ICOT Technical Report: TR-363

TR-363

# APRICOT－仮説推論を用いた問題解決

井上克巳

April, 1988



ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ICOT Technical Report

TR-363

# APRICOT
## ―仮説推論を用いた問題解決

APRICOT—Problem Solving with Hypothetical Reasoning

井上　克巳

Katsumi Inoue

ICOT

昭和６３年 ３月３１日

# 目次



## 概要


仮説推論は従来の古典論理の枠組みだけでは扱えなかった不完全な知識を扱うための一方式であり、真偽が不明な知識を取り敢えず真と仮定して推論を進める。後に矛盾する状況が起こったときには基になる仮説を修正することから非単調推論を実現する必要がある。従来の仮説推論の研究はより基礎的な推論メカニズムの立場から行われてきており、実際の問題解決においてどのように使われるかという観点からの研究は手がつけられていなかった。このため、本論文ではこの仮説推論を利用して問題解決を行うための枠組みについて考察する。特にＩＣＯＴにおいて次世代知識システムのための基盤ソフトウェア・ツールとして検討を進めている仮説推論システムＡＰＲＩＣＯＴの構成と、その中で利用すべき基礎技術である仮説集合の計算手続き、および知的探索のためのアルゴリズムについて説明する。

# ＡＰＲＩＣＯＴ―仮説推論を用いた問題解決

１９８８年３月

ＩＣＯＴ 第５研究室　井上 克己

## １　はじめに

すべてのＡＩシステムにおいて探索は必要不可欠である。たとえば、合成問題（設計／計画）や、いわゆる高次推論機能（非単調推論、帰納推論、類推等）は本質的に組み合わせ問題としての性格を持っており、組み合わせ爆発の回避は問題解決における重要な課題の一つである。従来から人工知能の研究では、この問題に対処するために探索を効率化するという観点を基に問題解決の枠組みが考えられており、人間の知的行為に見られるように、何らかの heuristics を基にした best-first search の必要性が論じられてきた [28]。一般に問題解決の過程はインクリメンタルに AND/OR グラフを構築しながら同時に探索しているとしてモデル化することができる。このグラフ上の探索として、従来のルールベースシステムを捉えると、強力な heuristics として生成規則が与えられることにより、一つのパスを決定している。言い替えると、従来のエキスパートシステムでは古典論理に基づいて真偽が確定した完全な知識のみを対象としている。しかしながら、知識が競合しており一意には決定できない場合の OR の処理や、例外を持つような不完全な知識の処理は、すべての可能な解を探すことや、ある枝を仮に選んでみて推論を進め、もしうまくいかない場合はバックトラックして別の解を探すことが必要となるが、現在の知識ベースシステムではそのような処理がうまく行えない。これに対する一つの技法として考えられるものは非決定的あるいは不完全な知識を仮説として設定して推論を進める仮説推論を利用することである。仮説推論では後に矛盾が生じることがあるため、その原因となった仮説を再検討し修正する。ＩＣＯＴ ではこの仮説推論に注目し、ＡＩにおける探索の効率化のための一つの有力な解決策として AND/OR グラフの探索を仮説推論の枠組みの中で定式化することを研究の一つの動機としている。

仮説推論（hypothetical reasoning）[18,22] は、問題解決の過程で競合する知識や不完全な知識（常に成立するとは限らない知識）等を取り扱える推論方式の一つであり、それらの不明であり不足しているデータや知識をとりあえず真と仮定して（仮説と見立てて）それに基づいた処理を行うことにより進めていく推論の形態をいう。この仮説は不完全な知識であるため、制約による無矛盾性のチェックや、矛盾が生じたときにその再検討や修正が必要となり、信念の翻意（belief revision）や非単調（non-monotonic）推論等の技術が必要となる。この仮説推論は、実際に人間が行っている推論であり、高次推論機能の多くを実現するための一つの鍵でもある。ところが、この仮説推論の観点からみると、Doyle の TMS [10] ではバックトラッキング及び仮説選択の制御を知的に行う方法は与えられておらず、同様に de Kleer の ATMS [4] でも組合せ爆発の問題に対しては解決策を与えるには至っていない。仮説推論の実用化のためには、これら TMSs の上に知的な制御機構を実現することが大きな課題として挙がっている。そこで仮説推論を知的な探索の手法で制御することがもう一つの目標である。

以上のことから、仮説推論に関して形式的な枠組みと知的な探索制御の方式を検討および提案するべく、仮説推論の研究に取り組んでいる。現在、ＩＣＯＴにおいて仮説推論システムとして実際にインプリメントしているものは、PSI-Ⅱ 上において de Kleer の basic ATMS [4]（簡易ヴァージョン）がある [17] が、これの論理ベースによる拡張と効率的探索の導入に向けての検討を行っている。ここではこの仮説推論に対して次の３つの観点から報告する。

(1) 最終的な成果イメージとして、仮説推論システム（名称:ＡＰＲＩＣＯＴ）のアーキテクチュアの概念検討。—— 2 章参照
(2) 論理ベースのＴＭＳにおいて導出を用いてデータの支持仮説を求める手続きに関する検討。 —— 3 章参照
(3) 仮説推論を用いた問題解決として多重コンテキストの展開に関して効率的な探索手続きと制約充足問題への適用の検討。 —— 4 章参照

## 2　仮説推論システムのアーキテクチュア

### 2.1　仮説推論のアーキテクチュア

仮説推論は次世代知識システムにおいて要求される高次推論機能の多くのものを実現するための一つの鍵ともなると思われるため、出来るだけ自然に実現できる機構が望まれている。これまで「仮説推論」という用語に対して、その背景、必要性が統一的に論じられたことも殆どなかったが、多くの高度な問題解決・推論を統合的に説明するための一つの候補として、仮説推論に対して統一的枠組を与えることは必要である。仮説推論を広義に解釈すれば、従来の仮説推論の研究は以下のごとく様々な観点から行われてきており数多くの技術が提案されている。

