，本章で用いる仮説推論の枠組について説明し，この分野での関連研究について述べる．

3.2.1 コストに基づく仮説推論の定式化

本章では，仮説の選択の基準として知識ベースに含まれる各仮説に重み（コスト）が与えられている場合を対象に，与えられた観測に対して最適な説明を求める仮説推論を考える．各仮説に与えられる重みは正の数とし，説明が最適であるとは説明のコ

[1] 本研究は文献 [KSI94a],[KSI94b] の報告をもとにして本論文にまとめたものである．

スト[2]が最小であることとする.

定義 3.2.1 Fをホーン節集合(「事実」と呼ぶ), Hを基底単位節の集合(「仮説集合」と呼ぶ)とする. また, Oを存在束縛されたアトムの連言(「観測」, または単に「ゴール」と呼ぶ)とする. このとき, Oの$F \cup H$による最適な説明とは, 以下の条件を満足するような, Hの要素の代入例から成るマルチセットEを求めることである.

$$F \cup E \vdash O \qquad (F \cup E\text{から}\ O\text{が証明される}) \qquad \text{(CBA1)}$$

$$F \cup E \nvdash false \qquad (F \cup E\text{は無矛盾である}) \qquad \text{(CBA2)}$$

$$cost(E) \leq cost(D) \text{ for all } D : F \cup D \vdash O, F \cup D \nvdash false$$

(条件 CBA1,CBA2を満たすマルチセットの中でEのコストが最小) (CBA3)

ここで, Fは常に成り立つ知識として扱われる. 一方で, Hの代入例からなる部分集合はFと矛盾する可能性がある. また, FとHの和集合をプログラムと呼び, Pで表記する. なお, 本章では説明Eはマルチセットとして扱われる. ■

注 3.2.1 本研究で提案するシステムの一応用例である計画問題においては, 例えば, 観測の説明Eは与えられた目標状態を達成する作業手順として表現される[3]. このような場合には, Eをマルチセットとして扱い, Eの要素について重複を許したり, 順序を考えるほうが好ましいと考えられる. 観測の説明を仮説の集合として表現する定式化もあるが(例えば文献 [Poo88], [井上 92]), ここで提案する推論方法に少しの変更を加えることによってこのような場合にも対応できる. ■

定義 3.2.2 Fの中に存在する負節を「一貫性制約節」(あるいは単に「制約式」)と呼ぶ. 制約節は, 論理式 $false \leftarrow A_1, \cdots, A_n$ で表現される. ただし, A_k $(1 \leq k \leq n;\ n \geq 1)$ はアトムであり, $false$ は「矛盾」を表す. ■

例 3.2.1 図3.1に示す簡単な例題プログラムを考える. P_{ex}に対して, 問合せ "$\leftarrow p(X,Y)$" を与える. 同図の知識ベースが事実と仮説の和集合 $F \cup H$ であり, アトム $p(X,Y)$

[2] 観測の説明のコストとは, 観測の説明に現れる仮説の重みの和で与えられる.

[3] 仮説推論を用いて計画問題を解く場合には, 状態空間を F, 目標を達成し得る個々の操作(部分計画)を Hとして問題が知識表現され, Oとして目標状態が与えられる.

が観測 Oである.

事実	仮説	コスト	仮説	コスト
p(X,Y) ← e(X),s(Y).	q(1).	3	q(2).	2
p(X,Y) ← f(X),t(Y).	r(1).	5	r(2).	2
e(X) ← q(X).	s(1).	6	s(2).	4
f(X) ← r(X).	t(1).	4	t(2).	2
false ← s(X),t(X).				

図 3.1: 例題プログラム P_{ex}

■

図 3.2に例 3.2.1に対する SLD 反駁 [Llo84] を用いた仮説推論の実行例を示す. 同図においてゴール節に現れる M なるオペレータが付いたアトム "MA" は, サブゴール ← A_0（ここでアトム A は A_0のある代入例）を解く際に, 仮説を入力節として導出を行なったこと（すなわち, A に対して仮説が立てられたこと）を示すためのマークである[加藤 94].

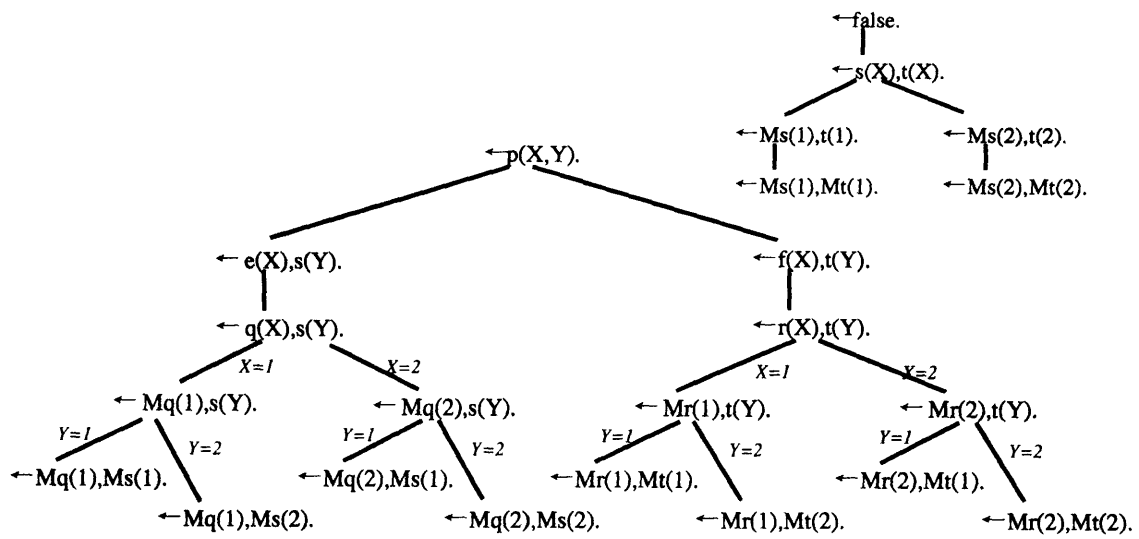


図 3.2: 仮説推論の実行例

ここで, 図 3.2の$P_{ex} \cup \{\leftarrow false\}$の SLD 反駁木から, 成功葉に現れる仮説のマルチセット$\{s(1), t(1)\}$または$\{s(2), t(2)\}$を仮定することは知識ベースに対して矛盾を起こすことがわかる（このような仮説のマルチセットを「矛盾仮説」と呼ぶ）. また, 同図の SLD 反駁木より, 左から 3 番目の成功葉から得られる仮説のマルチセット $\{r(2), t(2)\}$

のコストが反駁木の任意の成功葉から得られる仮説のマルチセットのコストにおいて最小であるので，これを説明としてゴール$\leftarrow p(X,Y)$の最適解$\{X=2,Y=2\}$が得られる．

しかしながら，一つの最適解を得るために全解を求めるのは無駄な探索空間が大きく極めて非効率である．従って本研究では，最適な説明の導出には無関係な探索を回避して効率的に最適解を得るために，SLD 反駁に対して最適解探索アルゴリズムが持つ探索制御技術を導入する．さらに，本研究で提案する推論方法の探索制御能力を決定づけるヒューリスティック評価関数の導出方法を提案する．

なお，本研究では，反駁木に現れるゴール内のアトムの選択規則は最左優先とする．

3.2.2 関連研究

仮説推論を前節のように定義し，各仮説に好ましさの基準が与えられることによって，最適な説明を求める問題は最適解探索の問題に帰着される．これまでに提案されている最適解を保証する仮説推論システムでは，主に最良優先探索が用いられている（例えば文献 [CS90], [Poo92], [近藤 94] など）．

文献 [CS90],[Poo92] では，最良優先探索の手法を用いることにより最適な説明を効率的に求める仮説推論が提案されている．しかしながら，効果的なヒューリスティック評価関数の求め方については提案されていない．

本研究では，仮説推論を実現する推論処理技術に対して A* アルゴリズム[Nil80]の持つ探索制御技術を導入する．さらに，本仮説推論方法の探索制御能力を決定づけるヒューリスティック評価関数の導出方法を提案する．

文献 [CH91] では，実行可能条件を満たすヒューリスティック評価関数の導出方法が提案されている．しかしながら，その導出法は命題論理で表現されたプログラムを対象に説明されており，述語論理で表現されたプログラムに対する導出方法については言及されていない．プログラムが述語論理で表現されている場合には，本論文で述べるような何らかの抽象化を行なわない限り導出のオーバーヘッドは無視できないもの

と思われる．さらにその方法を，再帰的に定義された一階述語論理プログラムに対応できるように拡張するのは難しいと考える．

本研究で提案するヒューリスティック関数の導出法は次節で提案する推論アルゴリズムを直接用いるので，システムの構成が簡単であるのみならず，任意の（再帰的に定義された）一階述語論理プログラムにも対応できる．

また，文献 [近藤 94] では，ビーム探索·分岐限定法および最良優先探索を用いて効率的に最適解を求める仮説推論を提案している．文献 [近藤 94] の研究については 3.3.2 節で述べる．

3.3 A*アルゴリズムに基づく探索制御

本節では，A^*アルゴリズムの持つ探索制御技術を導入することにより効率的な推論処理方法を提案する．ここでは，簡単のため，SLD 反駁を用いて本手法について説明する．

なお，ここでは，知識ベースに現れる制約式については$\leftarrow false$に対する反駁を行ない，矛盾仮説のマルチセットをあらかじめに求めておくことにする．[4]

3.3.1 A*アルゴリズムを用いた SLD 反駁

定義 3.3.1 Pをプログラム，Oを与えられた観測とする．また，Pにおけるゴール$\leftarrow O$に対する SLD 反駁木（$P \cup \{\leftarrow O\}$の SLD 反駁木と呼ぶ）における任意のゴール節 $g = \leftarrow \mathsf{M}L_1, \cdots, \mathsf{M}L_i, A_{i+1}, \cdots, A_n$について，$c(g)$ を反駁木の根から gに至る導出におけるコスト，$\hat{h}(g)$ を gから成功葉に至る最適導出におけるコストの予測値（$h(g)$ は実際のコストの値）とする．このとき，最適解探索における gの評価値 $f(g)$ は以下の式で与えられる．

$$f(g) = c(g) + \hat{h}(g)$$

[4] 一般に，$\leftarrow false$に対する反駁の計算コストは大きくなり得るが，そのようなときは，プログラム解析[KSI93b]を用いることにより無駄な計算を極力回避する

$c(g)$ = マルチセット$\{L_1, \cdots, L_i\}$のコスト

$h(g)$ = マルチセット$\bigcup_{j=i+1}^{n} A_j$に対する最適な説明 のコスト

$0 \leq \hat{h}(g) \leq h(g)$

本研究では，A_jの説明が存在しないとき（つまり，$\leftarrow A_j$の反駁が存在しないとき）には，$h(g) = \infty$ とする． ■

定義 3.3.2 A*アルゴリズムを用いたSLD反駁

Pをプログラム，Oを与えられた観測とする．また，$OPEN_i$, $CLOSED_i$, $SELECTED$ および SG をゴールの集合とする．$OPEN_i$および$CLOSED_i$は，まだ展開されていないゴールの中で最適解を求めるために有望なゴールおよびもうすでに展開されたゴールの集合をそれぞれ示している．また，$SELECTED$ は現時点で最も有望なゴールを要素に持つ集合である．アルゴリズムを図3.3に示す．

ここで，ゴール$A = \leftarrow a_1, \cdots, a_n$, $B = \leftarrow b_1, \cdots, b_m$ が$A =^* B$とは，AとBがvariant[5]であることとし，$A \supset^* B$とは，$A' = \leftarrow a'_1, \cdots, a'_m$（ここで，$^\forall a'_j\ (1 \leq j \leq m) \in A'$は$A$に現れるアトムであるとする）と$B$がvariantであることとする． ■

任意のSLD反駁木において，すべてのゴール節gに対し条件$\hat{h}(g) \leq h(g)$（実行可能条件と呼ぶ）が成立する時，本アルゴリズムは実行可能であり，最適なアルゴリズムである（A*の実行可能性[Nil80]および最適性[DP85]より明らか）．また，本アルゴリズムは$OPEN_i$の更新処理（図3.3,9-19行）においてループ・チェック機能を備えており，再帰構造を持つプログラムにも対応する．しかしながら，ヒューリスティック探索の持つ性質として知られているように，本アルゴリズムの性能はgから成功葉へ至る導出のコストの予測値（ヒューリスティック評価関数）$\hat{h}(g)$に大きく依存する．

従って本研究では，実行可能条件を満たし，かつ実際のコストになるべく近いようなgのコストの予測値$\hat{h}(g)$を求める方法を次節で提案する．

[5] 論理式E,Fがvariantとは，E,Fに対して次の単一化が成り立つことを言う（ただしθ, σはともに最汎単一化代入(mgu)）．$E = F\theta$ かつ $F = E\sigma$

A*アルゴリズムを用いたSLD反駁

入力: プログラム P, 観測 O

出力: 解 Ans: (解代入例, 最適な説明)（成功裏）または $false$（失敗裏）

```
1  begin
2    OPEN_0 := {←O}; CLOSED_0 := φ; i := 0;
3    repeat
4      SELECTED := {g_min ∈ OPEN_i | for all g ∈ OPEN_i : f(g_min) ≤ f(g)};
5      各← g_min ∈ SELECTEDに対して Pにおける一段階の SLD 導出で得られる
       サブゴールをすべて求め†, これらの集合を SG とする;
6      CLOSED_{i+1} := CLOSED_i ∪ SELECTED;
7      SG について無矛盾性の検査††を行ない Pと矛盾するゴールを SG から削除する;
8      OPEN' := OPEN_i \ SELECTED;
9      for each sg ∈ SG
10       if sg ⊅* ∃g ∈ OPEN' ∪ CLOSED_{i+1} then
11         if sg =* ∃g ∈ OPEN' then
12           begin
13             if f(sg) < f(g) then OPEN' := OPEN' \ {g} ∪ {sg}
14           end
15         else if sg =* ∃g ∈ CLOSED_{i+1} then
16           begin
17             if f(sg) < f(g) then OPEN' := OPEN' ∪ {sg}
18           end
19         else OPEN' := OPEN' ∪ {sg};
20     OPEN_{i+1} := OPEN';
21     i := i + 1;
22   until (∃g_min ∈ SELECTED は成功葉である) or (OPEN_i = φ);
23   if ∃g_min ∈ SELECTED は成功葉である then
         Ans := (解代入例, 最適な説明);    % (成功裏に終了)
24   if OPEN = φ then Ans := false;        % (失敗裏に終了)
25 end.
```

† この時点ではサブゴールの状態は未定 (unspecified) である.

†† サブゴールに現れる仮説が矛盾仮説を含むかどうかで検査する.

図 3.3: A*アルゴリズムを用いたSLD反駁

3.3.2 ヒューリスティック評価関数$\hat{h}(g)$の導出方法

本節では，本アルゴリズムの探索制御能力を決定づけるヒューリスティック評価関数$\hat{h}(g)$の算出方法について述べる．

AIにおける探索アルゴリズムの分野では，与えられた問題に対して実行可能なヒューリスティクスは抽象化された問題から生成することができ，さらに，このようにして得られたヒューリスティクスが効果的なものとなるには，抽象化された問題が，それを効率的に解くことができる範囲で元の問題の近似である必要があることが知られている（例えば，文献 [Pri93]）．

従って本研究では，述語論理式で表現された知識ベース並びに問合せに対して，その近似としてそれらの述語の引数を無視した命題論理版を対象として，事前解析を行なう．事前解析の精度を上げるためにはより細かいレベルまで（例えばアトムの第1引数まで考慮するなど）考えて解析を行なえば良いが，これは事前解析にかかるコストとのトレードオフの問題となる．

Pをプログラム，Pを命題論理に抽象化したものを$\bar{P}$とする（以下，論理式（あるいはその集合）Fに対し，これを命題論理に抽象化したものを$\bar{F}$ で表記する）[6]．

まず，$\bar{P}$に現れるアトム$\bar{A}$ついて，$\bar{P} \cup \{\leftarrow \bar{A}\}$に対する A* に基づく SLD 反駁（定義 3.3.2参照）を実行し，$\bar{A}$と$\bar{A}$ の最適な説明 $H_{\bar{A}}$の組 $(\bar{A}, H_{\bar{A}})$ を求める[7]．ここでは各反駁木における任意のゴール節$\bar{g}$対して$\hat{h}(\bar{g}) = 0$ とする．次に，それらの組から得られる情報を用いてヒューリスティック評価関数$\hat{h}(g)$ を定義する．

例 3.3.1 図 3.1のプログラム P_{ex}を命題論理に抽象化したプログラム$\bar{P}_{ex}$を図 3.4に示す．ここで，例えば仮説 qのコスト 2 は P_{ex}に含まれる仮説 $q(1), q(2)$ のコストの最小値で与えられている．

$\bar{P}_{ex}$に対してゴール節$\leftarrow p, \leftarrow f, \leftarrow t, \leftarrow e, \leftarrow s, \leftarrow q, \leftarrow r$をそれぞれ $OPEN$ の初期集合の

[6] Pに含まれる仮説 Eに対して，$\bar{E}$のコストは$\bar{L} = \bar{E}$となるような仮説 $L \in P$のコストの最小値で与えられる．

[7] アトム$\bar{A}$の最適な説明 $H_{\bar{A}}$が存在しない（つまり$\leftarrow \bar{A}$の反駁が存在しない）場合には，そのコストが ∞ となるような仮の仮説のマルチセット H'_Aを用いて，$(\bar{A}, H'_{\bar{A}})$ とする．

事実	仮説	コスト
p ← e,s.	q.	2
p ← f,t.	r.	2
e ← q.	s.	4
f ← r.	t.	2
$false$ ← s,t.		