(1) 不確実性の扱い : MYCIN や EXPERT を始めとするエキスパートシステムにおいては ヒューリスティック・ルールを基にして中間仮説や最終結果を導くときに、仮説間に確からしさの数値を導入することにより重み付けしている。 また、Dempster-Shafer 理論や Fuzzy 論理もこれに関連する。 確信度の意味付け及び設定が困難なこととそれらの組合せ方に問題がある。

(2) Truth Maintenance Systems (TMSs) : 現在、仮説推論として注目されている方式であり、(1)とは異なり仮説間の数値による重み付けは行わずに論理的なアプローチに主眼がある。TMSs では推論の実行毎に推論結果に対する理由付け（justification）を記録することにより、推論データベース（Working Memory）の無矛盾性の維持や矛盾からの回避（belief revision）を可能としている。MIT の BL [33], AMORD [7] 及び Doyle の TMS [10], RMS [11] やそれらから派生している WATSON [15] や McDermott の TMS [24] 等は、Justification-based TMS (JTMS) として分類される。これは与えられた理由付けを基にして推論データベースの一貫性を管理するものであるが、仮説の与え方までは余り議論されておらず、効率の面では未解決な問題が多い。また JTMS での問題点に対して並列性を導入した de Kleer の TMS [4,5,6,9] や商用ツール ART の手法 [36] は仮説の組合せに着目する点で Assumption-based TMS (ATMS) として分類される。ATMS もその探索の制御に課題を残している。

(3) 論理的推論による仮説選定 : 既知の無矛盾な知識集合のもとで、観測事実を説明できる仮説を、やはり前もって与えられなければならない可能な仮説集合から発想推論の形式に基づいて（演繹を使って）選定するもので、無矛盾性のチェックを論理的に行えることと論理による定式化が可能なことから、今後の進展が期待される。 代表的なものに RESIDUE [13] や、Theorist [29] 等があるが、CMS [32] のように ATMS との関連も深い。

(4) 非単調推論 : Doyle の JTMS の定式化を試みた Non-monotonic Logic I [25] やデフォルト論理 [12,30]、Circumscription [23] 等の非単調推論の研究が盛んである。 仮説推論においては新事実の追加・削除時のデータベースに及ぼす影響が非単調であることから関連が深く、またこれらの非単調論理における信念の修正機構としての TMSs の役割が最近重要視されている。

これらの研究は相互に関連する部分もあり、特に上記 (2)-(4) の関係を明確にして、仮説推論の統一的枠組みを構築する必要がある。また、こうした理論的枠組みを問題解決システムの中で利用する場合、全体システムの中では「仮説の生成・選択・検証」という形で取り入れられるべきであるが、上記の各々はその中の一部分についての技術に留まっており、全体像が明確には与えらることは少なかった。そこで、ＩＣＯＴ で現在研究を進めている「仮説推論システム」ＡＰＲＩＣＯＴ (Assumption-based PRoblem solving Interface for COnsistent Theories ; または、Assumption-based PRoblem solver in ICOT) （4 章参照）では問題解決まで考慮にいれたアーキテクチュアを検討している。また、仮説推論にとって理論面・効率面の両観点からアーキテクチュアの検討を行うに当たって、一つの方法として「ATMS ＋探索＋論理」という枠組みを用いる。

## 2.2 仮説推論の実用化検討

仮説推論を実用化することについては決定版となる技術が発見されていないために、残された課題が多い。従って、実用的なシステムの構築についての議論をする前に、問題点を理論ベースで追及することや、実際の問題で何が仮説となりえるか、そして如何なる技術が必要かを追及することの方が現状では重要である。

現在のエキスパートシステムやエキスパート・シェルでは仮説推論のための機能が備わっているものは少なく、また単に JTMS や ATMS を組み込んだだけでは組合せ爆発の問題が起きる。基本的推論方式として仮説推論が定式化され、統一的な枠組が与えられれば、それに越したことはないが、多くの未解決な問題を全て説明できるようにするには多大の努力を必要とする。従って、全体システムの構成を考えつつ、扱える仮説の種類や推論の機能を絞って適切なアプローチをする必要がある。さらに、実用化へのアプローチとしては、仮説推論が対象とする応用問題の中で何に対して最も有効かを考え、その点に関して取り入れていく方法も必要となる。

次に考えなければならないことは、仮説推論に対する技術は汎用的なアーキテクチュアのみからは実用化は困難であるということである。特に仮説生成や検証については応用問題から考えていかなければならない部分が多いと思われる。例えば領域固有の知識や深い知識を利用することで可能な仮説集合を生成することが考えられる。

仮説推論を取り入れた例としては以下のものがある。

(1) ＡＩ技術 : 制約伝播 [2], 制約充足 [3], 診断 [8,29,31], 設計 [13], 定性推論 [14], フレーム問題 [26].
(2) エキスパート・シェル : ART [36], KEE [27].
(3) アプリケーション : 回路解析 [33], 計画 [1].