図 3.4: 抽象化されたプログラム$\bar{P}_{ex}$

要素として定義3.3.2のアルゴリズムを実行する．その結果，各反駁木の成功葉よりアトムとその最適な説明の組，$(p,\{r,t\}),(e,\{q\}),(f,\{r\}),(q,\{q\}),(r,\{r\}),(s,\{s\}),(t,\{t\})$.を得る． ■

定義 3.3.3 Pをプログラム，Oを与えられた観測，AをPに含まれるアトムとし，H_AをAの最適な説明を表す仮説のマルチセットとする．

$P\cup\{\leftarrow O\}$のSLD反駁木に現れるゴール節$g=\leftarrow \mathrm{M}L_1,\cdots,\mathrm{M}L_i,A_{i+1},\cdots,A_n$に対して，$g$からある成功葉へ至る最適導出のコストの予測値$\hat{h}(g)$は以下の計算によって求められる．

$$\begin{aligned}&\hat{h}(g)=h(\bar{g})=cost(H_{\bar{g}})\\&H_{\bar{g}}=\bigcup_{j=i+1}^{n}H_{\bar{A}_j}\end{aligned}\qquad(*)$$

ここで，$cost(H)$は仮説のマルチセットHに含まれる仮説のコストの和を表すとする．

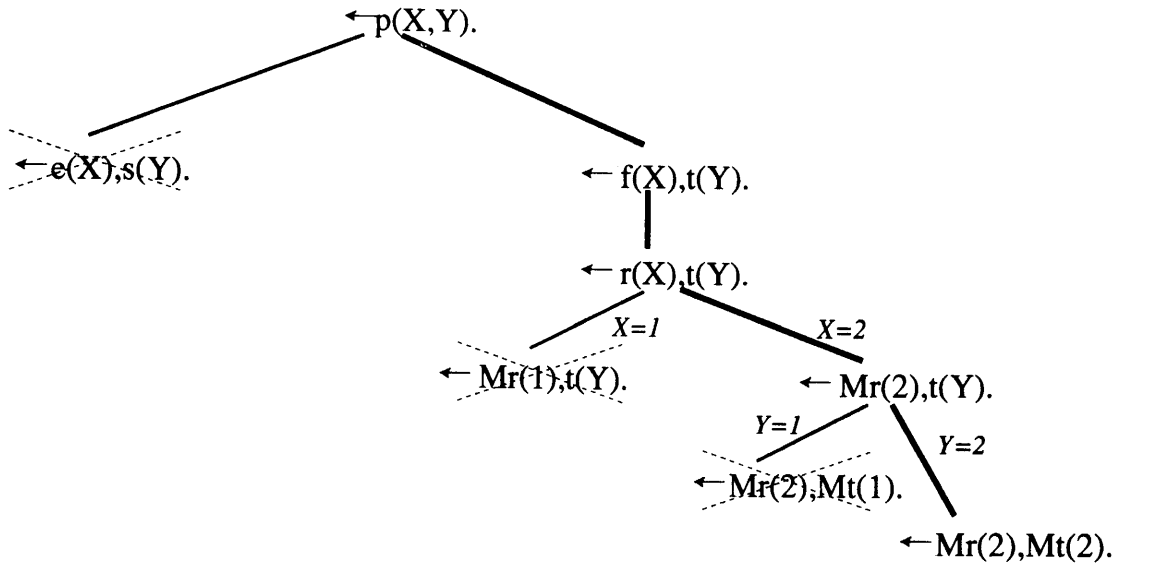
■

このとき，次の性質が成立することが容易にわかる．

定理 3.3.1 式$(*)$から得られる予測値$\hat{h}(g)$は本アルゴリズムの実行可能条件$\hat{h}(g)\leq h(g)$を満たす（証明は付録Bに示す）． ■

このようにして求められたヒューリスティック評価関数を用いることにより，本アルゴリズムは，実行可能性を保持しつつ，効率的な反駁を行なうことが可能となる．

例 3.3.2 図3.5に$P_{ex}\cup\{\leftarrow p(X,Y)\}$に対するA*アルゴリズムを用いたSLD反駁の実行例を示す．同図の反駁木の成功葉から得られる仮説のマルチセット$\{r(2),t(2)\}$は問合せ$p(X,Y)$の最適解$p(2,2)$の最良な説明となる．



初期状態 $OPEN_0$= {< ←p(X,Y). , 0, 4 >}
反復 1 $OPEN_1$= {< ←f(X),t(Y). , 0, 4 >, < ←e(X),s(Y). , 0, 6 >}
反復 2 $OPEN_2$= {< ←r(X),t(Y). , 0, 4 >, < ←e(X),s(Y). , 0, 6 >}
反復 3 $OPEN_3$= {< ←Mr(2),t(Y). , 2, 2 >, < ←e(X),s(Y). , 0, 6 >, < ←Mr(1),t(Y)., 5, 2 >}
反復 4 $OPEN_4$= {< ←Mr(2),Mt(2). , 4, 0 >, < ←Mr(2),Mt(1). , 6, 0 >, < ←e(X),s(Y). , 0, 6 >, < ←Mr(1),t(Y). , 5, 2 >}

$CLOSED_0$= ϕ
$CLOSED_1$= {←p(X,Y).}
$CLOSED_2$= {←p(X,Y). , ←f(X),t(Y).}
$CLOSED_3$= {←p(X,Y). , ←f(X),t(Y). , ←r(X),t(Y).}
$CLOSED_4$= {←p(X,Y). , ←f(X),t(Y). , ←Mr(2),t(Y).}

図 3.5: $P_{ex} \cup \{\leftarrow p(X,Y)\}$の SLD 反駁木における A*による枝刈り

同図において，⋙印が付けられたゴールは，そのゴールに対する展開が枝刈りによってそれ以上行なわれないことを示している．また，図 3.2の反駁木に比べて枝の数および M 演算子付アトムの数が少ないことから，仮説推論の過程において真と仮定される仮説の数が制限され,「合理的な仮説の選択」が実現されていることがわかる．このように，推論において生成されるゴールの数を制限することができ，仮説推論の探索空間を絞り込むことが可能となる． ■

文献 [近藤 94] では，ビーム探索·分岐限定法および最良優先探索の考えを用いて効率的に最適解を求める仮説推論システムが提案されている．探索を制御するパラメータとして，固定的なしきい値および動的に変化する暫定値を用い，さらに最小コストの説明を効率的に求めるために各仮説の組合せのコストの下界値を考えている．その結果，文献 [近藤 94] では仮説の生成および合成の回数を制限することにより，仮説推論の探索空間を絞り込んでいる．

しかしながら，コストの下界値に関する情報が十分に活かされず，効果的なヒュー

リスティクスが得られない場合がある．本研究では，例3.2.1の知識ベース P_{ex} を用いて観測 $p(X,Y)$ の最適な説明を求める仮説推論における探索空間について本手法と文献 [近藤 94] の推論方式との比較実験を行った．表3.1に比較結果を示す．同表から仮

表3.1: 探索空間の比較結果

	仮説生成数	仮説の合成回数
全解探索	8	8
本手法	4	2
文献11) の手法		(θ =しきい値)
$\theta \geq 9$	8	5
$\theta = 7, 8$	8	4
$\theta = 6$	8	3
$\theta = 5$	7	1
$\theta = 4$	6	1
$\theta \leq 3$	—	—

説の生成数についてはしきい値θの値に関わらず本手法の方が最適な説明に無関係な仮説の生成を回避でき，仮説の合成回数についてもθの値によっては本手法の方が無駄な合成を回避していることがわかる．文献 [近藤 94] の効率化手法はθの値に大きく依存する．与えられた例題に対して何らかの解析を行わない限りθの値を適切に求めることは困難であると考える（例3.2.1の問題では$\theta \leq 3$ の下では解を求めることができない）．本推論システムでは，与えられた知識ベース並びに問合せに対して命題論理のレベルで事前解析を行い，実際の仮説推論で用いられる探索制御のための情報を自動に抽出している．以上の結果より，本研究で提案されたヒューリスティック評価関数が本アルゴリズムの探索制御能力を高め，その結果，仮説推論の探索空間を効果的に絞り込んでいることがわかる．

3.4 前向き推論を用いた実装

本節では，3.3節で述べた手法に基づいて，コストに基づく効率的な仮説推論を実現するための方法を述べる．本研究では，SLD 反駁と本質的に等価な計算を実行する前向き推論を用いてこれを実現する．その理由は，前向き推論は，SLD 反駁などの後向

き推論に比べ，大量のデータを対象とする計算に適しており，大規模データベースを扱う問合せ処理システムにおいて多く用いられるからである．本研究では，例 3.2.1 の知識ベース P_{ex}に含まれる仮説の数を変化させて後向き推論と前向き推論の間で推論時間の比較を行った．

図 3.6に例 3.2.1の知識ベース P_{ex}を用いた推論時間の比較結果を示す．P_{ex}に含まれる仮説の数を m とし m を変化させて比較を行った．同表から知識ベースの規模が大きな場合には，前向き推論の方が後向き推論に比べて 5 倍程度速く，有効な推論方法となることがわかる．

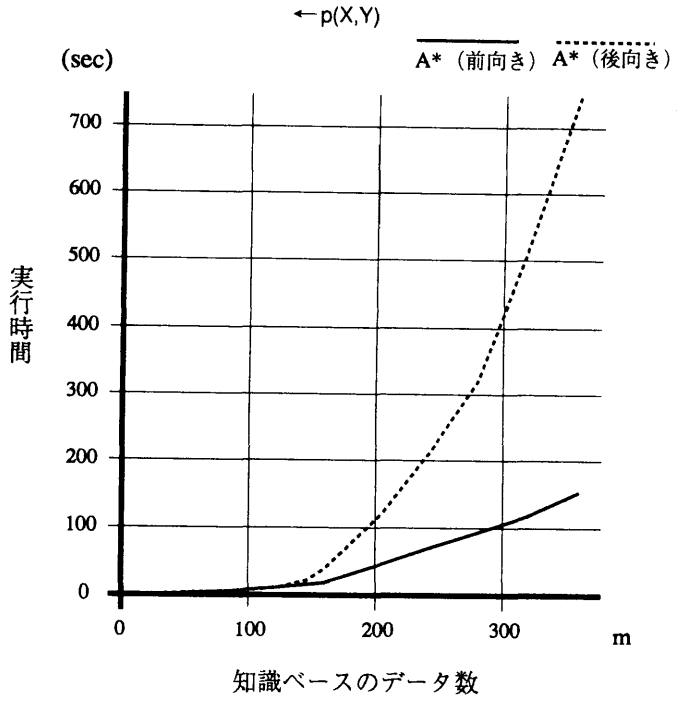


図 3.6: 推論時間の比較

まず，3.3節で提案した推論アルゴリズムを前向き推論で実現するためにプログラム変換法を提案する．本変換法によって変換されたプログラムに対する前向き計算は，幅優先の SLD 反駁をシミュレートしつつ，導出されたゴールの評価値を算出する．次に，A*アルゴリズムで行なわれている *OPEN* , *CLOSED* , *SELECTED* の更新手続き (定義 3.3.2参照) を実現するプログラムを提案し，A*を用いた SLD 反駁と実質的に等価な計算を実現する前向き計算手続きを定義する．

3.4.1 前向き推論のためのプログラム変換

本節では，プログラムに適切な変換を施し，その変換されたプログラムを前向き計算によって処理すると，それが元のプログラムの幅優先のSLD反駁をシミュレートしつつ，導出されたゴールの評価値を算出できることを，図3.1の例題知識ベース P_{ex} を用いて説明する．

例 3.4.1 図3.7に変換されたプログラム P_{ex}^{tr} を示す．ここで，P_{ex}^{tr} に現れる述語 $goal(M, G, V, H, State)$ は $P_{ex} \cup \{\leftarrow p(X,Y)\}$ のSLD反駁木に現れるゴール g に対応しており，述語の第1引数 M は反駁木の根から g に至る導出において生成された仮説のマルチセット，第2引数 G は g に現れるアトムのリスト，第3引数 V は問合せ(ゴール)の解代入を蓄えるための変数をそれぞれ示している．また，第4引数 H は g の評価に関する情報を蓄えており，$cost(H)$ は $\hat{h}(g)$ の値をとる（定義3.3.1参照）．なお，第5引数 $State$ は g の状態を表す[8]ものであり，$open, closed, selected, unsp$ のいずれかの値をとる．また，P_{ex}^{tr} に現れる述語 $\underline{c}(A, H_A)$ はアトム A の最適な説明が H_A であることを示す．

例えば，プログラム P_{ex}^{tr} の節番号1のルールは，後向き推論における確定節 $p(X,Y) \leftarrow e(X), s(Y).$ に対応しており，次の意味である．

> ゴールの状態が $selected$ であるゴールの最左アトムが $p(X,Y)$ と単一化可能であり，かつ，反駁木の根（初期ゴール）からそのゴールへ至る導出のコストが $cost(M)$，そのゴールから成功葉へ至る最適導出のコストの予測値が $cost(H)$ であり，アトム p, e, f の最適な説明がそれぞれ Hp, He, Hf であるとき，ゴールの先頭のアトム $p(X,Y)$ を $e(X), f(X)$ に置き換えたサブゴールを生成する．このとき，反駁木の根からサブゴールへ至る導出のコストは $cost(M)$ であり，サブゴールから成功葉へ至る最適導出のコストの予測値は $cost(H \cup Hc \cup Hs \setminus Hp)$ となる．また，サブゴールの状態は $unsp$（未定, $unspecified$）である．

[8] 例えばゴール節 g の状態が $open$ とは，定義3.3.2のアルゴリズムにおいて g が集合 $OPEN$ の要素となることを意味する．

事実	$goal(M,[p(X,Y)\|G],V,H,selected),\underline{c}(p,Hp),\underline{c}(e,He),\underline{c}(s,Hs)$	
	$\rightarrow goal(M,[e(X),s(Y)\|G],V,H\cup He\cup Hs\setminus Hp,unsp).$	(1)
	$goal(M,[p(X,Y)\|G],V,H,selected),\underline{c}(p,Hp),\underline{c}(f,Hf),\underline{c}(t,Ht)$	
	$\rightarrow goal(M,[f(X),t(Y)\|G],V,H\cup Hf\cup Ht\setminus Hp,unsp).$	(2)
	$goal(M,[e(X)\|G],V,H,selected),\underline{c}(e,He),\underline{c}(q,Hq)$	
	$\rightarrow goal(M,[q(X)\|G],V,H\cup Hq\setminus He,unsp).$	(3)
	$goal(M,[f(X)\|G],V,H,selected),\underline{c}(f,Hf),\underline{c}(r,Hr)$	
	$\rightarrow goal(M,[r(X)\|G],V,H\cup Hr\setminus Hf,unsp).$	(4)
仮説	$goal(M,[q(1)\|G],V,H,selected),\underline{c}(q,Hq)\rightarrow goal(\{q(1)\}\cup M,G,V,H\setminus Hq,unsp).$	(5)
	$goal(M,[q(2)\|G],V,H,selected),\underline{c}(q,Hq)\rightarrow goal(\{q(2)\}\cup M,G,V,H\setminus Hq,unsp).$	(6)
	$goal(M,[r(1)\|G],V,H,selected),\underline{c}(r,Hr)\rightarrow goal(\{r(1)\}\cup M,G,V,H\setminus Hr,unsp).$	(7)
	$goal(M,[r(2)\|G],V,H,selected),\underline{c}(r,Hr)\rightarrow goal(\{r(2)\}\cup M,G,V,H\setminus Hr,unsp).$	(8)
	$goal(M,[s(1)\|G],V,H,selected),\underline{c}(s,Hs)\rightarrow goal(\{s(1)\}\cup M,G,V,H\setminus Hs,unsp).$	(9)
	$goal(M,[s(2)\|G],V,H,selected),\underline{c}(s,Hs)\rightarrow goal(\{s(2)\}\cup M,G,V,,H\setminus Hs,unsp).$	(10)
	$goal(M,[t(1)\|G],V,H,selected),\underline{c}(t,Ht)\rightarrow goal(\{t(1)\}\cup M,G,V,H\setminus Ht,unsp).$	(11)
	$goal(M,[t(2)\|G],V,H,selected),\underline{c}(t,Ht)\rightarrow goal(\{t(2)\}\cup M,G,V,H\setminus Ht,unsp).$	(12)
	$goal(M,G,V,H,selected)\quad\rightarrow goal(M,G,V,H,closed).$	(a)
	$goal(M,G,V,H,closed)\quad\rightarrow goal(M,G,V,H,closed).$	(b)
	$goal(M,G,V,H,open)\quad\rightarrow goal(M,G,V,H,open).$	(c)

図 3.7: 変換されたプログラム P_{ex}^{tr}

■

3.4.2 A*を実現する前向き計算

次に，A*アルゴリズムが行なっている *OPEN* , *CLOSED* , *SELECTED* の更新手続き (定義 3.3.2参照) を前向き計算で実現するために図 3.8, 図 3.9に示す 2 つのプログラム P_{Agg1}, P_{Agg2}を考える.

ここでは，集合演算を行なう組み込み述語 $\mathbf{agg}(A,CL,S)$ を導入する．第一引数 A は述語，第二引数 CL は条件式から成るリスト，第三引数 Sは集合を示し，現在までに導出されたアトムの集合 Iに対して，$\mathbf{agg}(A,CL,S)$ は，**(i)** Iから A と単一化可能なアトム $A'\theta$ $(\theta = mgu(A,A'))$ をすべて求め（これらのアトムの集合を S_rとする），**(ii)** S_rから CL に含まれる条件式をすべて満足するようなアトムをすべて取りだし，これらのアトムの集合を Sとする.

定義 3.4.1 Pをプログラム，Oを与えられた問合せ，P^{tr},P_{Agg1}, P_{Agg2}に対する直接帰結オペレータをそれぞれ$T_{P^{tr}}$,$T_{P_{Agg1}}$, $T_{P_{Agg2}}$とする．また，$\underline{C}_{\bar{P}}$を$\bar{P}$に含まれるアトム A およびその最適解 H_Aを示す述語$\underline{c}(A,H_A)$ からなる集合とする（例 3.3.1参照）.

$P^{tr}\cup P_{Agg1}\cup P_{Agg2}\cup\underline{C}_{\bar{P}}$に対して図 3.10に示す手続きによって定義される前向き計

$goal(M, G, V, H, open)$,
agg($goal(M', G', \_, H', unsp), [M \supseteq M', G \supseteq^* G',\ cost(M')+cost(H') < cost(M)+cost(H)], S'), S' = \phi$
$\rightarrow goal(M, G, V, H, open).$ (A.1)
$goal(M, G, V, H, unsp)$,
agg($goal(M', G', \_, H', \_), [M \supset M', G \supset^* G'], S'), S' = \phi$,
agg($goal(M'', G'', \_, H'', open), [M = M'', G =^* G'',\ cost(M'')+cost(H'') \leq cost(C)+cost(H)], S''), S'' = \phi$,
agg($goal(M''', G''', \_, H''', closed), [M = M''', G =^* G''',\ cost(M''')+cost(H''') \leq cost(C)+cost(H)], S'''), S''' = \phi$
$\rightarrow goal(M, G, V, H, open).$ (A.2)
$goal(M, G, V, H, closed)$ $\rightarrow goal(M, G, V, H, closed).$ (A.3)

図 3.8: プログラム P_{Agg1}

$goal(M, G, V, H, open)$,
agg($goal(Mo, \_, \_, Ho, open), [cost(Mo)+cost(Ho) < cost(C)+cost(H)], S), S = \phi$
$\rightarrow goal(M, G, V, H, selected).$ (B.1)
$goal(M, G, V, H, open), goal(Mo, \_, \_, Ho, open), cost(Mo)+cost(Ho) < cost(C)+cost(H)$
$\rightarrow goal(M, G, V, H, open).$ (B.2)
$goal(M, G, V, H, closed)$ $\rightarrow goal(M, G, V, H, closed).$ (B.3)

図 3.9: プログラム P_{Agg2}

算を行なえば，A* に基づく探索制御を実現することができる． ■

A*を実現する前向き計算手続き
入力: プログラム $P^{tr} \cup P_{Agg1} \cup P_{Agg2} \cup \subseteq_{\bar{P}}$, 観測 O
出力: 解 Ans: (解代入例, 最適な説明)（成功裏）または $false$（失敗裏）

```
1  begin
2    I_0:={goal({ }, [O], O, H_Ō, selected)}
3    i := 0
4    repeat
5      I_{i+1} := T_{P_Agg2}(T_{P_Agg1}(T_{P^tr}(I_i ∪ ⊆_P̄)));
6      I_{i+1}について無矛盾性の検査†を行ない Pと矛盾するゴールに
       対応するアトム goal(M, G, V, H, State) を I_{i+1}から削除する;
7      i := i + 1;
8    until (goal(M, [ ], V, 0, selected) ∈ I_i ) or
           (goal(_, _, _, _, selected) ∉ I_i);
9    if goal(M, [ ], V, 0, selected) ∈ I_i then
         Ans := (V, M);   % (成功裏に終了)
10   if goal(_, _, _, _, selected) ∉ I_i then
         Ans := false;    % (失敗裏に終了)
11 end.
```

† $goal(M, G, V, H, State)$ の第1引数 Mが矛盾仮説を含むかどうかで検査する．

図 3.10: A*を実現する前向き計算手続き

第 i 回目の反復において $T_{P^{tr}}$は 図 3.3のアルゴリズムの 5 行目の処理に対応しており，与えられたゴールから導出可能なサブゴールをすべて導出する．この時点においては，それらのサブゴールの状態は “unspecified” である．そして，与えられたゴールの状態を “closed” とし，第 $i-1$ 回目までの反復において生成されたゴールの状態が

"open" または "closed" であるゴールを導出する．また，$T_{P_{Agg1}}$ は 同アルゴリズムの 9-19 行目の処理に対応しており，$T_{P^{tr}}$ によって導出されたサブゴールの状態を更新する．$T_{P_{Agg2}}$ は同アルゴリズムの 4 行目の処理 に対応しており，$^{\forall}goal(M,G,V,H,open) \in I_i$ のなかで評価値 $cost(M) + cost(H)$ が最小であるアトムを選びそのアトムの状態を *selected* に移す．

例えば，$P_{ex}^{tr} \cup P_{Agg1} \cup P_{Agg2} \cup \subseteq_{\bar{P}_{ex}}$ に対してゴールに相当するアトム $goal(\{\ \}, [P(X,Y)], P(X,Y), \{r,t\}, selected)$ を与えて，定義 3.4.1の前向き計算を行なえば，図 3.5と同様にして，無駄な探索を絞り込むことができる．

3.5 実験結果

本章で提案した効率化法の有効性を確認するために，推論時間の比較実験を行なった．例題としては，ロボットの荷物運搬作業に関する計画問題を用いた．問題の概念図および知識ベースを図 3.11および図 3.12にそれぞれ示す． 仮説は，ロボットの一

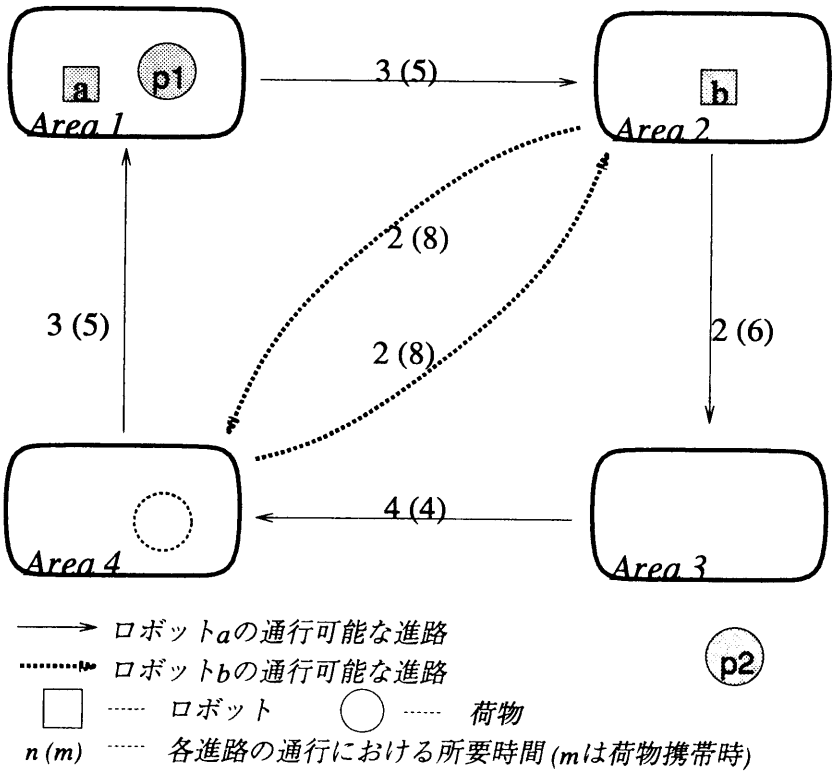


図 3.11: ロボットの荷物運搬作業に関する計画問題の概念図

ルール			
$carry(X,Y,P,X,Y,B,B) \leftarrow transport\_a(X,Y,P).$			(1)
$carry(X,Y,P,A,Y,B,B) \leftarrow move\_a(A,X), transport\_a(X,Y,P).$			(2)
$carry(X,Y,P,A,A,X,Y) \leftarrow transport\_b(X,Y,P).$			(3)
$carry(X,Y,P,A,A,B,Y) \leftarrow move\_b(B,X), transport\_b(X,Y,P).$			(4)
$carry(X,Y,P,Ia,Y,Ib,B) \leftarrow transport\_a(Z,Y,P), carry(X,Z,P,Ia,Z,Ib,B).$			(5)
$carry(X,Y,P,Ia,Y,Ib,B) \leftarrow transport\_a(Z,Y,P), move\_a(A,Z), carry(X,Z,P,Ia,A,Ib,B).$			(6)
$carry(X,Y,P,Ia,A,Ib,Y) \leftarrow transport\_b(Z,Y,P), carry(X,Z,P,Ia,A,Ib,Z).$			(7)
$carry(X,Y,P,Ia,A,Ib,Y) \leftarrow transport\_b(Z,Y,P), move\_b(B,Z), carry(X,Z,P,Ia,A,Ib,B).$			(8)
$move\_a(X,Y) \leftarrow step\_a(X,Y).$			(9)
$move\_a(X,Y) \leftarrow step\_a(X,Z), move\_a(Z,Y).$			(10)
$move\_b(X,Y) \leftarrow step\_b(X,Y).$			(11)
$move\_b(X,Y) \leftarrow step\_b(X,Z), move\_b(Z,Y).$			(12)
仮説集合			
$(transport\_a(1,2,P).,5)$	$(transport\_a(3,4,P).,5)$	$(transport\_a(2,3,P).,3)$	
$(transport\_a(4,1,P).,3)$	$(transport\_b(2,4,P).,7)$	$(transport\_b(4,2,P).,7)$	$\}(H)$
$(step\_a(1,2).,3)$	$(step\_a(2,3).,3)$	$(step\_a(3,4).,2)$	
$(step\_a(4,1).,2)$	$(step\_b(2,4).,4)$	$(step\_b(4,2).,4)$	

図 3.12: 知識ベース

行程の作業とし，仮説のコストは，その一行程にかかる作業時間で与えた．また，知識ベースに現れる仮説の数をkとし，kをパラメータとして知識ベースの規模を変化させて実験した．

実験結果を図3.13に示す．実験は計算機SUN4/75上で言語SICStus Prologを用いて行なった．実線は本章で提案した手法，破線はヒューリスティック評価関数を用いない（すなわち$\hat{h}(g) \equiv 0$）最良優先探索に基づく手法の結果を示している．知識ベースの規模が増大するほど，本手法を用いることにより，推論時間がそれぞれ大幅に短縮されることがわかる．また，ヒューリスティック評価関数$\hat{h}$を求めるための事前解析自身に要する時間は，命題論理のレベルに抽象化してプログラム解析を行なっているため，知識ベースの規模kが大きい場合には，実際の仮説推論に要する時間の約1%以下であり，無視できる程度であった．

3.6 まとめ

コストに基づく仮説推論において効率的な最適解探索法を提案し，一階述語論理ホーン節を対象とする仮説推論システムを実現した．

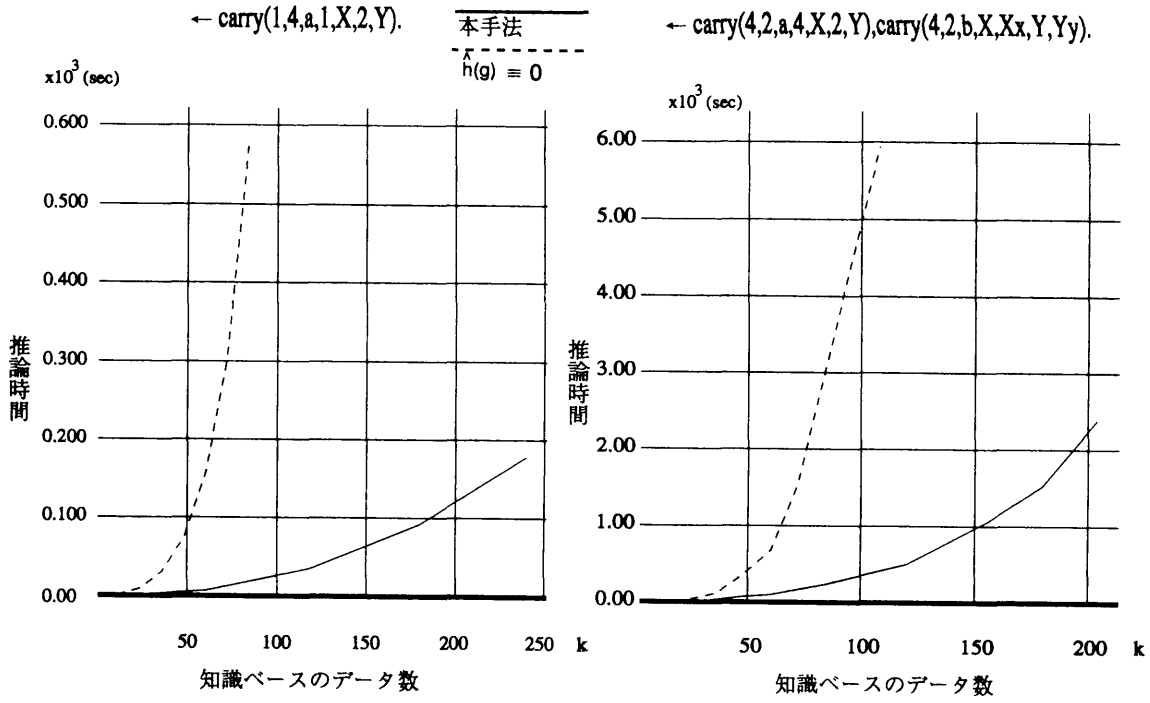


図 3.13: 実験結果

本章で提案した仮説推論システムは，論理プログラミングの分野における問合せ処理技術に対して A*アルゴリズムの持つ探索制御技術を導入することにより，与えられた観測に対して最良の説明を効率的に求めるものである．さらに，実行可能条件を満たし，かつ，実際のコスト h になるべく近い値をとるような，ヒューリスティック評価関数を求めるために，プログラム解析を用いた $\hat{h}$ の導出方法を提案した．

また，本章で提案した仮説推論システムを前向き推論を用いて実装し，その有効性を確認した．本推論システムの実装法は，前向き計算の一制御法として捉えることができ（例えば文献 [RSS92]），演繹データベースの分野においても興味深いものであると思われる．

本章では，観測の説明は仮説マルチセットで表現されるとしたが，定義 3.3.1 を少し変更する（付録 C に示す）ことにより，説明が通常の仮説集合で表現される場合にも対応できる．

また，本章では，仮説の選択の基準として各仮説に与える重みは正の数とし，観測

の説明のコストが観測の説明に現れる仮説の重みの和で与えられる場合について考えた．しかしながら，仮説推論の応用例題として知られる診断問題等では，各仮説の重みは確率で表されることが多い（例えば，文献 [Poo93], [CS90], [Poo92] など）．本章で提案した仮説推論システムは，最適解探索として一般性の高い A* アルゴリズムの考えを導入しているため，定義 3.3.2のアルゴリズムを少し変更することにより，確率に基づく推論における最適解探索にも対応できる[KSI94b]．

A*アルゴリズムは，ヒューリスティックな評価関数を最も効率的に利用できる最適解探索法として広く知られているが，探索の制御に用いる集合 $OPEN, CLOSED$の保持のために必要な記憶量が極めて大きいという問題点を持つ．従って，今後の課題としては，A*の改良アルゴリズムである IDA*[Kor85]を本手法に適用することが挙げられる．

第4章

PARCAR: コストに基づく並列仮説推論システム

4.1 はじめに

知識ベースシステムは，AIの基礎的研究テーマであると同時にAIの実用化や知識情報処理システム構築のための要素技術として不可欠な研究課題である．また，実用化に耐える規模の知識ベースシステムが要求する計算能力にこたえるためには並列処理が欠かせないものと考える．本研究では，知識ベースシステムの推論機構として高次推論の一形式である仮説推論に着目し，その並列化について検討する．

仮説推論は，与えられた観測を説明するために，一時的に成り立つ知識をとりあえず正しいと仮定したり，足りない知識を仮説で補うなど，柔軟で知的な推論処理である [Poo88],[井上 92]．しかしながら，仮説推論システム実現上の問題点として，仮説推論の計算量が極めて大きいことが知られており [井上 92]，仮説推論の探索空間を絞り込む工夫について多くの研究結果が報告されている．例えば，文献 [OI93],[加藤 94] などにおいて，演繹データベースの分野における問合せ処理のさまざまな最適化技術が仮説推論に応用され効率の良い仮説推論が提案されている．

仮説推論では一般に，観測を説明する仮説集合は複数存在する．しかしながら，診断·計画問題等において見られるように，ユーザが要求する解はすべての説明ではなく，むしろ最も好ましい説明であることが多い．このような要求に対しては，仮説に好ましさの基準（確率，コスト等）を与える手法 [Poo93],[CS94] が報告されている．そこで文献 [加藤 95] では，A* アルゴリズムを仮説推論に応用することにより，最も

好ましい説明を求める仮説推論が提案されている．

本章では，これらの工夫に加えて，仮説推論の探索空間を複数のプロセッサへ分散し推論処理を並列化することにより，仮説推論の高速化を実現する．仮説推論の推論処理技術に並列ヒューリスティック探索が持つ探索制御技術を導入し，与えられた観測を説明する最小コストの仮説集合を効率的に求めることにより最適な説明を得る並列仮説推論を提案する．本章で提案する並列仮説推論システム PARCAR (PARallel Cost-based Abductive Reasoning system) を MIMD 型分散メモリ並列計算機 AP1000 上に実装し，その実験結果についても報告する．

本章の構成は以下の通りである．まず **4.2節**で本研究で対象とする仮説推論の枠組について簡単に述べ，その並列化について言及する．次に **4.3節**で並列仮説推論システム PARCAR を提案し，**4.4節**で関連研究について述べる．そして，**4.5節**では定量的な解析および実際の並列計算機を用いた比較実験の 2 側面より PARCAR の性能について評価を行う．

4.2 仮説推論の並列化

本章では，前章と同様に仮説の選択の基準として知識ベースに含まれる各仮説に重み（コスト）が与えられている場合を対象に，与えられた観測に対して最適な説明を求める仮説推論を考える．各仮説に与えられる重みは正の数とし，仮説集合のコストは集合に含まれる仮説の重みの和で与えられるものとする．説明が最適であるとは説明を表す仮説集合のコストが最小であることとする．前章で説明 Eはマルチセットとして扱われたのに対して，本章では集合で扱われる．

定義 4.2.1 Fをホーン節集合（「事実」と呼ぶ），Hを基底単位節の集合（「仮説集合」と呼ぶ）とする．また，Oを存在束縛されたアトムの連言（「観測」，または単に「ゴール」と呼ぶ) とする． このとき, Oの $F \cup H$による最適な説明とは，以下の条件

[0] 本研究は文献 [KKSI96],[加藤 96],[KSI96],[KSI97] の報告をもとにして本論文にまとめたものである．

を満足するような，Hの要素の代入例から成る集合 Eを求めることである．

$$F \cup E \vdash O \quad (F \cup E\text{から}\ O\text{が証明される}) \tag{CBA1}$$

$$F \cup E \nvdash \mathit{false} \quad (F \cup E\text{は無矛盾である}) \tag{CBA2}$$

$$cost(E) \leq cost(D) \text{ for all } D : F \cup D \vdash O, F \cup D \nvdash \mathit{false}$$

(条件 CBA1,CBA2を満たす集合の中で Eのコストが最小)

$cost(H)$ は仮説集合 Hに含まれる仮説のコストの和を表すとする．ここで，Fは常に成り立つ知識として扱われる．一方で，Hの代入例からなる部分集合は Fと矛盾する可能性がある．また，Fと Hの和集合をプログラムと呼び，Pで表記する． ■

定義 4.2.2 Fの中に存在する負節を「一貫性制約節」(あるいは単に「制約式」) と呼ぶ．制約節は，論理式 $\mathit{false} \leftarrow A_1, \cdots, A_n$ で表現される．ただし，A_k $(1 \leq k \leq n;\ n \geq 1)$ はアトムであり，false は「矛盾」を表す． ■

例 4.2.1 図 4.1に示す簡単な例題知識ベース P_{ex}を考える． P_{ex}に対して，問合せ "$\leftarrow p(X,Y)$" を与える．同図の知識ベースが事実と仮説の和集合 $F \cup H$であり，問合せ $\leftarrow p(X,Y)$ が観測 Oである．

事実	仮説	コスト	仮説	コスト
p(X,Y) ← a(X),c(Y).	c(1).	5	c(2).	6
p(X,Y) ← b(X),d(Y).	c(3).	7		
a(X) ← e(X).	b(1).	2	b(2).	3
e(1).	b(3).	2		
false ← b(X),d(X).	d(1).	2	d(2).	4

図 4.1: 例題知識ベース P_{ex}

図 4.2に例 4.2.1に対する SLD 反駁 [Llo84] を用いた仮説推論の実行例を示す．同図においてゴール節に現れる M なるオペレータが付いたアトム "MA" は，サブゴール $\leftarrow A_0$（ここでアトム A は A_0のある代入例）を解く際に，仮説を入力節として導出を行なったこと（すなわち，A に対して仮説が立てられたこと）を示すためのマークである．また，ゴール内のアトムの選択規則は最左優先とする．

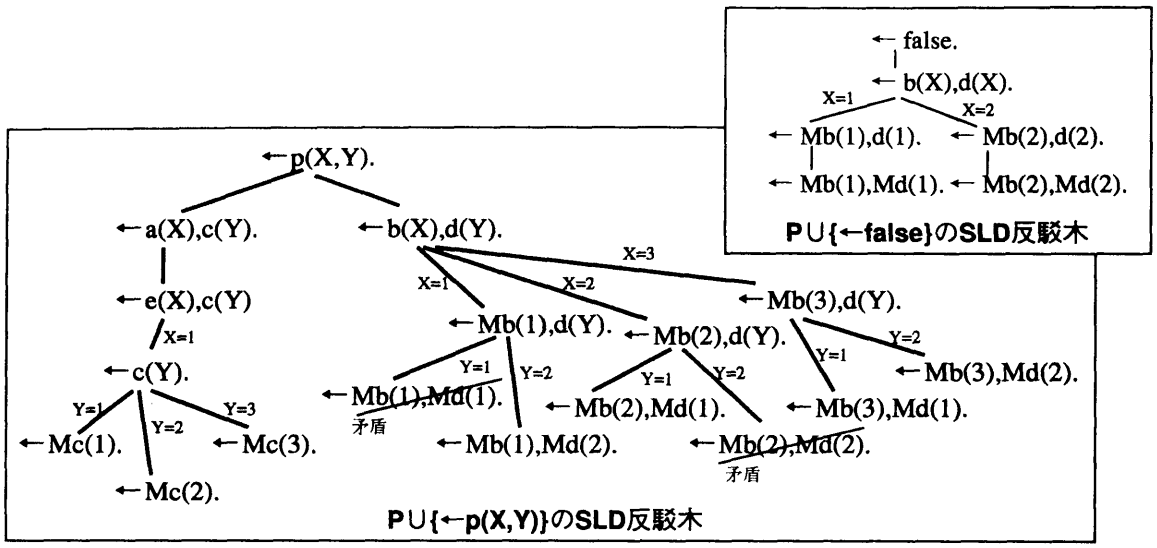


図 4.2: 仮説推論の実行例

ここで，図4.2の$\leftarrow false$に対するSLD反駁木から，成功葉に現れる仮説集合$\{b(1), d(1)\}$または$\{b(2), d(2)\}$を仮定することは知識ベースに対して矛盾を起こすことがわかる（これらの仮説集合を矛盾仮説と呼ぶ）．また，同図の$\leftarrow p(X, Y)$に対するSLD反駁木より，右から2番目の成功葉から得られる仮説集合$\{b(3), d(1)\}$のコストが反駁木の任意の成功葉から得られる仮説集合のコストにおいて最小であるので，これを説明としてゴール$\leftarrow p(X, Y)$の最適解$\{X = 3, Y = 1\}$が得られる． ■

しかしながら，一つの最適解を得るために全解を求めるのは，無駄な探索空間が大きく極めて非効率である．そこで，与えられた観測に対する最適な説明を効率的に求めるために，本仮説推論が対象とする探索空間の評価関数を以下のように定義する．

定義 4.2.3 Pをプログラム，Oを与えられた観測とする．また，Pにおけるゴール$\leftarrow O$に対するSLD反駁木（$P \cup \{\leftarrow O\}$のSLD反駁木と呼ぶ）における任意のゴール節$g = \leftarrow \mathsf{M}L_1, \cdots, \mathsf{M}L_i, A_{i+1}, \cdots, A_n$について，$\hat{h}(g)$を$g$から成功葉に至る最適導出におけるコストの予測値，$h(g)$を実際のコストの値とする．このとき，最適解探索におけるgの評価値$f(g)$を以下のように定義する．

$$f(g) = cost(\{L_1, \cdots, L_i\}) + \hat{h}(g)$$

$\hat{h}(g)$ は以下の条件を満足するような予測値とする.

$$0 \leq \hat{h}(g) \leq h(g)$$

$$h(g) = cost(\bigcup_{j=i+1}^{n} A_j \text{の最適な説明}) \quad \blacksquare$$

ここで, $cost(\{L_1, \cdots, L_i\})$ は反駁木の根から gに至る導出におけるコストの値を示している. また, 本論文では, A_jの説明が存在しないとき（つまり, $\leftarrow A_j$の反駁が存在しないとき）には, $h(g) = \infty$ とする.