上で挙げたことに加えて、仮説推論に対するアプローチとして以下の問題を考慮することが重要である。

・何が仮説となり得るか？
・仮説間の優先順位はどのように表現し得るか？
・何をもって仮説を真あるいは偽であると判定するか？
・何が矛盾か（矛盾をどのように表現し与えるか）？

矛盾が生じたときに、それをデータベースにどのような形でストアしておくか？

・矛盾の記録を後の推論にどのように活用するか？
・仮説推論をどのレベルでサポートするべきか？

仮説には、ユーザが与える仮説とユーザは仮説と意識していないがシステムが機能、効率面で仮説として扱って処理するものの二通りが考えられる。後者の場合だと非決定性を有する選言的知識はすべて仮説と考えることができる。

・大規模・複雑な問題を扱う場合、問題解決において仮説は様々なレベルで考えなければならないが、それらを ATMS のように同一の仮説組合せで考えることは自然ではない。また、あるレベルでは仮説推論を取り入れたいが、そうでないレベルも存在するかもしれない。階層的な仮説推論のアーキテクチュアを考えることは実用的な観点からは意義が大きい。

・TMSs にはデータ間の dependency を記録するシステムが多いが、これには非常に多くの記憶領域を必要とする。そのため、必要となる部分のみ記録を取る等の方法が考えられないか？

・応用事例で TMSs を考えることについて：①問題の特徴と TMSs が必要な理由は何か？②どのような仮説があるか？③仮説の組み合わせはどれくらいあって、そのオーダーは？④TMSs を適用した場合の評価は？⑤TMSs に欠けている点は何か？

## 2.3 仮説推論システムの概念構成

以上 2.1 と 2.2 で現在の仮説推論のためのアーキテクチュアとその応用に向けての検討課題を述べた。特に仮説の生成・選択・検証の過程が統一的に扱われていないこと、及び問題解決の側面がアーキテクチャでは考慮されていないことが最も重要な課題である。そこでＩＣＯＴでの研究の基本姿勢として、仮説推論システムＡＰＲＩＣＯＴを構築するにあたり、次の２点を重視している。

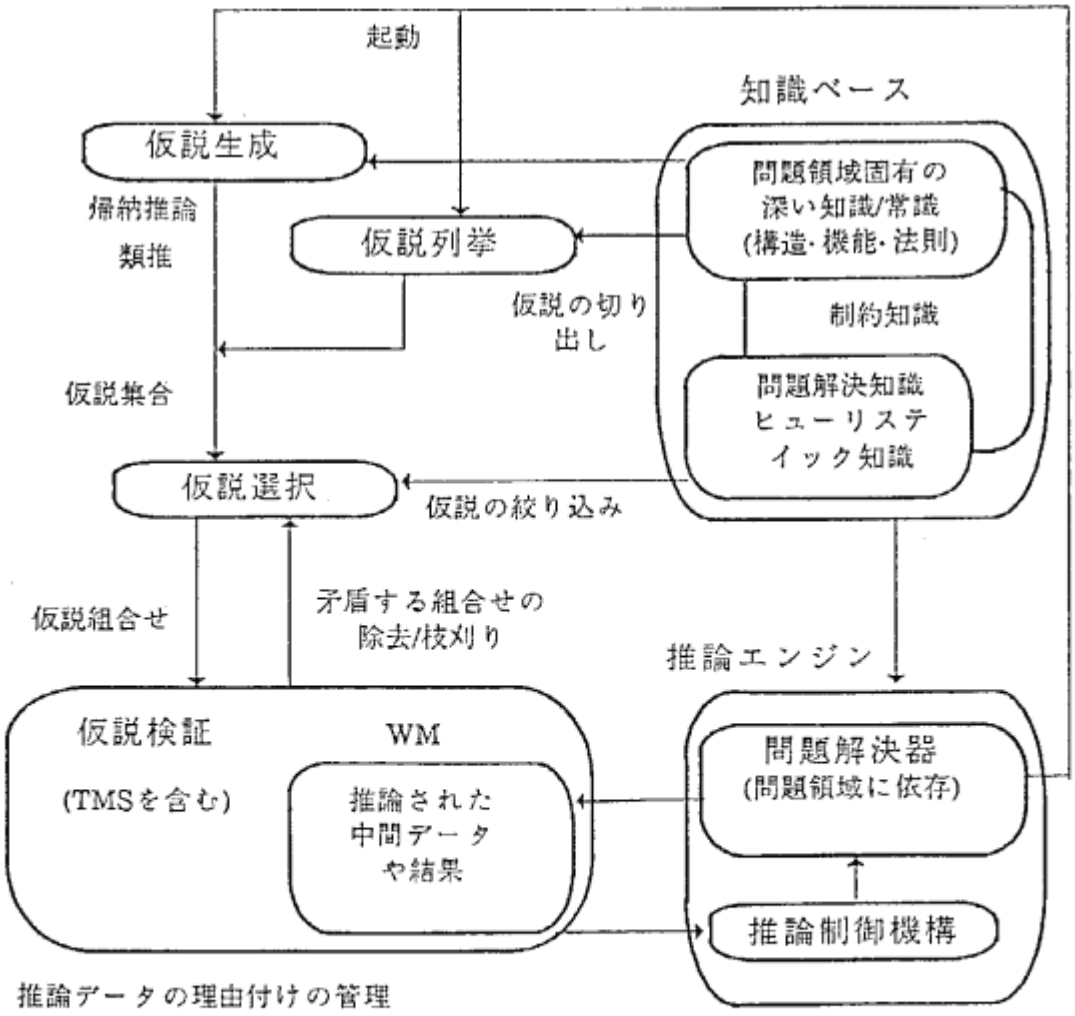
① 対象とする問題領域に固有の知識、特に深い知識（構造や機能の知識）や常識（物理法則などの知識）や制約知識を利用することにより、仮説の生成・列挙を自動的に行え、従来のヒューリスティック・ルールを重視した問題解決よりも知的であること。
② 推論制御のためのメカニズムを仮説推論機構と領域に依存した問題解決器の中間に位置付けることにより、効率面での改善を図ること。