仮説推論の探索空間に対してこのような評価関数が与えられることによって, 最適な説明を求める問題は最適解探索の問題に帰着される. そこで文献 [加藤 95] では, SLD 反駁に対して A* アルゴリズムが持つ探索制御技術を導入することにより効率的な仮説推論システムが提案されている. 文献 [加藤 95] で実現した仮説推論の探索空間の絞り込みに加えて, 本章では, その探索空間を複数のプロセッサに分散することによる仮説推論システムの並列化について報告する. なお, 本章では, 知識ベースに現れる制約式については$\leftarrow false$に対する反駁[1]を行ない, 矛盾仮説の集合をあらかじめに求めておくことにする.

4.3 並列仮説推論システム PARCAR

本章で提案する並列仮説推論システム PARCAR は. 並列ヒューリスティック探索が持つ探索制御機構を仮説推論の推論機構に導入し, 複数のプロセッサが同期・通信を通じて並列に推論を行なうことにより最適な説明を効率的に求めるものである [加藤 96]. 仮説推論のノード展開は述語のユニファイや変数の束縛情報の伝搬等をともなう SLD 導出による. したがってノード展開にかかる計算量は大きく, 生成される子ノードの数によってその計算量は大きく異なる. そこで PARCAR では, 各プロセッサで一回の反復で行なう推論の処理量は展開されるノード数ではなく, 生成されるノード数に基づくものと考え, 全プロセッサ間でそのノード数が一定になるように

[1] PARCAR を用いてこの反駁を並列に実行することも可能だが, その処理自体は, 知識ベースの規模が大きい場合には仮説推論処理と比較して無視できるほど軽い.

負荷分散を行っている．また，仮説推論では展開されるノードは述語論理ホーン節で表現される．したがってノードのデータ構造は複雑でありノード分散にかかる通信コストも大きくなる．そこでPARCARでは各プロセッサが1反復に処理する処理量をある程度大きくし，プロセッサ間の通信回数を抑え，通信コストを抑えている．

4.3.1 Two Phases

並列仮説推論システムPARCARは知識ベースと観測を受け取ると，全てのプロセッサに知識ベースをブロードキャストし，観測をあるプロセッサに送信して推論を実行する．PARCARにおける並列推論処理の流れを図4.4に示す．

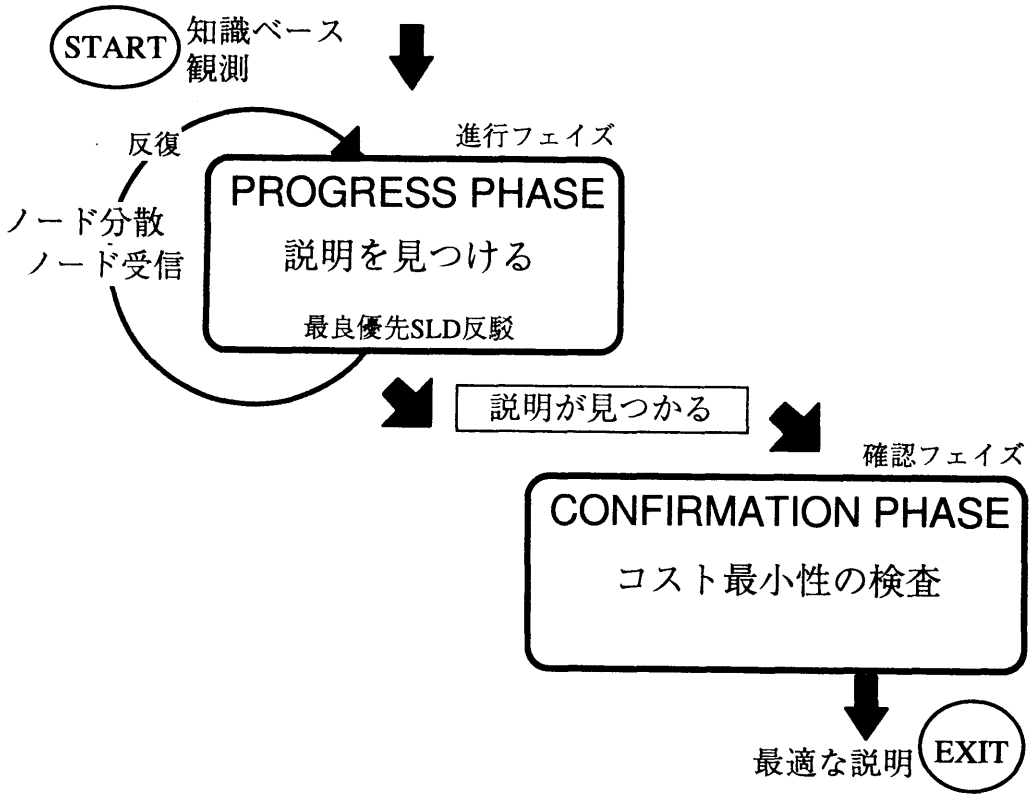
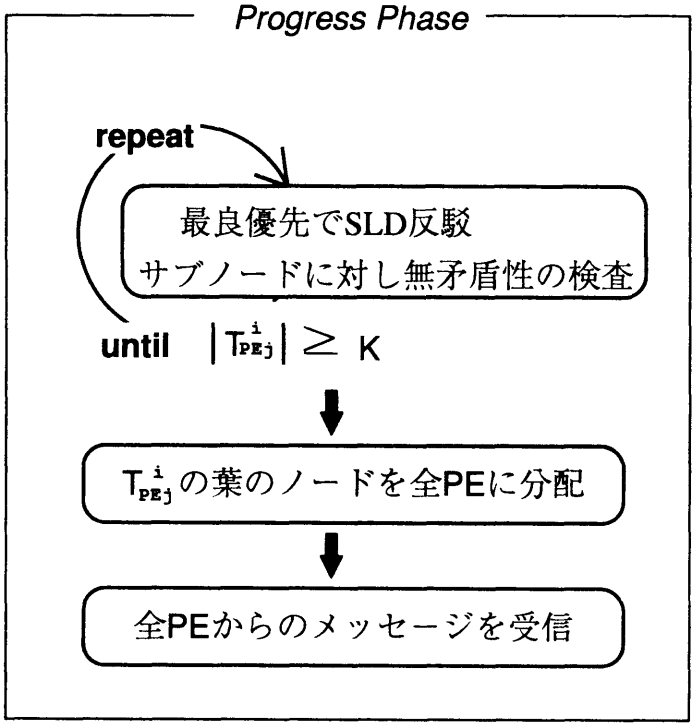


図4.4: 並列仮説推論処理の流れ

PARCARは「進行」(progress) フェイズおよび「確認」(confirmation) フェイズの2つの段階を経て解を求める．進行フェイズでは，各プロセッサは複数プロセッサ間と同期・通信をしながら分割された探索空間に対して局所的な推論を行う．同フェイ

ズでは，個々のプロセッサは与えられた観測に対する説明を (一つ) 見つけることを目標として,「ゴール節展開」「サブゴール節分散」「サブゴール節受信」の 3 つの手続きを繰り返し実行する．進行フェイズでの処理の流れを図 4.6に示す．



$|T^i_{PEj}| = T^i_{PEj}$に含まれるノード数
T^i_{PEj} : i 回目の反復において PE_jによって生成される部分探索木

図 4.6: 進行フェイズ

あるプロセッサにて説明が見つかると，全てのプロセッサは確認フェイズに移行する．確認フェイズでは，各プロセッサは「ゴール節展開」手続きを実行し，求められた説明のコストが最小かどうかを検査する．同手続きにおいて，よりコストが小さい説明が見つかった場合には説明を更新する．進行フェイズにおける最良優先に基づく推論処理により，多くの場合，同フェイズで得られた説明は既に最小コストを有する．しかしながら，同フェイズではプロセッサ内での局所推論が行われるため，確認フェイズによる説明のコスト最小性の検査が必要になる． 確認フェイズでの処理の流れを図 4.8に示す．

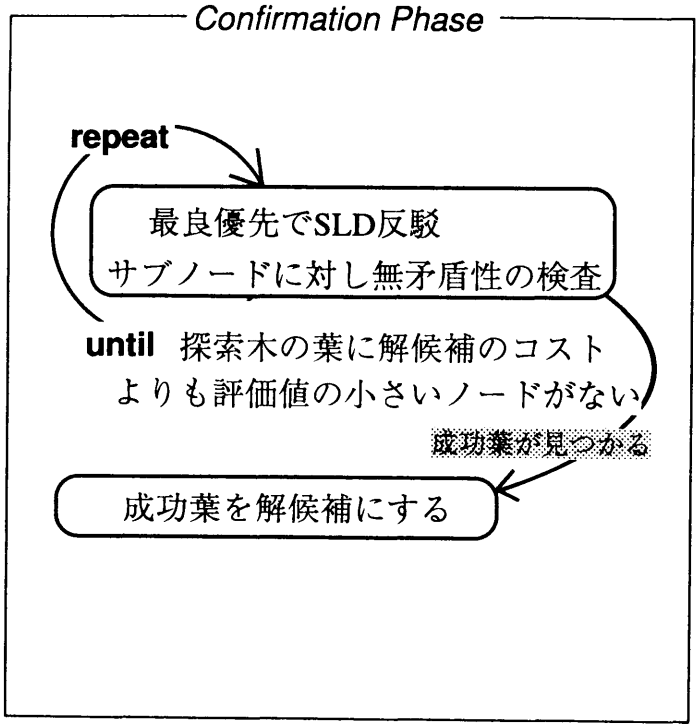


図 4.8: 確認フェイズ

4.3.2 システムのアルゴリズム

図 4.9に PARCAR のアルゴリズムを示す．同図において，n はプロセッサの台数，$OPEN_i^j$, $CLOSED_i^j$, $Gmin^j$, SG^j, RS^j 及び ANS^j はプロセッサ $j(1 \leq j \leq n)$ で扱われるゴール節の集合を表す．$OPEN_i^j$, $CLOSED_i^j$, SG^j は，i 回目の反復において，プロセッサ jによってまだ展開されていないゴール節の集合，既に展開されたゴール節の集合，および，形成された部分木の葉に現れるゴール節の集合をそれぞれ示している．また ANS^j はプロセッサ jにおいて導出された最適な説明の候補を表すゴール節の集合を示す．なお，これらの集合は要素であるゴール節 gの評価関数 $f(g)$ に基づいた順序つき集合として計算機上に実装される．

本アルゴリズムは,「ゴール節展開」「サブゴール節分散」「サブゴール節受信」の 3 つの手続きの反復で構成される．直観的には，プロセッサ jにおける i 回目の反復でこれらの手続きは次のような動作をとる．$T_{i,l(1 \leq l)}^j$ を PARCAR がプロセッサ jで形成する部分 SLD 反駁木とし，$|T_{i,l}^j|$ を $T_{i,l}^j$に含まれるゴール節の数とする．

並列仮説推論システム PARCAR
入力: プログラム P, 観測 O
出力: 解: (解代入例, 最適な説明) (成功裏) または false (失敗裏)

```
                                                        % 各プロセッサの初期設定
 1 OPEN_0^1 := {←O};  phase^{j(1≤j≤n)} := progress;
 2 OPEN_0^{j(1<j≤n)} := φ;  CLOSED_0^{j(1≤j≤n)} := φ;  ANS^{j(1≤j≤n)} := φ;
 3 begin                                                % プロセッサ j内の処理
 4   i := 0; t := ∞;
 5   OP_0^{j(1<j≤n)} := φ;                              % 4.3.3節参照
 6   repeat
 7     OPEN' := OPEN_i^j; CLOSED' := φ; SG^j := φ;  escape := false; shifted := false;
 8     repeat                                           % ゴール節展開手続き
 9       Gmin^j := {∀g ∈ OPEN' ∪ SG^j | for all g' ∈ OPEN' ∪ SG^j : f(g)≤f(g')};
10       Gmin^j に対して Pにおける 1 段階の SLD 導出で得られるサブゴールをすべて求め
         これらの集合を RS^j とする;
11       RS^j について無矛盾性の検査†を行ない Pと矛盾するゴールを RS^j から削除する;
12       ANS^j := {∀g∈RS | gは成功葉, かつ, 成功葉のコスト中で f(g) が最小 };
13       SG^j := SG^j∪RS^j ;
14       CLOSED' := CLOSED' ∪ {g} ;
15       OPEN' := OPEN' \ Gmin^j ; SG^j := SG^j \ Gmin^j ;
16       if (ANS^j ≠ φ)      then escape := true ;
17       if ( phase^j = progress and (|SG^j| + |CLOSED'|≥K or OPEN' ∪ SG^j = φ) )
                             then escape := true ;
18       if ( phase^j = confirmation and (∀g ∈ OPEN' ∪ SG^j | f(g) > t or ∃g ∈ ANS^j | f(g) < t )
                             then escape := true ;
19     until (escape)
20     OPEN_{i+1}^j := OPEN';  CLOSED_{i+1}^j := CLOSED_i^j ∪ CLOSED';
21     if (phase^j = progress) then
22       begin
23         distribution_goals(ANS^j, SG^j, {OP_i^1, ···, OP_i^n});        % サブゴール節分散手続き
24         reception_goals(OPEN_{i+1}^j, phase^j, shifted, {OP_i^1, ···, OP_i^n}); % サブゴール節受信手続き
25       end
26     i := i + 1;
27   until ( (phase^j = confirmation and not shifted) or (OPEN_i^{j(1≤j≤n)} = φ) )
28   ANS := {∀g∈ANS ∪ ANS^{j(1≤j≤n)} | for all g'∈ANS ∪ ANS^{j(1≤j≤n)} : f(g)≤f(g')}
29   if (∃← ML_1 ··· ML_m ∈ ANS) then return (解代入例,{L_1···L_m})
30                              else return false
31 end.
```

† サブゴールに現れる仮説が矛盾仮説を含むかどうかで検査する.

図 4.9: 並列仮説推論システム **PARCAR** のアルゴリズム

ゴール節展開手続き 同手続きが取るフェイズに基づき以下のいずれかの手続きを実行する.

進行フェイズ

$OPEN_i^j$に含まれるゴール節を根とするような部分SLD反駁木$T_{i,l}^j$を$\sum_l |T_{i,l}^j| \geq K$となるまでゴール節の評価値に対する最良優先で生成する. Kは推論開始時にユーザから与えられるパラメータである. ただし, 解が得られた, あるいは, 展開するゴール節が存在しない場合には直ちに手続きを終了する.

確認フェイズ

$OPEN_i^j$に含まれるゴール節を根とするような部分SLD反駁木$T_{i,l}^j$を$T_{i,l}^j$に現れるすべての葉のゴール節gが$f(g) \geq t$となる, あるいは, tよりも評価値が小さい新しい解が得られる[2]までゴール節の評価値に対する最良優先で生成する. tは進行フェイズにおいて求められた最適な説明の候補のコストである. PARCARはこのフェイズを経て成功裏に終了する.

図4.3.2にプロセッサjにおけるi回目の反復で生成されるSLD反駁木の模式図を示す.

サブゴール節分散手続き ゴール展開手続きにおいて最適解の候補が得られたかどうかを調べ, 得られた場合には, そのコストおよび確認フェイズへの移行要求を全プロセッサに放送する. そして, 同手続きにおいて生成された部分反駁木$T_{i,l}^j$の葉のゴール節をその評価値が小さい物から順にゴール節の数が均一となるように全プロセッサに分散する. このとき, あるプロセッサに評価値の小さいゴール節を集中して分配しないようにプロセッサjで生成されたゴール節でその評価値がr番目に小さいものはプロセッサ$(r+j) \bmod n$へ送信する. 本手続きのアルゴリズムを図4.11に示す.

サブゴール節受信手続き あるプロセッサから確認フェイズへの移行要求が届いた

[2]定義4.2.3より, 反駁木Tに現れるゴール節gとそのサブゴール節sgとの間には必ず$f(g) \leq f(sg)$の関係が成り立つので, 実際にはTの葉のゴール節についてのみtとの大小関係を調べれば良い.

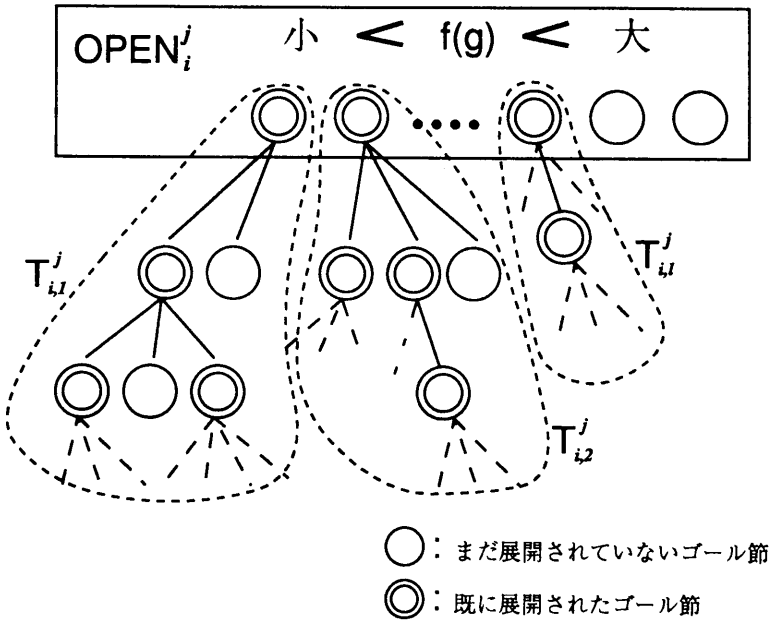


図 4.10: プロセッサ jで生成される SLD 反駁木の模式図 (反復 i 時)

らフェイズを移行し，*shifted* の値を true にする．このとき，付随して届く最適解の候補のコストを確認フェイズでの探索終了条件となる閾値 t に代入する．なお，複数のプロセッサから同様のコストが届いた場合にはその最小値を t に代入する．全プロセッサから送信されたゴール節を $OPEN_{i+1}^j$に追加する．このとき，$\bigcup_j^n OPEN_{i+1}^j = \phi$となればPARCARは失敗裏に終了する．本手続きのアルゴリズムを図 4.12右に示す．

本アルゴリズムでは，各プロセッサは上記 3 手続きにおいて直接同期を取るためのフラグを持っておらず，これらの手続きは非同期に処理される．しかしながら，「サブゴール節受信」手続きは全プロセッサからデータを受信するまで処理が終了しない仕組みになっている．これにより 3 手続きの反復回数については全プロセッサ間で同期が実現されている．

任意の有限な SLD 反駁木において，すべてのゴール節 gに対し条件$\hat{h}(g) \leq h(g)$ が成立する時，本並列アルゴリズムは実行可能であり，解が存在する場合には必ず最適解を検出する．

procedure *distribution_goals*(ANS^j, SG^j)

```
 1 begin
 2   l := j;
 3   dist[l(1 ≤ l ≤ n)] := φ;
 4   if (∃g ∈ ANS') then
 5     begin
 6       メッセージ MSG_SHIFT とともに f(g) を
         全プロセッサに放送;
 7       while ( ∃g' ∈ SG | f(g')≤f(g) )
 8         begin
 9           SG から評価値が最小のゴールを取り出し
             dist[l mod n] に追加する;
10           l := l + 1 ;
11         end
12       ANS := ANS ∪ ANS^j ;
13     end
14   else
15     begin
16       while ( SG ≠ φ )
17         begin
18           SG から評価値が最小のゴールを取り出し
             dist[l mod n] に追加する;
19           l := l + 1 ;
20         end
21     end
22   メッセージ MSG_SG とともに dist[j(1 ≤ j ≤ n)] を
     プロセッサ j へ送信する;
23   ANS^j := φ;
24 end.
```

図 4.11: サブゴール節分散手続き

procedure *reception_goals*($OPEN^j_{i+1}, phase, shifted$)

```
 1 begin
 2   ctr := 0 ;
 3   while ( ctr ≤ n )
 4   begin
 5     case メッセージ MSG_SHIFT:
 6       if (t > 受信データ) then t := 受信データ ;
 7       phase := confirmation shifted := ture ;
 8     case メッセージ MSG_SG:
 9       受信データを OPEN^j_{i+1} に追加;
10       ctr := ctr + 1 ;
11   end
12 end.
```

図 4.12: サブゴール節受信手続き

4.3.3 動的な負荷分散方法

本節では，各プロセッサが持つワークリスト ($OPEN^j$) に仮説推論の探索空間を分割する有効な方法として，動的な負荷分散方法である"Dynamic Work Distribution" [KSI97] を提案する．

Dynamic Work Distribution は 2 つの効果を狙った分散方法である． 一つは進行フェイズにおける「最良優先に基づく SLD 導出」の適切な推論制御，もう一つは確認フェイズにおける各プロセッサの処理粒度調節である．

静的な負荷分散方法の問題点

PARCAR では，進行フェイズの「ゴール節展開」手続きにおいて，各プロセッサで生成されるゴールの数 (4.3.2節の Kに相当) を固定することによって，プロセッサの処理粒度がなるべく一定になるようにしている．同手続きの後,「ゴール節分散」手続きにより，プロセッサ内で生成されたゴール節のうち未展開の節 (形成された反駁木の葉ノードに相当) を各プロセッサに分配することにより負荷分散が実行される．推論処理の並列効果を引き出すためには，一回の反復で各プロセッサで展開されるゴールの数を均一にすることが効果的に働くが，次の問題が生じる．

本研究が対象とする仮説推論をはじめとして多くの推論・探索過程では，一般に，生成される子ノードの数は，反駁を行うゴール節や展開するノードによって，大きく異なる．また，PARCAR のように最良優先に基づき推論・探索を実行するようなシステムでは，プロセッサ内で最良な評価値を持つゴール節が優先的に展開されるので，プロセッサ内に形成される探索空間は展開されるゴール節，即ちそれぞれのプロセッサが持つワークリスト $OPEN_i^j$の内容に大きく依存する．

例えば，推論を進めてもなかなか評価値が変化しないゴール節 (そのゴール節から最適な説明が導出される見込みが高いことを意味する) を持つプロセッサは，図 4.13 の様な探索空間を形成する傾向にある．逆に，見かけは評価値が良くても推論を進め

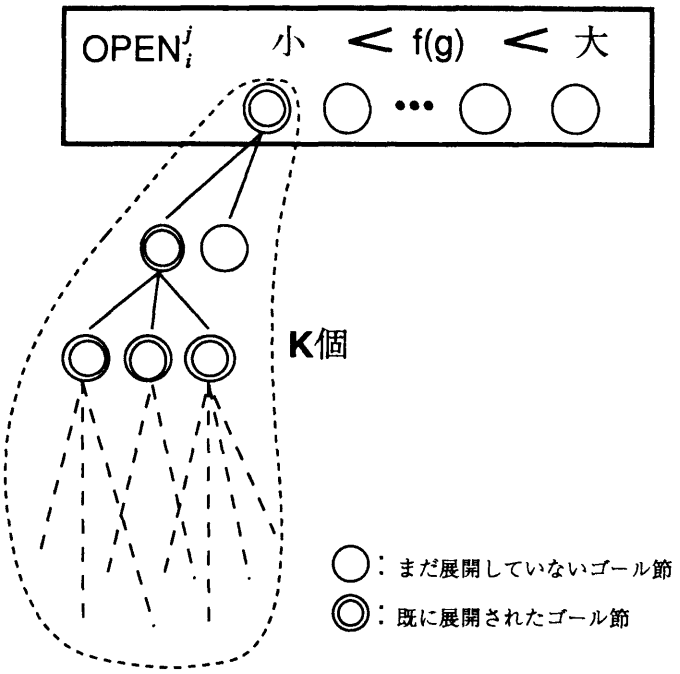


図 4.13: 深い反駁木を形成する場合

ると失敗に終ったりすぐに評価値を上げてしまう (コストが大きい仮説を生成することを意味する) ゴール節を多く持つプロセッサは図 4.14のような探索空間を形成するであろうことは容易に予想できる.

ここで注目すべきことは, 同じ「ゴール節展開」手続きを並列に実行しても, 前者のプロセッサには未だ評価値の良いゴール節が多く残っているのに対し, 後者のプロセッサにはあまり評価値の良くないゴール節しか残らないことである. PARCAR のゴール節分散方法は図 4.11のアルゴリズムより, 単純な方法で各プロセッサに分配するゴール節の評価値がなるべく均等になるような方式をとっているため, 進行フェイズにおける反復処理において一つプロセッサのワークリストが図 4.13のような状態になる場合, ある特定のプロセッサに良いゴールが集中してしまう可能性がある. このような集中を回避し, すべてのプロセッサに良いゴールを万遍なく分散させることが, 最適な説明を高速に得る[3] ために必要である[4].

[3] 最適解を求める問題では, 基本的には評価値の良い部分問題を優先的に処理してゆくことが処理速度の向上につながる.

[4] 問題に対してヒューリスティックの信頼性が低く (即ち難問題を解く場合), 計算資源に十分な余力がある場合などには, 稼働していないプロセッサに見かけ上評価値のあまり良くない部分問題を解

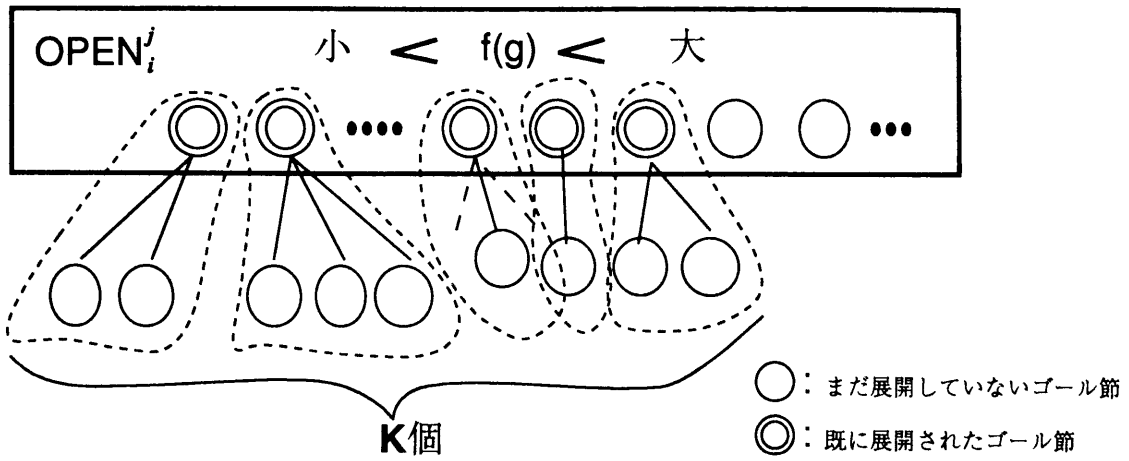


図 4.14: 浅い反駁木を複数形成する場合

また，一方，確認フェイズでは最適な説明の候補のコストよりも評価値の小さい未展開ゴールが無くなるまで展開を続けるため，各プロセッサで生成されるサブゴール節の数は未展開ゴール節のコストの善し悪しに大きく依存する．即ち，確認フェイズでの各プロセッサの処理粒度は，同フェイズに移行した直後の各プロセッサが持つワークリスト $OPEN^j$の質 (qualkity) に大きく依存する．

確認フェイズでは，プロセッサ間の通信や負荷分散は行われないので，同フェイズに移行した直後の $OPEN^j$の質を一様にすることが，各プロセッサの負荷均衡 (load balancing) において重要である．そこで，プロセッサ間の $OPEN^j$の質を可能な限り一様にするような動的な負荷分散方法が各プロセッサの確認フェイズの処理時間を速めることになる．

Dynamic Work Distribution の導入

PARCAR の負荷分散方法に各プロセッサが持つワークリストの質に応じて動的に分配先のプロセッサを決定するゴール節分散方式を導入する．

本節で提案する負荷分散方式は，各プロセッサが持つ $OPEN^j$に含まれるゴール節の評価値の差異をなるべく少なくする，つまり $OPEN^j$の質を一様に保とうとするこ

かせることの効果が期待できるが，これについての議論は他に譲る．

とにより，進行フェイズにおける反復処理においては，ある特定のプロセッサに良いゴールが集中するのを防ぎ，確認フェイズの処理時間を高速化するものである．

procedure *distribution_goals* $(ANS^j, SG^j, \{OP^1, \cdots, OP^n\})$

```
1 begin
2   dist[l(1 ≤ l ≤ n)] := φ;
3   if (∃g ∈ ANS') then
4     begin
5       メッセージ MSG_SHIFT とともに f(g) を
        全プロセッサに放送;
6       dynamic_work_distribution ({g' ∈ SG | f(g')≤f(g)}, dist[], {OP^1, ..., OP^n});
7       ANS := ANS ∪ ANS^j;
8     end
9   else
10    begin
11      dynamic_work_distribution (SG, dist[], {OP^1, ..., OP^n});
12    end
13  メッセージ MSG_SG とともに dist[j(1 ≤ j ≤ n)] を
    プロセッサ jへ送信する;
14  メッセージ MSG_QUALITY とともに{best N goals ∈ OPEN_i^j}を
    全プロセッサに放送する;
15  ANS^j := φ;
16 end.
```

図 4.15: ゴール節分散手続き< 2 >

PARCAR のアルゴリズム (図 4.9参照) に対して「サブゴール節分散」「サブゴール節受信」の両手続きを図 4.15, 図 4.16に変更する．

この変更により，i 回目の反復処理において各プロセッサ jは全プロセッサの $i-1$ 回目の反復処理におけるワークリスト $OPEN^j_{i-1}$の最良 N個のゴール節の集合 OP^j_i を持つ[5]．この集合を用いて以下のように各プロセッサのワークリスト $OPEN^j$の質 (quality) を近似する．

i 回目の反復におけるプロセッサ jのワークリスト $OPEN^j_i$の質 $\approx quality(OP^j_i)$

$$quality(OP^j_i) = \text{average of } f(g) \mid g \in OP^j_i$$

i 回目の反復においてプロセッサ jで生成された反駁木 T^i_{PEj} :に含まれるサブゴール節のうち葉にあたるサブゴール節が分散の対象になるが，それらの節 gはその評価

[5] 4.5.3節で述べる実験では $N = 6$ とした．

procedure *reception_goals* $(OPEN^j_{i+1}, phase, shifted, \{OP^1, \cdots, OP^n\})$

```
 1 begin
 2   ctr1 := 0; ctr2 := 0;
 3   while ( ctr1 ≤ n or ctr2 ≤ n )
 4   begin
 5     case メッセージ MSG_SHIFT:
 6       if (t > 受信データ) then t := 受信データ ;
 7       phase := confirmation shifted := ture ;
 8     case メッセージ MSG_SG:
 9       受信データを OPEN^j_{i+1} に追加;
10       ctr1 := ctr1 + 1 ;
11     case メッセージ MSG_QUALITY:
12       OP^j_{i+1} :=プロセッサ jからの受信データ;
13       ctr2 := ctr2 + 1 ;
14   end
15 end.
```

図 4.16: ゴール節受信手続き< 2 >

値 $f(g)$ の値によって以下のように分類される.

$f(g) \leq quality(OP^j)$: 見込みのある (promising) ゴール節
$f(g) > quality(OP^j)$: 見込みの無い (unpromising) ゴール節

前者は推論を押し進めても評価値がそれほど悪くならなかったことを意味しており, そのゴール節から最適な説明が導出される見込みが高いことが期待できる. このようなゴール節は次の反復でも積極的に推論処理されるべきであり, 質の悪いワークリストを持つプロセッサに優先的に分配される. 一方後者に分類されるゴール節は, 推論を進めるとすぐに評価値が悪くなったことを意味する. このようなゴール節は次の反復では処理されない可能性が高いと考えられるので, 従来の方式に従って分配先のプロセッサを決定する. 図 4.17にこの負荷分散方法のアルゴリズムを示す.

本手法の効果については 4.5.3節で述べる.

4.4 関連研究

ここで関連研究について簡単に述べる.

演繹推論の並列化に関しては, 代表的な演繹推論機構である Prolog の OR-並列化について多くの研究報告がされている (文献 [LWH+90],[AK90] など). Prolog が形成

procedure *dynamic_work_distribution* $(SG, dist[], \{OP^1, \cdots, OP^n\})$

```
1 begin
2   k := j + 1;
3   l := j + 1;
4   while (∃g ∈ SG^j)
5     begin
6       t := quality(OP_i^j);
7       if (f(g) ≤ t) then          % 見込みのある (promising) ゴール節
8         begin
9           for any m such that
                m ≥ l and quality(OP_{i-1}^{(m mod n)}) ≥ t
10            begin
11              g を dist[m mod n] に追加する;
                  (ゴール節 gはプロセッサ (m mod n) に分配される)
12              OP_{i-1}^{(m mod n)} := OP_{i-1}^{(m mod n)} ∪ {g};
13              l := m;
14            end
15        end
16      else                        % 見込みの無い (unpromising) ゴール節
17        begin
18          g を dist[k mod n] に追加する;
              (ゴール節 gはプロセッサ (k mod n) に分配される)
19          OP_{i-1}^{(k mod n)} := OP_{i-1}^{(k mod n)} ∪ {g};
20          k := k + 1;
21        end
22      SG^j := SG^j \ {g};
23    end
24 end.
```

図 4.17: 動的な負荷分散方法

する探索空間が深さ優先のSLD反駁であるのに対して，本稿が対象とする仮説推論ではコスト最小の説明が要求されるため，幅優先で反駁木が形成される．このため探索の並列性はより高く，探索空間の分散が容易に行なえる．各プロセッサは割り当てられた部分空間に対して最良優先探索をおこなう．

文献[HD89]ではA*アルゴリズムを並列化したPIA*アルゴリズムが提案されている．PIA*ではi回目の反復で各プロセッサjは優先度付き待ち行列WL_i^j(work list)（PARCARにおける$OPEN_i^j$に相当）に含まれるノードの中で評価値が閾値t^i以下のノードに対して1段階の導出を行なう．ここでt^iの値はすべてのプロセッサが持つ$WL_i^{j(1\leq j\leq n)}$に含まれるノードの評価値の最小値をとるのでA*アルゴリズムを単純に並列化したものとなっている．また，各プロセッサで展開されたノードの分散はPARCARと同様の方式をとっている．

各プロセッサによって展開されるノードの数は，PIA*のノード展開の手続きより均等になる．しかしながら，一般的にはノードによって展開時に生成される子ノードの数は異なる．例えば図4.1の例題に対してPIA*では2回目の反復においてプロセッサ1はゴール節$\leftarrow a(X), c(Y)$，プロセッサ2はゴール節$\leftarrow b(X), d(Y)$を展開する（図4.2の$\leftarrow p(X)$に対するSLD反駁木の深さ2～3の展開）．プロセッサ1で生成される子ノード数は1であるのに対し，プロセッサ2は3個の子ノードを生成する．本仮説推論においては，子ノード生成は述語のユニファイ，変数の束縛情報の伝播などの処理を伴いその計算量は無視できないものとなり，生成される子ノードの数の差[6]がPIA*の負荷分散を結果的に不均衡にする要因となる．

プロセッサの処理の単位を生成される平均ノード数とした場合の各プロセッサの粒度は，PIA*では細かい[7]．したがってプロセッサ間の通信回数が増大し，台数を増やした場合に通信コストがボトルネックとなり得る．一方，PARCARでは粒度はパラメータKにほぼ等しく，Kの値を調節することによってそのばらつきを押えつつ，通

[6]実験で扱った例題(4.5.2節参照)では1個のゴール節展開につき最大6となる．
[7]4.5.2節で行った実験ではPIA*の粒度は2.1となった．

信コストを軽減している．

また，本仮説推論ではゴール節展開時において仮説を入力節とした導出が行なわれない限り，生成されたサブゴール節の評価値は親のゴール節の評価値と同じ値をとる．例えば図4.2の$\leftarrow p(X)$に対するSLD反駁木において，ゴール節$\leftarrow a(X), c(Y)$とそのサブゴール節$\leftarrow e(X), c(Y)$の評価値は$f(\leftarrow a(X), c(Y)) = 0$, $f(\leftarrow e(X), c(Y)) = 0$と等しい．このような場合は，$i+1$回目の反復での閾値$t^{i+1}$は$t^i$と等しくなり，PIA*の閾値更新手続きは無駄な処理となる．

4.5 PARCARの評価

本節では，並列仮説推論システムPARCARの性能を2つの側面から評価する．まず，4.5.1節でPARCARの定量的な解析を試みる．そして4.5.2節では仮説推論の応用の一つである診断問題から一例題を取り挙げ，実際の並列計算機を用いた実験結果について述べる．PARCARの並列仮説推論手法に対して，前者ではプロセッサ間通信，後者では特にゴール節展開手続きにおける負荷分散において有効性を確認するものである．

4.5.1 PARCARの定量的解析

本節ではPARCARの定量的な解析を試みる．ここでは，搭載する各プロセッサに対して次の仮定がなされた並列計算機を用いてPARCARの計算時間·速度向上比について評価し，前節で述べたPARCARの有効性を確認する．

仮定 4.5.1 SLD導出処理に関する仮定

- アトム間のユニファイにかかる計算時間はアトムの述語名·引数などに関わらずC_{unify}と一定である．
- ゴール節と入力節の導出にかかる計算時間は節の形状に関わらずC_{res}と一定である．

■

従って，Rを知識ベースのルール数，C_{match}を一つのゴール節に対するSLD導出で生成されるサブゴール節の数とすると，一つのゴール節に対する一段のSLD導出にかかる計算時間は

$$T_{sld}(1) = R \cdot C_{unify} + C_{match} \cdot C_{res}$$

となる．C_{match}はゴール節および知識ベースの形に依存し変動するが，ここでは対象とするすべてのゴール節に対してC_{match}が一定となるような知識ベースを考える．