仮説推論システム ＡＰＲＩＣＯＴ は仮説推論を問題解決に使って行く場合における基本的な枠組みを提供し、論理ベースの TMS に基づいて多重コンテキストの管理を行う。

ここで、コンテキストとは仮説を組み合わせたときにそれらの基で成立する矛盾しないデータの集合である。ＡＰＲＩＣＯＴ は問題解決器（problem solver）と知識集合に加えて、仮説推論を行うための仮説生成・仮説選択・仮説検証部を持つ。知識集合は問題領域固有の深い知識／常識と問題解決知識・ヒューリスティック知識を有し、それらのあるものは制約知識として書かれる。問題解決器は問題領域に依存して種々の推論を行い、TMS は汎用の管理機構として推論結果及び多重コンテキストを管理する。また TMS と問題解決器のインタフェースとして、汎用の問題解決戦略を幾つか有するが、その一つが dependency-directed search (DDS) (4 章参照) である。

問題解決においては、例えば設計におけるモデルの選択を考えた場合、用いる知識によって複数の競合する可能世界が考えられるため、多重コンテキストの管理が必要となる。多重コンテキストがある場合に、それらのどれを選択するかという問題は、現状では論理的に決定できる枠組みは存在しないし [16]、本来問題解決において決定されるべきものであると考えられる。従って、推論制御として多重コンテキストの管理を考え、DDS を多重コンテキストを管理するための汎用の TMS と、種々の推論を行うための問題領域に依存する問題解決器とのインタフェースを行うための汎用の枠組みとして位置付ける。

以下に ＡＰＲＩＣＯＴ のシステム構成図を示す。




推論データの理由付けの管理
複数の無矛盾なコンテキストの同時管理

・仮説生成（／列挙）部 ： 知識の集合から仮説空間を決定し、その中から問題解決において意味のある要素から構成される仮説集合を生成（／列挙）する。
起動は問題解決の過程で問題解決器からの要請による。
問題によっては、領域固有の知識や深い知識（構造と機能の知識）をうまく利用することで、可能な仮説集合を自動的に列挙する。
高次推論（帰納推論・類推）により生成することも考えられる。
仮説生成（／列挙）は一括的、あるいは段階的、階層的に行われる。
場合によってはこの部分は省略されてユーザまたは problem solver が陽に与えることもある。

・仮説選択部 ： 仮説生成部から与えられる仮説集合から、仮説となる要素を候補として選択する。 また仮説検証部で得られた結果がフィードバックされて、矛盾する、あるいは不要である仮説の選択を回避する。
また、問題解決知識やヒューリスティックスの導入により仮説の絞り込み、及び探索順序の決定を行う。
仮説検証と合わせて、仮説の展開の順序による探索の制御の問題でもある。 陽に仮説間に半順序関係を与えることがある。

・仮説検証部 ： 仮説選択部で選択された仮説に基づいて推論を展開し、その正当性を判別する。 その際、多重コンテキストを有する Truth Maintenance 機能によって推論データベース内のデータの各々に対して依存関係に基づいた管理を行い、矛盾が検出されたときに失敗に直接関係のある仮説の組合せを棄却／枝刈りすることや、新しい観測を得ることにより、仮説選択部に対して制御のための指示を与える。
ATMS のように複数の無矛盾なコンテキストを同時に管理する。
問題解決器とのインタフェースを通じて問題解決の焦点を決定する。

## 2.4 応用問題からの検討

仮説推論における汎用機能は対象とする問題領域に応じた推論戦略と合わせて適宜使って行く必要がある。その一つの応用として設計/計画問題を考慮している。生成検査法（Generate& Test ; G&T）は仮説の生成・選択と検証として仮説推論そのものの枠組みの中で捉えることができる。また、制約（constraints）の扱い [35]、再設計（redesign）、試行錯誤過程、failure recovery、等において、それらと仮説推論との相関を探る。

これらのうち、制約の問題に対しては CSP (constraint satisfaction probrem) として位置付けることができる。その探索制御にバックトラッキングを適用することにより、失敗に陥った原因を解析し、記憶しておくことによって、後の決定のガイドを行い、同様の失敗を引き起こさないように有効に使用する。ここに Truth Maintenance のアプローチが適用できる（4.3 参照）。

また一般に制約を扱う場合に困難とされているのは、動的な constraints の扱いである。 これには次の二通りが考えられる：

・弱い制約（weak constraints）の緩和（relaxation）
・途中で作り出される constraints

これに対処する一つの方法としては、その constraints が有効か無効かで仮説世界を形成することが考えられる。

こうした応用問題を実際に扱うことにより、実用的な仮説推論システムの構築が可能となり、特に各種の問題解決システムの制御用のコアとして位置付けることができる。ＩＣＯＴでは特に機械設計を例題としてＡＰＲＩＣＯＴの適用を検討中である。

## 3　仮説推論の論理的枠組み［19］

現在、仮説推論として注目されている方式の一つとして TMSs が挙げられるが、これは前述の通り次の２方式に分類できる：① JTMS [10]: 与えられた仮説から導かれるコンテキストの一貫性をデータ依存関係に基づいて管理する；② ATMS [4]: 仮説の組合せに着目する点で①の効率を改善している。また、これとは別に論理に基づく仮説推論として［13］や［29］等があり、既知の無矛盾な知識集合のもとで、観測事実を説明できる仮説を、与えられた仮説集合から演繹により選ぶ。これらの研究は相互に関連する部分もあり、CMS [32]では上記２方式の理論的基礎を与えている。ここでは、仮説推論システム　ＡＰＲＩＣＯＴ　における仮説推論の論理的枠組みについて説明する。