仮定 4.5.2 無矛盾性検査処理に関する仮定

- ゴール節と無矛盾性制約節の一組の無矛盾性検査の計算時間はC_{cc}と一定である．

■

従って，R_{cc}を知識ベースに含まれる無矛盾性制約節数とすると，C_{match}個のゴール節に対する無矛盾性検査の計算時間は

$$T_{cc}(C_{match}) = C_{match} \cdot C_{cc} \cdot R_{cc}$$

となる．

仮定 4.5.3 プロセッサ間のメッセージ通信に関する仮定

- 一つのデータを送信するのにかかる時間はプロセッサ間の距離データ長に無関係にC_{comm}と一定である． ■

仮定 4.5.4 集合計算に関する仮定

- 集合 $OPEN$,$CLOSED$ および SG に挿入される要素 e は平均してそれぞれC_{mo},C_{mc}およびC_{ms}番目の要素である． ■

例えば集合 $OPEN$ にK個のゴール節をマージするのに必要な計算時間は

$$T_{merges}(OPEN, K) = K \cdot C_{mo}$$

となる．

以上の仮定の下で，T_{exp}を「ゴール節展開」手続きの実行時間，T_{comm}を「サブゴール節分散」「サブゴール節受信」両手続きの実行時間とすると，各プロセッサの一回

の反復あたりの実行時間T^1_{PARCAR}は次の式で表される．

$$T^1_{\mathrm{PARCAR}} = T_{exp} + T_{comm}(K)$$

PARCAR ではプロセッサは一回の反復あたりK個のサブゴール節を生成する．一つのゴール節はC_{match}個のサブゴール節を生成するので，ゴール節展開手続きでは一段の SLD 反駁がK/C_{match}回行われる．従ってT_{exp}は以下のようになる．

$$T_{exp} = K/C_{match} \cdot (T_{sld}(1) + T_{cc}(C_{match})) + T_{merges}(CLOSED, K/C_{match}) \\ + T_{merges}(SG, K)$$

また,「サブゴール節分散」「サブゴール節受信」手続きでは，各プロセッサはK個のゴール節を送受信するので，$T_{comm}(K) = C_{comm} \cdot n + K \cdot C_{mo}$ となり，T^1_{PARCAR}は以下の式となる．

$$T^1_{\mathrm{PARCAR}} = K/C_{match} \cdot (T_{sld}(1) + T_{cc}(C_{match}) + C_{mc}) + K \cdot C_{ms} \\ + C_{comm} \cdot n + K \cdot C_{mo}$$

一方で，PIA* では，一回の反復で展開されるゴール節は 1 個であり一回の SLD 反駁が行われる．従って生成されるサブゴール節はC_{match}個であり，プロセッサ間で送受信されるゴール節の数もC_{match}個となるので，PIA* では一回の反復あたりの実行時間は以下のようになる．

$$T^1_{\mathrm{PIA^*}} = T_{sld}(1) + T_{cc}(C_{match}) + C_{mc} + C_{match} \cdot C_{ms} + C_{comm} \cdot n + C_{match} \cdot C_{mo}$$

ここで，PARCAR が解を得るまでに展開するゴール節の数を Gs とし，プロセッサ台数を n とすると，各プロセッサは Gs/n 個のゴール節を展開するので反復回数I_{PARCAR}は $(Gs \cdot C_{match})/(n \cdot K)$ となり，PARCAR の総計算時間T_{PARCAR}(Gs,n) は次の式で表される．

$$T_{\mathrm{PARCAR}}(Gs, n) = \sum^{I_{\mathrm{PARCAR}}} T^1_{\mathrm{PARCAR}} \\ = Gs/n(T_{sld}(1) + T_{cc}(C_{match}) + C_{match}(C_{mo} + C_{ms}) \\ + C_{mc} + C_{match}/K \cdot C_{comm} \cdot n)$$

一方でPIA* では反復回数$I_{\mathrm{PIA}*}$はGs/nとなり，同アルゴリズムの総計算時間T_{PIA^*}(Gs,n)は以下のようになる．

$$\begin{aligned} T_{\mathrm{PIA}^*}(Gs,n) &= \sum^{I_{\mathrm{PIA}*}} T^i_{pia^*} \\ &= Gs/n(T_{sld}(1) + T_{cc}(C_{match}) + C_{match}(C_{mo} + C_{ms}) \\ &\quad + C_{mc} + C_{comm} \cdot n) \end{aligned}$$

ここで，

$$A = T_{sld}(1) + T_{cc}(C_{match}) + C_{match} \cdot (C_{mo} + C_{ms}) + C_{mc}$$

とすると両アルゴリズムの総計算時間の比は，

$$\frac{T_{\mathrm{PIA}^*}(Gs,n)}{T_{\mathrm{PARCAR}}(Gs,n)} = \frac{K(A + C_{comm} \cdot n)}{K \cdot A + C_{match} \cdot C_{comm} \cdot n}$$

となる．C_{comm}の係数に着目するとKがC_{match}に比べて大きいほど[8]PARCAR の方がPIA* よりも効率的となることが予想される．

また，$T_{\mathrm{PIA}^*}(Gs,n)$，$T_{\mathrm{PARCAR}}(Gs,n)$ は以下のように，両アルゴリズムの並列性に関する線形要素 (両式の第1項) および逐次要素 (第2項) の和の形で構成される．

$$\begin{aligned} T_{\mathrm{PARCAR}}(Gs,n) &= Gs \cdot A/n + Gs \cdot C_{comm} \cdot C_{match}/K \\ T_{\mathrm{PIA}^*}(Gs,n) &= Gs \cdot A/n + Gs \cdot C_{comm} \end{aligned}$$

両アルゴリズムとも，「ゴール節展開手続き」で実行されるSLD 導出および無矛盾性検査関係の処理は線形要素，「サブゴール節分散」で実行されるプロセッサ間通信は逐次要素となっている．両式の第2項を比較するとKがC_{match}に比べて大きいほどPARCAR の逐次要素は少なくなり速度向上比においてもPIA* よりも優れることが予想される．これは，パラメータKの値を適切に調節することによりPARCAR がプロセッサ間通信のオーバーヘッドを軽減できることを示している．

なお，本節ではプロセッサ間通信のオーバーヘッドを中心に考察するために，一段のSLD 導出において生成されるサブゴールの数C_{match}は一定であると仮定した．し

[8]PARCAR では与えられた知識ベースに応じて$K \gg C_{match}$となるようにパラメータKの値を設定している．

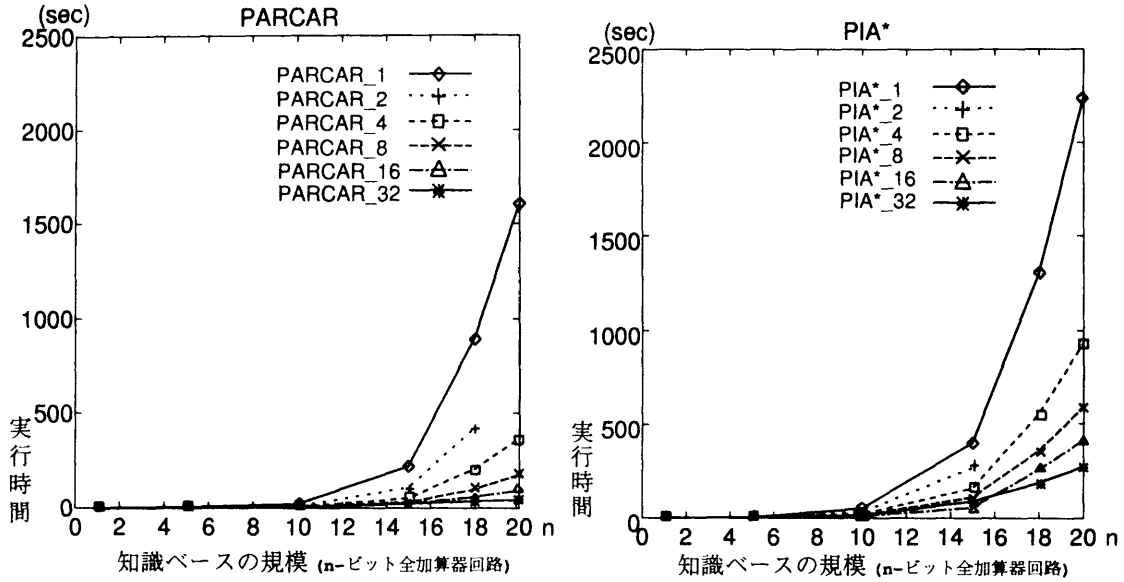


図 4.18: 実験結果

かしながら，実際にはこの数はゴール節の形状に依存して大きく異なる[9]．このような場合には，PIA* では，一回の反復で生成されるサブゴール節はC_{match}個であり，プロセッサ間で送受信されるゴール節の数もC_{match}個となっているので，プロセッサ間での$T^1_{\text{PIA*}}$の実行時間に大きなばらつきが生じ負荷分散に悪影響を及ぼすことが懸念される．一方で PARCAR では，一回の反復で生成されるサブゴール節はC_{match}の値に関わらずKと一定にすることにより，負荷分散に及ぼす影響を少なくしている．

4.5.2 実験結果

本論文で提案した並列化の有効性を確認するために，推論時間の比較実験を行なった．例題としては，論理回路の故障診断問題を用いた．回路はn-ビット全加算器を対象にした．$n = 1$ の場合の例題知識ベースを付録 Dに示す．回路を構成する素子は「正常」「開放」「短絡」の 3 つの状態をとるものとし，これを仮説として表現した．また，各仮説のコストは，各素子が上記の状態となる確率$\mathcal{P}$を考え，$-log_e\mathcal{P}$で与

[9]例えば，4.5.2節で扱う例題についてはゴール節の最左アトムに依存して$0 < C_{match} < 6$の値となる．

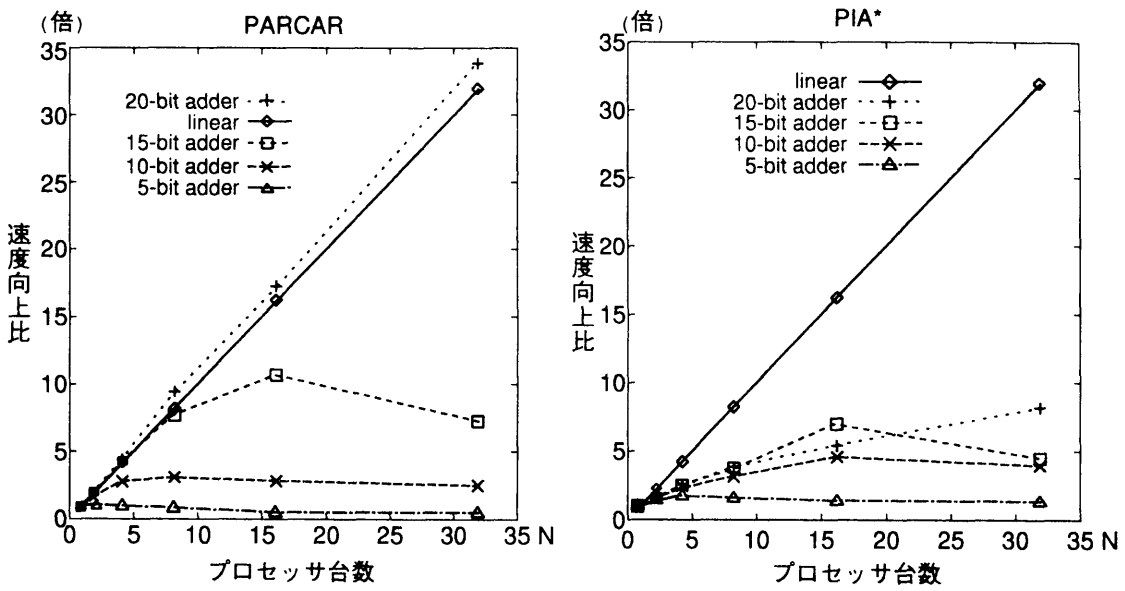


図 4.19: 速度向上比

表 4.1: 問題規模 $n = 20$ における実験結果の詳細

		PARCAR				PIA*			
PE 数		32	16	8	4	32	16	8	4
実行時間	(sec)	47	94	174	366	275	416	593	931
反復回数		15	40	62	120	571	733	925	1137
$\|CLOSED\|$		24508	23653	23275	23437	22473	22978	23046	23121
$\overline{Aber}(T_{exp})$	(%)	55.8	12.6	7.7	8.3	71.4	57.3	48.5	45.5
$\overline{T}_{comm}$	(sec)	1.089	1.241	1.457	1.740	0.588	0.552	0.592	0.774

えた[10]．n をパラメータとして n-ビット全加算器回路の規模を変化[11]させて実験を行なった．実験は富士通 AP1000 上で C 言語を用いて行なった．AP1000 は MIMD 型分散メモリアーキテクチャの並列計算機である．実験結果を図 4.18に示す．プロセッサ数は 1 台のみ~32 台まで用いた．同図左が PARCAR を用いた結果，右が PIA* を用いた手法の結果をそれぞれ示している．PARCAR 実行時に与えるパラメータ Kの値は 50 とした．両手法とも，探索木に現れるゴール節 gのコストの予測値（定義 4.2.3 参照）は$\hat{h}(g) = 0$ とした．知識ベースの規模が増大するほど，PARCAR が PIA* を用いた手法に比べて優れていることがわかる．特に問題規模 $n = 20$, プロセッサ台数

[10]確率の積$\Pi_i P_i$が正の数の和$\Sigma_i(-log_e P_i)$ で表現でき，確率最大の説明を求める問題がコスト最小の説明を求める問題に帰着される．

[11]この問題を表現する知識ベースの規模は n に比例し，事実の知識は $17n + 23$ の述語論理ホーン節，仮説は $15n + 3$ の単位節から成る．

32 において本システムは PIA* を用いた手法の約 5.8 倍の推論速度が達成されている．図 4.19に両手法の速度向上比を示す．PIA* を用いた手法では並列化による台数効果は問題規模の大小に関わらずプロセッサ数の少ない時点から線形より大きく外れるが，PARCAR では問題規模が大きくなるにつれてプロセッサ台数を増やした場合でもほぼ線形の台数効果が得られているのが分かる．

次に，問題規模 $n = 20$ における実験結果の詳細を表 4.1に示す．同表にて，$|CLOSED|$ は両アルゴリズムが最適解を見つけるまでに展開したゴール節の数を示しており，探索空間の大きさの程度を表している．T_{exp}およびT_{comm}は両アルゴリズムの一回の反復におけるゴール節展開手続きおよびプロセッサ間通信[12]の処理時間であり，$\overline{T}_{exp}$および$\overline{T}_{comm}$は全プロセッサでのそれらの計算時間の平均時間をそれぞれ示している．さらに，T_{exp}については，各反復毎に全プロセッサに対する標準偏差 $Dev(T_{exp})$ を求め，$\overline{T}_{exp}$に対する比を調べた．$\overline{Aber}(T_{exp})$ は $Aber(T_{exp}) = \frac{Dev(T_{exp})}{\overline{T}_{exp}} \times 100$ の総反復での平均値であり，各プロセッサでのゴール節展開手続きの処理時間の平均時間に対するばらつきの度合を示す．両アルゴリズムとも，全プロセッサにゴール節が行き渡った以降の反復について測定を行なった．

表 4.1中 $|CLOSED|$ より，PARCAR と PIA* は共に同程度の大きさの探索空間を費やしており，両者の実行時間の優劣はアルゴリズムの効率に起因していると考えられる．そこで，まず両者のプロセッサ間通信の処理時間について調べた．表 4.1の結果より，一回の反復において，PARCAR は PIA* の 1.9 ～ 2.3 倍の通信時間を費やしているが，反復回数では 1/38 ～ 1/10 の回数に抑えられている．各プロセッサでのプロセッサ間通信の平均時間は$\overline{T}_{comm}$ × 反復回数であることから，PARCAR は各プロセッサの処理粒度を適度に大きくすることによりプロセッサ間の通信回数を抑え，プロセッサ間通信のコストを軽減していることがわかる．

また，アルゴリズムの並列性を向上させるには，各反復における各プロセッサのゴール節展開手続きの処理時間のばらつきを抑え，各プロセッサの負荷を出来るだけ均等

[12]サブゴール節分散およびサブゴール節受信手続きの処理時間の和である．

にすることが重要である．表4.1中$\overline{Aber}(T_{exp})$より，例えばプロセッサ数8においては，PARCARではゴール節展開手続きの処理時間が$\overline{T}_{exp}$±7.7%以内にほぼ収まっているのに対して，PIA*では同手続きの処理時間のばらつきが$\overline{T}_{exp}$の45%を超える結果となっている．したがって，各プロセッサの負荷分散においても，PARCARはPIA*よりも優れていることがわかる．

4.5.3 Dynamic Work Distributionの効果

4.3.3節で述べた動的な負荷分散方法の効果を確認するために前節で用いた全加算器回路故障診断問題で問題規模20，プロセッサ台数32の実験において更にいくつかの項目を測定した．

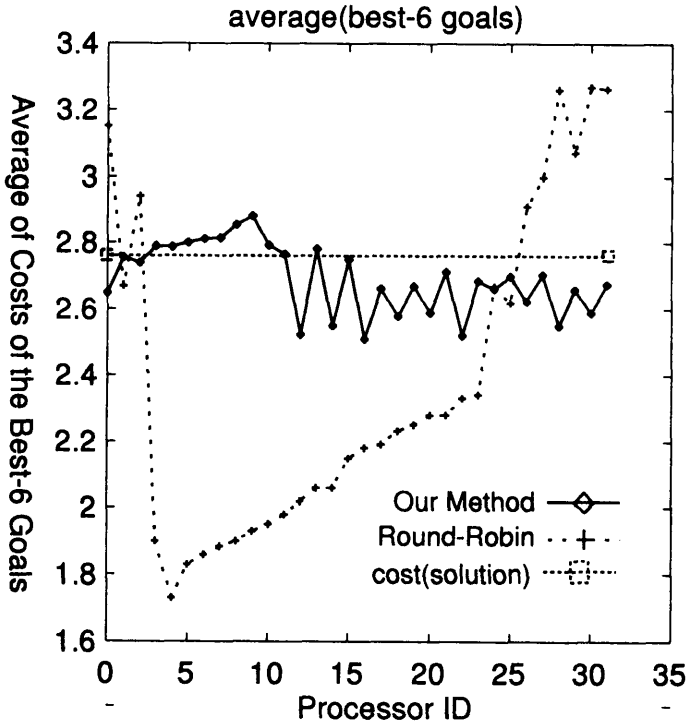


図4.20: 最終反復における各プロセッサのワークリスト $OPEN^j$の質 (問題規模 :$m = 20$, プロセッサ台数:32)

図4.20は確認フェイズへの以降時における各プロセッサが持つワークリスト*OPEN*の質を，図4.5.3は同じくワークリストの中の最良なゴール節の評価値を示している．図中にて"Round-Robin" はプロセッサ間通信の処理 (サブゴール節分散，サブゴール

節受信手続き) に図 4.11および図 4.12のアルゴリズムを用いた場合である．両図とも"Round-Robin" に比べ，動的な負荷分散方法の方が全体的に解のコストに近い．これは続く確認フェイズにおける説明のコスト最小性の検査処理が全体的に速くなることを意味している．

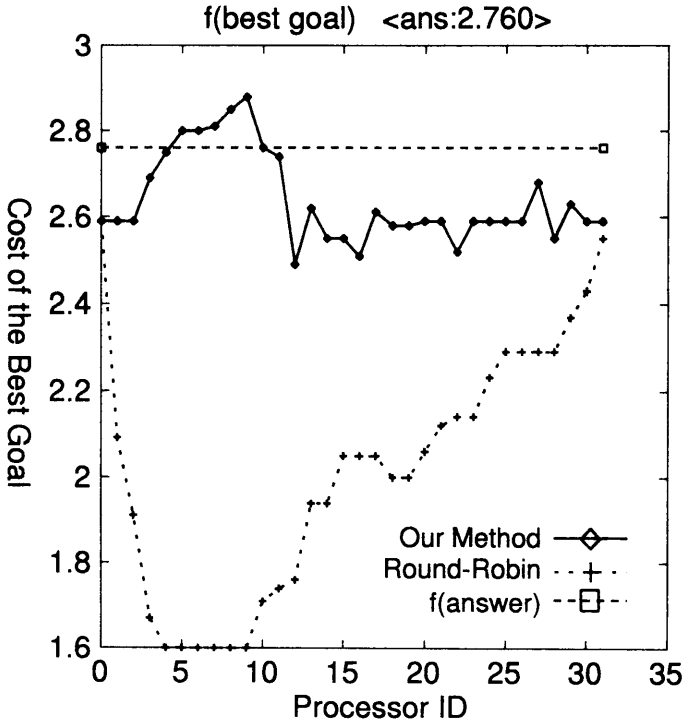


図 4.21: 最終反復における各プロセッサのワークリスト内のゴール節の評価値の最小値 (問題規模:$m = 20$, プロセッサ台数:32)

PARCAR では各反復における各プロセッサの処理粒度は進行フェイズにおいてはパラメータ Kの値に比例するが，確認フェイズ (最終反復) においては最適な説明候補のコストを閾値にして推論を行う (4.3.2節参照) ので各プロセッサの処理粒度は $OPEN$ の状態に依存する．したがって各プロセッサの最終反復の処理時間は負荷分散方法の効果を示すものとなる．図 4.22は各プロセッサの確認フェイズの処理時間を示している．同図より，動的な 4.3.3節で提案した負荷分散方法を用いることにより，各プロセッサでの確認フェイズ処理時間が速くなっていることがわかる．PARCAR 全体のパーフォマンスは一番遅いプロセッサの処理時間に等しいので，問題規模 20, プロセッサ台数 32 の場合には本方式を用いることで確認フェイズの処理は約 3.6 倍の速

さとなっている.