### 3.1　仮説推論

【仮説とは何か？】まず始めに伝統的論理学における次の２つの規則を考える。

① [hypothetical syllogism] または [modus ponens]:

　　p [assump.], p⊃q [premise]　∴ q

この規則においては q は p が真であるときに真である。ところが p の真偽が判定できなければ、q は仮説 (assumption) p の基で成立するということしか言えない。

② [fallacy of affirming the consequent]:

　　q [premise], p⊃q [premise]　∴ p

この規則は invalid であるため我々は p の真偽を判定することはできない。ところが

　　q [observation]　∵ p⊃q [premise], p [hypothesis]

のように使用すれば p を観測 q を説明するための仮説 (hypothesis) である、という言い方が可能である。この推論を発想推論 (abductive reasoning) という。

【仮説推論の２つの側面】上記の２種類の推論に対応して仮説推論も２種類考えられる。① はある仮説の基で如何なる知識が成立するかという点で仮説の利用に関係する。そのような仮説を含むデータベースの consistency を管理するものが TMSsである。② ではある観測を説明するためにデータベースと consistent な仮説を選択することが必要であり、そのための技術として、論理に基づく仮説推論がある。ところがこれら２方式はともに無矛盾性の維持という点に関しては一致しており、特にモデル論的に統一化することができる。すなわち、①では先に仮説が与えられたときに前向きに導かれるデータ集合の無矛盾性を保ち、②では先に観測が与えられたときに goal-driven で仮説を求める。詳細は［22］を参照のこと。

【仮説推論のモデル論】ここでは仮説推論の基本的な性質として次の関数 SUPPORTを考える。仮説推論はこの SUPPORT を効率良く計算しながら推論を進めて行く過程として捉えることができる。また、後述するように我々が扱う Truth Maintenance 機能にとって正確に答えを返すことが要請される問い合わせでもある。

形式：D＝SUPPORT(Σ, C).
入力：問題解決器から与えられた全ての式の集合 Σ、具象式 C。
問：C に対してその全ての最小な論理的サポートの集合 D を求めよ。
答：D は、∀d∈D に対して、次の三つの性質を満足する。

1．（因果性）　Σ ⊨ d ⊃ C．
2．（充足可能性）Σ ∪ {d} は充足可能。
3．（最小性）　d ⊃ d' なる d' は上記1．と2．を同時には満足しない。

手続き SUPPORT に対する意味論の詳細については [22]を参照のこと。ここで、D を Σ に関する C の支持仮説集合と呼び、d∈D を Σ に関する C の支持仮説と呼ぶ。また、C をアトミックに限ると、d は ATMS における環境（environment）、また D はラベル(label) に相当している。仮説推論においては、上記の Truth Maintenance 機能に対して、ある不完全知識の集合 Δ を取り扱うため、D は Δ の要素の具象例から構成されなければならない。このとき Δ を仮説集合としたとき、Δ の部分集合すなわち各要素の和集合（組み合わせ）は問題解決におけるコンテキストを決定する。

【仮説の定義】仮説（hypothesis, assumption）として考えられる知識としては例えば以下のものがある：① premise（恒真知識）及びそれらの知識だけからは正当化されなく、自分自身によってのみ正当化される知識；②矛盾を導く可能性がある知識；③不完全な（true が保証されない）知識；④選言的、あるいは非決定的な知識；⑤ 可解性 (feasibility) あるいは 最適性（optimality）が保証されない知識。これらの説明から仮説を syntactical に定義することは可能であり、それはシステムが持つ演繹規則にも依存する。重要なことは我々がいかなる問題を対象としているかによって、適切な定義を与える必要があることである。例えば常識推論や非単調推論のシステムを構築する場合、診断に適用する場合、あるいは設計に適用する場合でそれぞれ前述の仮説推論の2つの側面からも、また当然 syntax も違っていてもよい。但し、モデル論的には前述の仮説推論のモデル論で示したように統一的な semantics を与えることが可能であり、[22] に示されている。ここでは簡単に syntax を次の様に与える。仮説を任意の論理式 (well-formed formula) で取り扱うときに、一意に名前付けすることにより、一つの命題として表現するほうが計算が楽である。このために、いま γ なる論理式が真である事が保証されないとき、Γ⊃γ なる節（default logic [30] における :Mγ／γ に対応する）を Σ に追加して、Γ∈Δ を Σ に関する γ の支持仮説とする。このとき、Γ は ①明示的に仮説として導入され、② Γ 自身によって tautological に支持されている。また、Γ は真偽が判明したときには Δ から削除される。

【仮説の利用】仮説は、abductive reasoning の立場から言えば、ある観測事象 O に対する論理的整合性の取れる説明のために導入される。これは O に対して、その支持仮説集合として、全て Δ の要素である仮説から構成されるものを導出によって求めていることと論理的に等価である。このとき、支持仮説集合は ①右辺が O であるような implication の左辺を形成し、仮説以外のリテラルを含まず、② 論理的に無矛盾である。この解釈を適用すれば、ATMS ではこの支持仮説集合が「最小の」"環境" 集合（label）という形で表現されており、論理に基づく仮説推論では ¬O から goal-driven によって求まる "assumable" な命題の組合せに他ならない。また、O が仮説よりも先に陽に得られない場合もあるが、そのときもこの支持仮説集合の概念を用いることができる。

### 3.2 支持仮説集合の計算

Truth Maintenance 機能はコンテキストを陽に意識すれば、4 章の探索機能でも行っている。これはコンテキストが推論部にとって意味があるためである。これに対して仮説検証部でのデータ管理は推論部がその時点で着目するコンテキストにおける特定のデータの支持仮説集合を見付けることを主要タスクとする。但し一つのデータは複数のコンテキストに含まれてもよいため、支持仮説集合を求めることは他のコンテキストにも波及する。よって ＡＰＲＩＣＯＴ における Truth Maintenance 機能は ATMS あるいは CMS 的である。