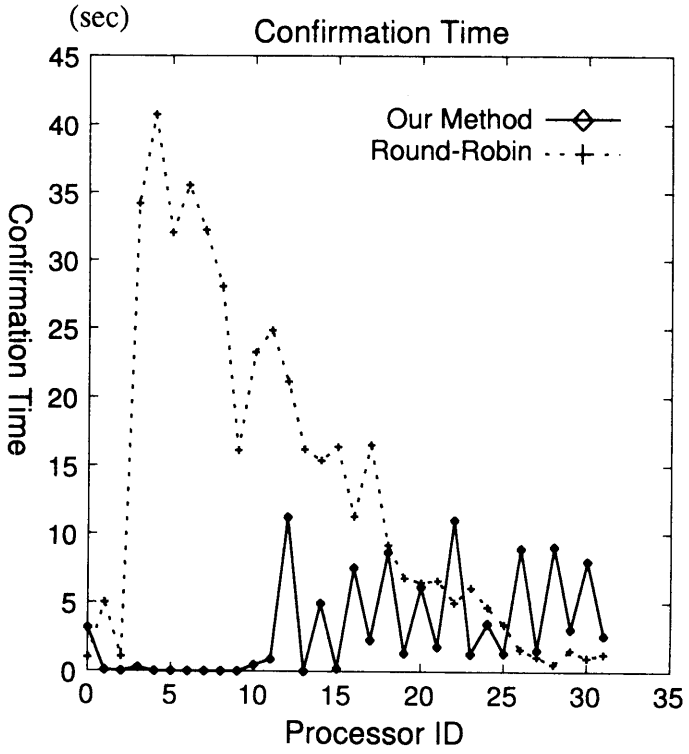


図 4.22: 確認フェイズの処理時間 (問題規模:$m = 20$, プロセッサ台数:32)

4.6 おわりに

コストに基づく仮説推論においてその最適解探索法の並列化を提案し，並列仮説推論システム PARCAR を実現した．本稿で提案した仮説推論システムを並列計算機 AP1000 上に実装し, 有効性を確認した.

本稿では，仮説推論における最適解探索の並列化について探索空間の分散方法を中心に本並列仮説推論システムを説明した．本システムが再帰的な知識ベースを扱うときに必要となるループ·チェック機能については，全プロセッサ間でのパイプライン処理を用いてこれを実現している.

また，4.5.2節で行った実験では，本並列仮説推論システムの並列性を評価する目的のために，各プロセッサが生成する探索木に現れるゴール節 gのコストの予測値（定義 4.2.3参照）は$\hat{h}(g) = 0$ とした．これに関しては，前章 3.3.2節で提案した知識ベー

スの事前解析を用いたヒューリスティック評価関数$\hat{h}$の算出方法を PARCAR に導入することでトータルのパーフォマンスはさらに向上する.

第5章
結論

本論文では，特に知識ベースシステムの推論処理の高機能化として仮説推論に着目し，その推論処理の高速化技法についてさまざまな側面から検討を行なった．

(1) プログラム解析に基づく仮説推論の高速化

まず始めに，一階述語論理を用いた仮説推論の高速化の一方法として，プログラム解析に基づく無矛盾性の検査の効率化の提案を行なった．本論文で提案したプログラム解析法は，ホーン節述語論理で表現された知識ベースを命題論理のレベルに抽象化することにより，与えられた問合せと知識ベースの論理的制約に対して，実際の推論において評価すべき探索空間を，近似的に求めるものである．この解析法を用いて，仮説推論における制約節の評価時において，無駄な探索空間の絞り込みを行なうような，また，マジックセット法に基づきつつ，それをプログラム解析結果を利用できるような形に拡張した方式により仮説推論システムを実装し，その有効性を確認した．

(2) コストに基づく仮説推論における推論制御の最適化

つぎに，コストに基づく仮説推論において効率的な最適解探索法を提案し，一階述語論理ホーン節を対象とする仮説推論システムを実現した．本論文で提案した仮説推論システムは，論理プログラミングの分野における問合せ処理技術に対して A*アルゴリズムの持つ探索制御技術を導入することにより，与えられた観測に対して最良の説明を効率的に求めるものである．本論文で導入した A*アルゴリズムを代表とするヒューリスティックな探索法においては，その探索制御能力はヒューリスティック評価関数$\hat{h}$に大きく依存する．そこで本論文では，実行可能条件を満たし，かつ，実際

のコスト h になるべく近い値をとるような，ヒューリスティック評価関数を求めるために，プログラム解析を用いた $\hat{h}$ の導出方法を提案した．さらに，本論文で提案した仮説推論システムを前向き推論を用いて実装し，その有効性を確認した．

(3) PARCAR: コストに基づく並列仮説推論システム

最後に，実用化に耐える規模の知識データベースシステムが要求する計算能力にこたえるためには並列処理が欠かせないとの立場にたち，仮説推論の探索空間を複数のプロセッサへ分散し推論処理を並列化することにより，高速な並列仮説推論システムを実現した．仮説推論の推論処理技術に並列ヒューリスティック探索が持つ探索制御技術を導入し，与えられた観測を説明する最小コストの仮説集合を効率的に求めるものである．本稿で提案した並列仮説推論システムを並列計算機 AP1000 上に実装し，有効性を確認した．

今後の課題としては，次のような問題が挙げられる．現在実現されている多くの仮説推論においては，扱うべき仮説は，不完全ではあるが背景的な知識として知識ベースに予めに与えられている [Poo88]．これらのシステムでは，知識ベースに現れない知識を仮定するなど，人間が持つ高度な知能活動の一つである「発想」の概念を本質的に実現するような知識処理はできない．そこで，予めに与えられた背景知識からは説明が得られない場合に，仮説を生成することにより，欠落した知識を補い観測を説明する「発想」の枠組への発展が期待されている [井上 92]．

本研究では，欠落した知識を補うための手法として，帰納推論に基づく機械学習 (帰納学習) が有効であると考えている．更に近年，帰納学習と論理プログラムを結び付ける研究として「帰納論理プログラム」[Mug92] と呼ばれる分野が人工知能の分野において注目を集めている [川村 93]．そこで今後の課題として，仮説推論の「発想」の枠組への発展のために，仮説推論と帰納論理プログラムの融合について研究を行うことが挙げられる．

また次世代の情報処理技術においては，今日のインターネットに代表されるような計算機ネットワーク環境の発達に伴い，膨大な量の知識がその属性や性質に応じて複数の計算機に分散されて管理されるようになる．そしてそれぞれの計算機が協調して全体として一貫性のある知識処理がおこなわれるような分散知識データベースシステムが要求されるものと考える．そこで今後の課題として，仮説推論の柔軟な推論処理能力を用いた分散知識データベースシステムの実現について検討することがあげられる．

謝 辞

本研究の機会を与えて下さり，本研究をすすめるにあたり終始適切なご指導をいただいた 伊藤 英則 教授 ならびに 世木 博久 教授 に深く感謝致します．また，本研究について多くの助言を下さいました，研究室の皆様に厚く御礼申し上げます．

参考文献

[AK90] Khayri A. M. Ali and Roland Karlsson. The Muse Or-Parallel Prolog Model and its Performance. In *Proc. of the North American Conf. on Logic Programming*, pp. 757–776. The MIT Press, 1990.

[BMSU86] F. Bancilhon, D. Maier, U. Sagiv, and J. D. Ullman. Magic Sets and Other Strange Ways to Implement Logic Programs. In *Proc. Fifth ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems*, pp. 1–15, 1986.

[CH91] E. Charniak and S. Husain. A New Admissible Heuristic for Minimal-Cost Proofs. In *Proc. of the AAAI-91*, pp. 446–451, 1991.

[CS90] E. Charniak and S. E. Shimony. Probabilistic semantics for cost based abduction. In *Proc. of the AAAI-90*, pp. 106–111, 1990.

[CS94] E. Charniak and S. E. Shimony. Cost-based abduction and MAP explanation. *Artificial Intelligence*, Vol. 66, pp. 345–374, 1994.

[DP85] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *J. Assoc. Comput. Math.*, Vol. 32, No. 3, pp. 505–536, 1985.

[HD89] S. Huang and L. S. Davis. Parallel Iterative A^* Search: An Admissible Distributed Heuristic Search Algorithm. In *Proc. of the IJCAI-89*, pp. 23–29, 1989.

[KKSI96] S. Kato, C. Kamakura, H. Seki, and H. Itoh. PARCAR: A Parallel Cost-based Abductive Reasoning System. In *Proc. of the 9th Intl. Conf. on*

Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, pp. 57–62, Fukuoka, June 1996.

[KMI91] A. Kondo, T. Makino, and M. Ishizuka. An Efficient Hypothetical Reasoning System for Predicate-logic Knowledge-base. In *Proc. IEEE, Intl Conf. on Tools for AI*, pp. 60–67, San Jose, 1991.

[Kor85] R. E. Korf. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, Vol. 27, pp. 97–109, 1985.

[KSI93a] S. Kato, H. Seki, and H. Itoh. A Deductive Database Approach to Abductive Reasoning. In *Proc. of the 3rd Intl. Symposium on Next Generation Database Systems and Their Applications*, pp. 77–84, Fukuoka, September 1993.

[KSI93b] S. Kato, H. Seki, and H. Itoh. An Efficient Abductive Reasoning System Based on Program Analysis. In *P. Cousot et al. eds. Static Analysis, Proc. of the 3rd Intl. Workshop, Lecture Notes in Computer Science*, Vol. 724, pp. 230–241, Padova, September 1993. Springer-Verlag.

[KSI94a] S. Kato, H. Seki, and H. Itoh. Cost-based Horn Abduction and its Optimal Search. In *Proc. of the 3rd Intl. Conf. on Automation, Robotics and Computer Vision*, pp. 831–835, Singapore, November 1994.

[KSI94b] S. Kato, H. Seki, and H. Itoh. Cost-based Horn Abduction to Focus on the Most Probable Diagnosis. In *Proc. of the 5th Intl. Workshop on Principles of Diagnosis*, pp. 148–152, New Paltz, NY, October 1994.

[KSI96] S. Kato, H. Seki, and H. Itoh. Parallel Cost-Based Abductive Reasoning for Distributed Memory Systems. In *N. Foo and R. Goebel eds. PRICAI'96: Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence*, Vol. 1114, pp. 300–311, Cairns, August 1996. Springer-Verlag.

[KSI97] S. Kato, H. Seki, and H. Itoh. A Parallel Implementation of Cost-Based Abductive Reasoning. In *Proc. of the second Intl. Symp. on Parallel Symbolic Computation (PASCO'97)*, pp. 111–118, Maui, Hawaii, July 1997. ACM press.

[Llo84] J. W. Lloyd. *Foundations of Logic Programming.* Springer, 1984. Second, extended edition, 1987.

[LWH+90] E. Lusk, David H. D. Warren, S. Haridi, et al. The Aurora or-parallel Prolog system. *New Generation Computing*, Vol. 7(2,3), pp. 243–271, 1990.

[Mug92] S. Muggleton. *Inductive Logic Programming.* Academic Press, 1992.

[Nil80] N. J. Nilsson. *Principles of Artificial Intelligence.* Tioga, Palo Alto, CA, 1980.

[NM91] H. T. Ng and R. J. Mooney. A Efficient First-Order Horn-Clause Abduction System Based on the ATMS. In *Proc. of the AAAI-91*, pp. 494–499, 1991.

[OI92] Y. Ohta and K. Inoue. A Forward-Chaining Hypothetical Reasoner Based on Upside-Down Meta-Interpretation. In *Proc. of the Intl. Conf. on FGCS '92*, pp. 522–529, 1992.

[OI93] Y. Ohta and K. Inoue. Incorporating Top-Down Information into Bottom-Up Hypothetical Reasoning. *New Generation Computing*, Vol. 11, pp. 401–421, 1993.

[Poo88] D. Poole. A Logical Framework for Default Reasoning. *Artificial Intelligence*, Vol. 36, pp. 27–47, 1988.

[Poo92] D. Poole. Logic Programming, Abduction and Probability. In *Proc. of the Intl. Conf. on FGCS '92*, pp. 530–538, 1992.

[Poo93] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial*

Intelligence, Vol. 64, pp. 81–129, 1993.

[Pri93] A. E. Prieditis. Machine Discovery of Effective Admissible Heuristics. *Machine Learning*, Vol. 12, pp. 117–141, 1993.

[RSS92] R. Ramakrishnan, D. Srivastava, and S. Sudarshan. Controlling the search in bottom-up evaluation. In *Joint Intl. Conf. and Symposium on Logic Programming*, pp. 273–287, 1992.

[Sek89] H. Seki. On the Power of Alexander Templates. In *Proc. Eighth ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems*, pp. 150–159, 1989.

[Sti91] M. E. Stickel. Upside-down meta-interpretation of the model elimination theorem-prover procedure for deduction and abduction. Technical Report TR-664, ICOT, January 1991. TR-664.

[TS86] H. Tamaki and T. Sato. OLD Resolution with Tabulation. In *Proc. of the 3rd ICLP*, pp. 84–98, London, 1986.

[Vie86] L. Vieille. Recursive Axioms in Deductive Databases: The Query/Subquery Approach. In *Proc. First Intl. Conf. on Expert Database Systems*, pp. 179–193, Charleston, 1986.

[井上 92] 井上克巳. アブダクションの原理. 人工知能学会誌, Vol. 7, No. 1, pp. 48–59, January 1992.

[加藤 94] 加藤昇平, 世木博久, 伊藤英則. プログラム解析に基づく仮説推論の高速化技法. 情報処理学会論文誌, Vol. 35, No. 10, pp. 2019–2028, October 1994.

[加藤 95] 加藤昇平, 世木博久, 伊藤英則. コストに基づく仮説推論における最適解探索の一方法. 情報処理学会論文誌, Vol. 36, No. 10, pp. 2380–2390, October 1995.

[加藤 96] 加藤昇平, 世木博久, 伊藤英則. コストに基づく仮説推論の並列化. 並列処

理シンポジウム JSPP'96, pp. 169–176, June 1996.

[宮崎 90] 宮崎収兄, 世木博久. 演繹データベースの問合せ処理. 情報処理, Vol. 31, No. 2, pp. 216–224, February 1990.

[近藤 94] 近藤朗子, 石塚満. 述語論理知識を扱う仮説推論における最適解の高速推論法. 人工知能学会誌, Vol. 9, No. 2, pp. 110–118, March 1994.

[石塚 88] 石塚満. 不完全な知識の操作による次世代知識ベースへのアプローチ. 人工知能学会誌, Vol. 3, No. 5, pp. 522–562, September 1988.

[川村 93] 川村正. 帰納論理プログラミング –論理プログラムの一般化を中心に–. コンピュータ・ソフトウェア, Vol. 10, No. 5, pp. 387–398, September 1993.

[太田 91] 太田好彦, 井上克巳. ATMS を用いた前向き仮説推論システムにおける効率的な推論方式. 人工知能学会誌, Vol. 6, No. 2, pp. 247–259, March 1991.

付録 A
命題 2.3.1の証明

定理 2.3.1 Pをプログラム，Pに含まれる事実を F，仮説集合を Hとし，Oを観測とする．Oに対する候補仮説 H_Oが条件 (AR2) を満たしていないとする．このとき，

(i) $P \cup \{\leftarrow false\}$の反駁木において，組 (H_O, H_{false}) が Oに対する矛盾仮説対となるような H_{false}を $false$ の候補仮説として持つ成功葉が必ず存在し，

(ii) (i) が成り立つならば，$(\bar{H}_{\bar{O}}, \bar{H}_{false})$ も$\bar{O}$に対する矛盾仮説対となる．

■

A.1 補題

命題 2.3.1を証明するために，まず次の補題を示す．

補題 A.1.1 Pをプログラム，$\leftarrow G$をゴールとする．$\Delta, \mathcal{H}_g$及びΞを M 演算子付アトムの連言，Γ及びΛをアトムの連言とする．$N_1 = \leftarrow \Delta, g, \Gamma$を$P \cup \{\leftarrow G\}$の OLDT 反駁木におけるあるゴールとし，$g$を選択されたアトムとする．更に，以下の条件が成り立つと仮定する．

- N_1の反駁における$\leftarrow g$の部分反駁で，$\leftarrow \mathcal{H}_g$を成功葉として持つものが存在する．
- $\bar{P} \cup \{\leftarrow \bar{G}\}$の OLDT 反駁木においてノード $N_2 = \leftarrow \Xi, \bar{g}, \Lambda$で，$\bar{g}$を選択されたアトムとするものが存在する．

このとき，N_2の反駁における$\leftarrow \bar{g}$の部分反駁で，$\leftarrow \bar{\mathcal{H}}_{\bar{g}}$を成功葉として持つものが必ず存在する．

■

証明: 部分反駁の長さに関する帰納法で証明する．ここでは，参照 (lookup) ノードの解テーブル参照による部分反駁の長さは，対応する解ノードに対する部分反駁の長さ

に等しいとする.

1. $\leftarrow g$の部分反駁の長さが1のとき,

   $^{\exists}g_o \leftarrow \in P$かつ$^{\exists}\bar{g_o} \leftarrow \in \bar{P}$（$g_o$は$g$と単一化可能なアトム）であるので，$N_1$及び$N_2$に対する一段階のOLDT導出を考えれば明らかである.

2. $\leftarrow g$の部分反駁の長さが$k+1$のとき,

   $P \cup \{\leftarrow G\}$のOLDT反駁木に現れるすべてのゴールに対して，その部分反駁の長さがk以下の場合については補題が成り立つと仮定する.

   場合 a: N_1が解ノードのとき,

   $^{\exists}g_o \leftarrow A_1, \text{-}, A_n \in P$より，$\leftarrow g$の部分反駁木は，$\leftarrow g$のサブゴールとして$\leftarrow A_1\theta, \cdots, A_n\theta$（$\theta = mgu(g, g_o)$）を持つ．$\leftarrow g$の部分反駁は$\leftarrow A_i\theta (1 \leq i \leq n)$の部分反駁を繋げたものであり，$\leftarrow A_i\theta$の部分反駁の長さはすべて$k$以下である．一方，$\bar{P} \cup \{\leftarrow \bar{G}\}$のOLDT反駁において，次の2つの場合を考える.

   場合 a-i: N_2が解ノードのとき,

   $^{\exists}\bar{g_o} \leftarrow \bar{A}_1, \cdots, \bar{A}_n \in \bar{P}$ より，N_2はサブゴール$\leftarrow \Xi, \bar{A}_1, \cdots, \bar{A}_n, \Lambda$を持つ．ここで，$\leftarrow \bar{A}_i (1 \leq i \leq n)$の部分反駁については，帰納法の仮定より補題が成立する．従って，$\leftarrow \bar{g}$の部分反駁で，$\leftarrow \bar{\mathcal{H}}_{\bar{g}}$を成功葉として持つものが存在する.

   場合 a-ii: N_2が参照ノードのとき,

   N_2に対応する解ノードについては場合a-iと同様のことが成り立つので，N_2についても上の場合と同様である.

   場合 b: N_1が参照ノードのとき,

   N_1に対応する解ノードの部分反駁の長さは$k+1$であるので，場合aと同様のことが成り立つ.

以上より，部分反駁の長さが$k+1$でも補題は成立する. ■

A.2 命題 2.3.1の証明

(i) の証明: 定義 2.1 より，F自身が矛盾することはない．すなわち $F \not\vdash \square$ である．従って，H_Oが条件 (AR2) を満たさないとき，つまり，$F \cup H_O \vdash \square$ となるときは，$F \cup H_O \cup \{\leftarrow false\}$の OLDT 反駁木の成功葉には必ず仮説集合が現れ，任意の候補仮説 H_{false}は H_Oの部分集合となる．ここで，$P \supseteq F \cup H_O$より，$P \cup \{\leftarrow false\}$の反駁において (H_O, H_{false}) が Oに対する矛盾仮説対となる H_{false}が必ず存在する．

(ii) の証明: 補題 1 より，$P \cup \{\leftarrow false\}$及び$\bar{P} \cup \{\leftarrow false\}$の OLDT 反駁木について考えれば明らか． ■

付録 B

定理 3.3.1の証明

定理 3.3.1 Pをプログラム，Oを与えられた観測，A を Pに含まれるアトムとし，H_A を A の最適な説明を表す仮説集合とする．また，$\hat{h}(g)$ を$P \cup \{\leftarrow O\}$の SLD 反駁木に現れるゴール節 $g = \leftarrow \mathsf{M}L_1, \cdots, \mathsf{M}L_i, A_{i+1}, \cdots, A_n$からある成功葉へ至る最適導出のコストの予測値とする．

このとき，式 $(*)$ から得られる予測値$\hat{h}(g)$ は実行可能条件$\hat{h}(g) \leq h(g)$ を満たす．

$$\begin{array}{l} \hat{h}(g) = h(\bar{g}) = cost(H_{\bar{g}}) \\ H_{\bar{g}} = \bigcup_{j=i+1}^{n} H_{\bar{A}_j} \end{array} \qquad (*)$$

■

B.1 補題

定理 3.3.1を証明するために，まず次の 2 つの補題を示す．

補題 B.1.1 Fを事実，Hを仮説集合とする（すなわち，$F \cup H$はプログラム）．このとき，以下の式が成立する．

$$^{\forall}h \in H \text{に対して} \quad ^{\forall}\bar{h} \in \bar{H} \text{かつ} \; cost(\{h\}) \geq cost(\{\bar{h}\})$$

証明: 本事前解析において採用している抽象化の方法より明らか． ■

補題 B.1.2 Pを $F \cup H$からなるプログラム，$\leftarrow G$をゴールとし，Δを M 演算子付アトムの連言とする．ここで以下の条件が成り立つと仮定する．

- $P \cup \{\leftarrow G\}$の SLD 反駁木において，ゴール$\leftarrow \Delta$を導出する
  ような$\leftarrow G$の反駁が存在する．

このとき，$\bar{P} \cup \{\leftarrow \bar{G}\}$の SLD 反駁において，ゴール$\leftarrow \bar{\Delta}$を導出するような$\leftarrow \bar{G}$の反駁が必ず存在する．

SLD 反駁の長さに関する帰納法で証明する.

1. $\leftarrow G$の反駁の長さが 1 のとき

   ${}^{\exists}G_o \leftarrow \in P$より，補題 B.1.1から明らかに${}^{\exists}\bar{G}_o \leftarrow \in \bar{P}$（$G_o$は$G$と単一化可能なアトム）である．従って$\leftarrow G$及び$\leftarrow \bar{G}$に対する一段階の SLD 導出を考えれば明らかである.

2. $\leftarrow G$の反駁の長さがk以下のもとで補題 B.1.2が成り立つと仮定する.

   このとき，長さ$k+1$の反駁について${}^{\exists}G_o \leftarrow A_1, \cdots, A_n \in P$より，$\leftarrow G$の反駁木は$\leftarrow G$のサブゴールとして$\leftarrow A_1\theta, \cdots, A_n\theta\,(\theta = mgu(G, G_o))$を持つ．$\leftarrow G$の反駁は$\leftarrow A_i\theta\ (1 \leq i \leq n)$の反駁を繋げたものであり，$\leftarrow G$の反駁の長さは$k+1$なので，$\leftarrow A_i\theta (1 \leq i \leq n)$の反駁の長さはすべて$k$以下である.

   一方，$\bar{P} \cup \{\leftarrow \bar{G}\}$の SLD 反駁において，${}^{\exists}\bar{G} \leftarrow \bar{A}_1, \cdots, \bar{A}_n \in \bar{P}$より，$\leftarrow \bar{G}$はサブゴール$\leftarrow \bar{A}_1, \cdots, \bar{A}_n$を持つ．ここで，$\leftarrow \bar{A}_i (1 \leq i \leq n)$の反駁については帰納法の仮定より補題 B.1.2が成立する．従って，$\leftarrow \bar{G}$に対して$\leftarrow \bar{\Delta}$を導出する$\leftarrow \bar{G}$の反駁が存在する.

   従って，部分反駁の長さが$k+1$のときも，補題 B.1.2は成立する.

■

B.2 定理 3.3.1の証明

Oを観測，EをHの部分集合とし，H_OをPにおけるOの最適な説明を示す仮説集合とする（この仮定より，$F \cup H_O \vdash O$）．補題 B.1.2より，$F \cup E \vdash O$となるようなすべてのEに対し，${}^{\forall}\bar{E}$もまた$\bar{F} \cup \bar{E} \vdash \bar{O}$となり，補題 B.1.1より，$cost(E) \geq cost(\bar{E})$が成り立つ．従って，任意の$H_O$に対してそれを命題論理に抽象化した$\bar{H}_{\bar{O}}$もまた$\bar{F} \cup \bar{H}_{\bar{O}} \vdash \bar{O}$となり，かつ，$cost(H_O) \geq cost(\bar{H}_{\bar{O}})$の関係を満たす．ここで$H_{\bar{O}}$を$\bar{P}$における$\bar{O}$の最適な説明を示す仮説集合とすると$cost(\bar{H}_{\bar{O}}) \geq cost(H_{\bar{O}})$であるので，$cost(H_O) \geq cost(H_{\bar{O}})$が成立する．よって，ゴール$g = \leftarrow \mathsf{M}L_1, \cdots, \mathsf{M}L_i, A_{i+1}, \cdots, A_n$に対して$\leftarrow A_{i+1}, \cdots, A_n$

及び$\leftarrow \bar{A}_{i+1}, \cdots, \bar{A}_n$の最適な説明をそれぞれ考えれば定理3.3.1は明らかに成立する. ■

付録 C

観測の説明が通常の集合である場合への対応

定義 3.3.1を以下のように変更する.

定義 3.3.1 Pをプログラム, Oを与えられた問合せとし, $g = \leftarrow \mathrm{M}L_1, \cdots, \mathrm{M}L_i, A_{i+1}, \cdots, A_n$ を Pにおけるゴール$\leftarrow O$に対する SLD 反駁木 ($P \cup \{\leftarrow O\}$の SLD 反駁木と呼ぶ) における任意のゴール節とする. このとき, 最適解探索における gの評価値 $f(g)$ は以下の式で与えられる.

$$f(g) = cost(\mathcal{C}_g \cup^* \hat{\mathcal{H}}_g)$$

$$\mathcal{C}_g = \text{集合}\{L_1, \cdots, L_i\}$$

$$\mathcal{H}_g = \text{集合}\bigcup_{j=i+1}^{n} A_j \text{に対する最適な説明}$$

$$0 \leq cost(\mathcal{C}_g \cup \hat{\mathcal{H}}_g) \leq cost(\mathcal{C}_g \cup \mathcal{H}_g)$$

ここで, $cost(\mathcal{C}_g)$ は反駁木の根から gに至る導出におけるコスト, $cost(\hat{\mathcal{H}}_g)$ は gから成功葉に至る最適導出におけるコストの予測値 ($cost(\mathcal{H}_g)$ は実際のコストの値) をそれぞれ示している. ■

上記の変更より, 本推論アルゴリズムの実行可能条件 (定理 3.3.1参照) は $cost(\mathcal{C}_g \cup \hat{\mathcal{H}}_g) \leq cost(\mathcal{C}_g \cup \mathcal{H}_g)$ となる. そこで, この実行可能条件を満たすような集合$\hat{\mathcal{H}}_g$ (定義 3.3.3の $H_{\bar{g}}$に対応) を求めるために本論文で提案した解析方法を変更する. 変更された解析方法の直観的な説明を以下に述べる.

ゴール gに対して, (i) アトム$\bar{A}_j$の各説明から, 仮説集合$\{L_1, \cdots, Li\}$の要素と重複する可能性のある仮説を削除し, (ii) それらの中で最小コストの仮説集合を$\bar{A}_j$の最適な説明とみなすという方法である. その具体的な手続きについて説明する.

$\hat{\mathcal{H}}_g$を求める計算手続き
入力: $\mathcal{C}_g$, $\{H^{all}_{\bar{A}_{i+1}}, \cdots, H^{all}_{\bar{A}_n}\}$
出力: $\hat{\mathcal{H}}_g$

```
1  begin
2    S := φ
3    for each H^all_Ā ∈ {H^all_Ā_{i+1}, ..., H^all_Ā_n}
4      begin
5        D := φ
6        for each E_Ā ∈ H^all_Ā
7          D := D ∪ E_Ā \ C_g ;
8        S := S ∪ {H_A ∈ D | for all H'_A ∈ D : cost(H_A) ≤ cost(H'_A)};
9      end
10   Ĥ_g := S;
11 end.
```

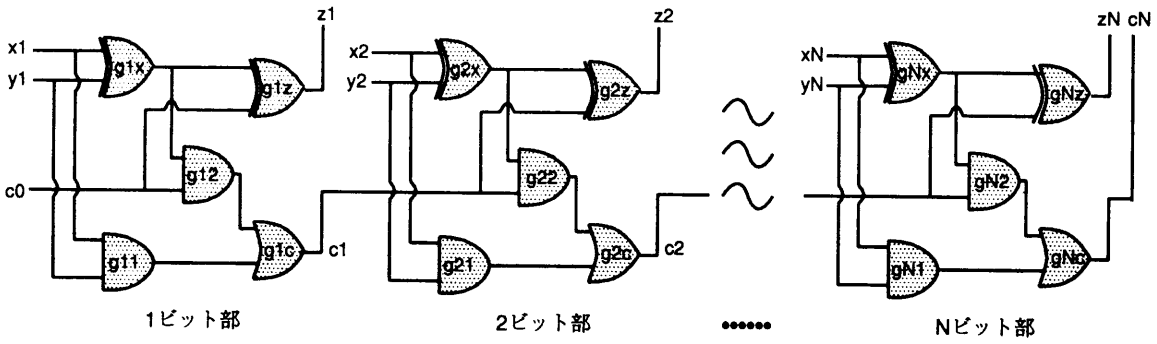
図 C.1: $\hat{\mathcal{H}}_g$を求める計算手続き

まず，$\bar{P}$に現れるアトム$\bar{A}$ついて，$\bar{P} \cup \{\leftarrow \bar{A}\}$に対する SLD 反駁を実行し，$\bar{A}$と$\bar{A}$のすべての説明から成る集合 $H^{all}_{\bar{A}}$の組 $(\bar{A}, H^{all}_{\bar{A}})$ を求める．次に，それらの組に対して図 C.1に示す計算手続きを用いて$\hat{\mathcal{H}}_g$ を求める．

以上の変更により，本推論方法は，観測の説明が通常の仮説集合で表現されるような場合にも対応する． ■

付録 D

n ビット全加算器回路の故障診断問題



n ビット全加算器回路

事実

```
val(Node2,V) ← conn(Node1,Node2),val(Node1,V).
val(out(D),V) ← dev(D,and),ok(D),val(in(1,D),A),val(in(2,D),B),and(A,B,V).
val(out(D),V) ← dev(D,or),ok(D),val(in(1,D),A),val(in(2,D),B),or(A,B,V).
val(out(D),V) ← dev(D,xor),ok(D),val(in(1,D),A),val(in(2,D),B),xor(A,B,V).
val(out(D),1) ← dev(D,Ano),stuck_on(D).
val(out(D),0) ← dev(D,Ano),stuck_off(D).
conn(in(1,x1),in(1,g1x)). conn(in(1,y1),in(2,g1x)). conn(in(1,x1),in(1,g11)). conn(in(1,y1),in(2,g11)).
conn(out(g0c),in(2,g12)). conn(out(g0c),in(2,g1z)). conn(out(g1x),in(1,g1z)). conn(out(g1x),in(1,g12)).
conn(out(g11),in(2,g1c)). conn(out(g12),in(1,g1c)).
and(0,0,0). and(0,1,0). and(1,0,0). and(1,1,1). xor(0,0,0). xor(0,1,1). xor(1,0,1). xor(1,1,0).
or(0,0,0). or(0,1,1). or(1,0,1). or(1,1,1). val(in(1,x1),0). val(in(1,y1),1). val(out(g0c),0).
dev(g1x,xor). dev(g1z,xor). dev(g11,and). dev(g12,and). dev(g1c,or). dev(g0c,or).
false ←ok(D),on(D). false ←ok(D),off(D). false ←on(D),off(D).
```

仮説: コスト

```
ok(g1x). :0.500 ok(g1z). :0.500 ok(g11). :0.500 ok(g12). :0.500 ok(g1c). :0.500 ok(g0c). :0.500
stuck_on(g1x).:0.170 stuck_on(g1z).:0.170 stuck_on(g11).:0.190 stuck_on(g12).:0.190
stuck_on(g1c).:0.220 stuck_on(g0c).:0.110 stuck_off(g1x).:0.330 stuck_off(g1z).:0.330
stuck_off(g11).:0.310 stuck_off(g12).:0.310 stuck_off(g1c).:0.280 stuck_off(g0c).:0.390
```

観測 ←val(out(g1z),1),val(out(g0c),1).

$n = 1$ の場合の例題知識ベース

関連発表

【論文誌】

1. S. Kato, H. Seki, and H. Itoh. An Efficient Abductive Reasoning System Based on Program Analysis. In *P. Cousot et al. eds. Static Analysis, Proc. of the 3rd Intl. Workshop, Lecture Notes in Computer Science*, Vol. 724, pp. 230–241, Padova, September 1993. Springer-Verlag.
2. 加藤昇平, 世木博久, 伊藤英則. プログラム解析に基づく仮説推論の高速化技法. 情報処理学会論文誌, Vol. 35, No. 10, pp. 2019–2028, October 1994.
3. 加藤昇平, 世木博久, 伊藤英則. コストに基づく仮説推論における最適解探索の一方法. 情報処理学会論文誌, Vol. 36, No. 10, pp. 2380–2390, October 1995.
4. S. Kato, H. Seki, and H. Itoh. Parallel Cost-Based Abductive Reasoning for Distributed Memory Systems. In *N. Foo and R. Goebel eds. PRICAI'96: Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence*, Vol. 1114, pp. 300–311, Cairns, August 1996. Springer-Verlag.
5. L. He, Y. Chao, S. Kato, T. Araki, H. Seki, and H. Itoh. A Multi-Agent Cooperative Reasoning System for Amalgamated knowledge Bases. In *C. Zhang and D. Lukose eds. Multi-Agent Systems, Lecture Notes in Artificial Intelligence*, Vol. 1286, pp. 92–105. Springer-Verlag, August 1997.
6. L. He, Y. Chao, S. Kato, T. Araki, H. Seki, and H. Itoh. Implementing an Automated Reasoning System for Multi-Agent Knowledge and Time . In *C. Zhang and D. Lukose eds. Multi-Agent Systems, Lecture Notes in Artificial Intelligence*, Vol. 1286, pp. 152–165. Springer-Verlag, August 1997.

【国際会議】

1. S. Kato, H. Seki, and H. Itoh. A Deductive Database Approach to Abductive Reasoning. *Proc. of the 3rd Intl. Symposium on Next Generation Database Systems and Their Applications*, pp. 77–84, Fukuoka, September 1993.
2. S. Kato, H. Seki, and H. Itoh. Cost-based Horn Abduction and its Optimal Search. In *Proc. of the 3rd Intl. Conf. on Automation, Robotics and Computer Vision*, pp. 831–835, Singapore, November 1994.
3. S. Kato, H. Seki, and H. Itoh. Cost-based Horn Abduction to Focus on the Most Probable Diagnosis. In *Proc. of the 5th Intl. Workshop on Principles of Diagnosis*, pp. 148–152, New Paltz, NY, October 1994.
4. S. Kato, C. Kamakura, H. Seki, and H. Itoh. PARCAR: A Parallel Cost-based Abductive Reasoning System. In *Proc. of the 9th Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, pp. 57–62, Fukuoka, June 1996.
5. S. Kato, H. Seki, and H. Itoh. A Parallel Implementation of Cost-Based Abductive Reasoning. In *Proc. of the second Intl. Symp. on Parallel Symbolic Computation (PASCO'97)*, pp. 111–118, Maui, Hawaii, July 1997. ACM press.

【口頭発表】

1. 加藤昇平, 世木博久, 伊藤英則. プログラム解析に基づく仮説推論の高速化技法. 人工知能学会第 7 回全国大会, pp. 75–78, July 1993.
2. 加藤昇平, 世木博久, 伊藤英則. コストに基づく仮説推論における最適解探索について. 人工知能学会第 8 回全国大会, pp = 47–50, June 1994.
3. 加藤昇平, 世木博久, 伊藤英則. コストに基づく仮説推論の並列化ついて. 人工知能学会第 9 回全国大会, pp. 17–20, July 1995.
4. S. Kato, H. Seki, and H. Itoh. PARCAR: A Parallel Cost-based Abductive Reasoning System. The Fifth Parallel Computing Workshop, pp. P1–H, March 1996.
5. 加藤昇平, 世木博久, 伊藤英則. コストに基づく仮説推論の並列化. 並列処理シンポジウム JSPP'96, pp. 169–176, June 1996.