【支持集合戦略を用いる導出による仮説集合の発見】

支持集合戦略 [37] は導出の適用を制限するためのものであり、"公理"間の導出は行わず導出形は"支持集合 (set-of-support) "に含める。いま C の支持仮説集合を求めたいとする。Σ を充足可能な節集合、Π (⊆Σ) を C を含む全ての節集合とする。以下の手続きは Π を支持集合とする支持集合戦略と飽和法を基本的に用いている。本手続きは不完全ではあるが、健全であり、段階的に Σ を構成する場合に効率が良い。
《Step-Ⅰ》Π と Σ-Π から節の導出を行う。その際、① C が導出形に残るか、あるいは ② ¬C を含む節の他のリテラルの集合が C を含む節の他のリテラルの集合を包含する、のいずれかである場合に導出を限定する。《Step-Ⅱ》へ。ここで導出がそれ以上行えないときは、Π の各節を C を imply する含意式に書き替えたときの左辺でかつ Δ の要素の具象例であるような環境をすべて合せたものが C の支持仮説集合であり、Π と Σ-Π を含めたものを Σ として手続き終了。
《Step-Ⅱ》導出形を Π に（C を含まない項のときは Σ-Π に）含め、それにより包摂される節をすべて Σ-Π と Π から消去する。《Step-Ⅰ》へ。

### 3.3 問題領域に依存した推論部

ＡＰＲＩＣＯＴ は仮説推論を問題解決に使って行く場合における基本的な枠組みを提供しているが、推論部は対象とする問題領域に依存しなければならないという考えに基づいている。このうち診断と制約充足に応じた推論戦略の概略は以下のようになる。

【診断向きの Problem Solver】仮説(hypothesis)はあるコンポーネントが正常であるとみる。観測Ｏが与えられたとき仮説の否定リテラルのみからなる和項を発見できれば、それらの和積標準形を積和標準形に変換することにより故障コンポーネントの組み合わせを発見できる。これは観測や目標を説明するための無矛盾なコンテキストをＡＰＲＩＣＯＴにより goal-oriented で形成すればよい。

【制約充足（ＣＳＰ）向きの Problem Solver】仮説(choice)はある変数への値の割り当てとみる。観測Ｏが発見された時点でその観測あるいは否定を支持する仮説を求め、現在のコンテキストとの無矛盾性を調べる。計画問題では種々の制約知識を満足するパラメータ値を無矛盾なコンテキストとして形成する。また設計問題では、設計モデルを複数管理しそれらを上位の仮説とし、下位はパラメータ値を仮説とするような階層構造でコンテキストを形成する。詳細は、4 章を参照のこと。

最後に、ここでは取上げなかったが、課題としては変数の取扱いが挙げられる。その

際、①形式的に仮説をどの範囲まで考えるのか？②自由変数を含む仮説はいつ、何に、どのように表われるか？③変数付仮説に対するアプローチの仕方は？等の点について十分に考慮する必要がある。

## 4　仮説探索アルゴリズム [20]

ここでは問題解決において複数のコンテキストに沿って推論を進める場合の一般的なアルゴリズムについて考察する。いわゆる chronological backtracking の冗長な計算と同じ失敗の繰返しを避けるために、dependency-directed backtracking (DDB) [33] では知的キャッシュを導入しており、DDB に基づく探索を dependency-directed search (DDS) といいＡＩシステムにおいて近年は一般的な推論制御手法となっている。DDS は TMSs (Doyle [10], McDermott [24], ATMS [4]等) の一機能として重要な役割を果たしている。TMSs は元々動的な知識の管理（無矛盾性の維持）が主要タスクであるが、DDS は問題解決のための指針を与える。DDS のメカニズムを論理的に解釈することにより、従来提案されていた方式の問題点を指摘しその改良を図ることが可能となる。つまり、コンテキストは仮説推論における仮説の組み合わせとして表現され、一方、AND/OR 木探索においては一つの解木 (solution tree) を構成する。矛盾を導くコンテキストは Truth Maintenance 機構を用いてチェックすることができ、これにより探索木の枝刈りを効率的に行うことができる。尚、本章に関する詳細は [21] を参照のこと。

### 4.1　ＴＭＳｓ のコンテキスト探索における問題点

多重コンテキストの管理という観点から従来の方式を考察すると、JTMS [10] は矛盾が生じたときには DDB によりそれを解消するが、バックトラッキングや仮説の選択における制御を知的に行う方法を与えていないため、多重コンテキストの問題には対処できない。従来コンテキスト機能は CONNIVER [34] のようなＡＩ言語で実現されていたが、McDermott [24] は TMS において実現する方法を与えた。さらに ATMS [4] ではそれを明確にしており、TMS を仮説の組合せに基づいて多重コンテキストに拡張し効率を上げているが、問題解決の効率化のためにはコンテキスト探索の制御は問題となる。ATMS においては environment lattice を想定した探索を実現しており、制御として最も好ましい仮説世界から順に選択していくことにより、解に早く到達することができ効率的探索が期待できる。言い替えると、ATMS の探索については node のラベル おいて最簡形の環境 が早く見つかるほどラベルの更新が少なくて済み、また nogood の最簡形が早く見つかるほど究極的に nogood となる環境に関係する問題解決のステップを削減できるために効率がよい。このため ATMS ではインタフェース (consumer architecture) を通して最も仮説の数が少ない環境から問題解決を行うようにしている [6]。探索順によりバックトラッキングは不可欠であり、de Kleer & Williams [9] では DDB による縦型探索を ATMS が扱う environment lattice の方式へ組み込むことにより、必ずしも全解が必要とされない問題に対して効率良く仮説空間を探索するための単純な手法が提案されてている。しかしながら、その手法は ATMS の利点を活かしてはいるものの、効率の面で最適とは言えない。また ATMS 自身の問題として階層的な仮説空間を構築できないことと、組み合わせ爆発に対して飛躍的な効率の増加も期待できない。以上のことから、如何なる TMSs に対しても知的な制御機構の実現が課題として挙がっている。

従来の TMSs が扱う DDS による他の限界としては、探索を一般の木（OR木）に限定していたことも挙げられる。例えば、SCHEMER [38] では探索木は binary OR 木として扱われている。我々は探索の背後にある論理に注目することにより、問題解決にとってより自然であり、OR 木を包含する AND/OR 木を探索の対象とするアプローチを取る。

### 4.2 コンテキスト探索アルゴリズム

ここでは Truth Maintenance 機能は前章の SUPPORT を実現できるものを想定する。多重コンテキストは基本的に AND/OR 木（あるいはグラフ）表現が可能である。従って、AND/OR 木探索のアルゴリズム(AO*, GBF, GBB等) [28] が適用できるはずである。このとき、一つの解木（solution tree；OR の枝のうちの１つ、AND の枝は全てを持つ部分木）が一つのコンテキストに相当する。そこで、コンテキストの展開を AND/OR 木 探索に見たてて解く。ここで OR ノードを仮説と考えたときのグラフ探索が仮説推論の戦略に相当する。ここでは、AND/OR グラフは問題解決の過程で段階的に構成されて行くものとする。即ち処理はインクリメンタルに進められるため、予め全ての仮説集合が展開されることはない。ところで、問題解決部は前述したように問題領域に依存しているので、問題解決の戦略を持つが、TMS は戦略を持たない。そのために TMS 上でのコンテキスト間での焦点は推論部が決定する必要がある。その際、ＡＰＲＩＣＯＴ の推論部が汎用的な制御戦略として取り入れることができる探索法の概略を示す。以下では、①仮説間に順序が与えられない、もしくは全順序が直列に与えられる場合、②仮説間にある種の優先順位が陽に（評価関数や半順序で）与えられる場合、に分けて考察する。

（１）【導出を用いる無評価ＡＮＤ／ＯＲ型仮説空間探索】

探索手続き（GSEARCH）の概要は次の通りである。まず解くべき問題 P が与えられる。問題は部分問題に分割されるが、その論理的関係を Σ に追加する（ADDCLAUSE)。問題解決の任意の時点で無矛盾なコンテキストは、リスト OPEN に順序付けられて格納される。１つの無矛盾なコンテキストは１つの解木に相当するが、木の葉接点のリストで表現することができる。OPEN の先頭から一つずつ解木が選択されて無矛盾性のチェック（CHECK）及び関連する問題解決手続きの実行を行い、無矛盾であれば展開（部分問題へ分割）される（EXPAND)。矛盾の生じた仮説の組合せ（ATMS の nogood に相当する）はリスト NOGOOD に入れられる。尚、前述の Truth Maintenance 機能は本探索法において特にコンテキストの矛盾の発見を効率良く行えるように工夫して用いられている。そのとき、導出が別のコンテキストの絞り込みのために用いられる。以下に探索アルゴリズム GSEARCH を示す。本アルゴリズムではリストとして、OPEN・NOGOOD・SOLUTION・D を用い、TMS の管理するデータ集合として、Σ を用いる。これらを引数とした手続きも可能だが、ここではグローバル変数としておく。リストの初期状態は、OPEN については目標となる問題Ｐのみを要素として持ち、他はすべて NIL である。D は各時点で無矛盾な部分コンテキストの最大集合のリストである。次の定義は幾つかのアルゴリズムのクラスとして拡張可能である。

定義 GSEARCH

１．終了条件（全解探索なら、OPEN = NIL；
　　一つの解だけでよいなら、SOLUTION ≠ NIL）を満たせば終了。
　　解の集合として SOLUTION が得られる。
２．OPEN の先頭の要素を取り出しＬとする。ここで、L = {C, C0, C1, ..., Ck} とする。

3．E := SUPPORT(Σ, C)。
もしEが {{}} （Σ⊨Cを意味する）ならばLは、
無矛盾であり、EXPAND(L)を実行して1．へ。
4．S := {{C}, {C, C0}, {C, C1}, ... {C, Ck},
{C, C0, C1}, ....., {C, C0, C1, ..., Ck}} 。
5．S = NIL のとき、Lは無矛盾であり、EXPAND(L)を実行して1．へ。
6．Sの先頭の要素 s を取り出し、CHECK(s, E) が TRUE なら s は無矛盾であり、
5．へ。また、CHECK(s, E) が FALSE ならLは矛盾するため、1．へ。

定義 CHECK(context, E)
1．context = {C, C0, C1, ..., Cn} とする。また、
F := CDR(context) = {C0, C1, ..., Cn} とおく。
2．もし ∃d∈D s.t. E∪F⊆d なら return(TRUE)。
3．もし ∃n∈NOGOOD s.t. E∪F⊇n なら return(FALSE)。
4．context に関する consumers を実行する。すべての実行に関して関係する節 p
に対して、ADDCLAUSE(Σ, p) を実行する。
もし p が矛盾を導けば return(FALSE)。
5．E∪FをDに追加し、E∪Fにより包含されるDの要素をすべてDから消去する。
return(TRUE)。

定義 ADDCLAUSE(Σ, s) [Truth Maintenance]
1．s をΣに追加する。
2．s 及び s によりΣの間で導出される節が矛盾 (contra) を導けば、
SUPPORT(Σ, contra) により矛盾する仮説集合 n を計算する。
n を NOGOOD に追加し、各 d∈D について、もし n が d を包含すれば、
d を d-n に置き換える。NOGOOD の中で他の要素に包含される全ての要素を
消去し、OPEN の中で NOGOOD の要素に包含される全ての要素を消去する。
3．return。

定義 EXPAND(L) [NB: depth-first version]
1．L := {C, C0, C1, ..., Ck} とし、F := {C0, C1, ..., Ck} とする。
2．Cが非端末ノードのとき、Cを部分問題に分割する。
Cの部分問題を {CS1, CS2, ..., CSm} とする。そのとき、ノードの展開に関して
ノード間の関係RをΣに追加 (ADDCLAUSE(Σ, R)) する。
ここで部分問題がＡＮＤ関係にあれば、OPEN の先頭に cons(CS1, F) を挿入して
return。さもなければ、すべての CSi (i=1, ... m) に対して cons(CSi, F) を
OPEN の先頭に並べて挿入して return。
3．Cが端末ノードのとき、Cの最も近い祖先のＡＮＤノードで兄弟ノードが残って
いる節点 AS へ戻り、cons(AS, L) を OPEN の先頭へ挿入して return。
そのようなＡＮＤが存在しなければ、Lは解であり、SOLUTION に追加して return。

上のアルゴリズムに対して幾つかのコメントが必要である。SUPPORT は GSEARCH のステップ3及び ADDCLAUSE のステップ2で計算されるが、暗黙的に幾つかの状況で必要に応じて使用される。例えば、consumers の実行に際して、関連するコンテキストをスケジュールするときに各 consumer は論理的な依存関係を知る必要がある。また、CHECK はある仮説がそれを導いたコンテキストの基で矛盾しないかどうかを検出する手続きで

ある。EXPAND における探索順序は depth-first に行うために ATMS とは異なるが、仮説の生成が段階的であり、１コンテキスト当たりのメモリ消費が大きいときは有効な方法である。

GSEARCH はインクリメンタルに階層化される木を探索するために次の利点を有する。まず、中間レベルでの矛盾に対応できるためにコンパイルされた知識に対応した問題解決が行える。次に、導出に対応する下位から上位への枝刈りが可能である。木上での枝刈りは１つの導出に相当するが、別のコンテキストの絞り込みを行う。これは主として NOGOOD とノード間の AND/OR 関係に基づいて行われる導出である。例えば、AND の関係にある子ノードは１つでも nogood であればその親ノードも nogood となり、OR の関係にある子ノードはすべて nogood のときに親ノードが nogood となる。親ノードの枝刈りは ATMS において justification として Horn clause s以外に positive clauses を扱う場合に完全性を保証するために用いる負超導出 [5] に相当する。つまり、探索においては、「解が存在するためには、仮説は～の条件を満たさないといけない」,「 n 個の内 (n-1) 個は矛盾を導くとすれば、残りの一個が解に違いない」といった意味での枝刈りが導出に相当している。

GSEARCH について以下の性質が成立する。（証明は [21] を参照のこと）

性質１ GSEARCH のステップ２で取り出されたＬからＣを除いたコンテキストＦは無矛盾である。また明らかに、$\exists d \in D.\ F \subseteq d$ 。

性質２ GSEARCH のどの時点においても、

(1) $D \supseteq$SOLUTION。但し、Ｄは nonmonotonic, SOLUTION は monotonic に増大する。

(2) $\forall d \in D.\ \neg \exists n \in$NOGOOD$.\ d \supseteq n$ 。

定理１ 最終的に求まる SOLUTION は、実際に無矛盾なコンテキストである (sound)。また全解探索のときは全ての無矛盾のコンテキストを保持する(complete)。

ＡＰＲＩＣＯＴ では２種類の仮説探索を行っている。 3.2 ではあるデータの支持仮説集合を求めるアルゴリズムを、またここでは矛盾を含むコンテキストを枝刈りするアルゴリズムを与えた。SUPPORT 以外にも、後者のアルゴリズム自身が前者を用いて実現できる。すなわち、負の和項 $\neg A \vee \neg B$ が Truth Maintenance 機能で管理されているとき、コンテキストの探索ではこれを仮説 Ａ と仮説 Ｂ が同時に存在することが矛盾する（ATMS における nogood{A, B} ）と見てそれに対応する解木の枝刈りを意味し、Truth Maintenance 機能の支持仮説集合の概念を用いると仮説 Ａ のコンテキストの基で仮説 Ｂ あるいは $\neg B$ の支持を問い合わせた場合、$A \supset \neg B$ によりそこでは仮説Ｂ が存在し得ないことを意味する。

（２）【評価関数を用いるＡＮＤ／ＯＲ木探索】

仮説間あるいはコンテキスト間に評価が与えられるときは論理の完全性は保証されないが、最良優先探索によって、組合せ的爆発を抑えることが期待できる。この場合の探索法には従来から研究されているゲーム木探索や組合せ問題に適用されている Branch and Bound（Ｂ＆Ｂ）等を (1) のアルゴリズム中に埋め込むことにより最適解を求めることができる。評価関数が使える時は、そのまま探索に反映できるため、上記 (1) のアルゴリズムは DDB + environment lattice [9] 以上の効率化が可能である。換言すれば、ATMS のような複数の並行した仮説世界上では、単なる深さ優先探索よりも、最良優先探

索を組み入れたほうがよりその特徴が発揮され、さらに効率化が期待されることが考えられる。また TMSs の整合ラベリングや最小集合被覆問題としての性質によるＮＰ完全性に対して、いずれのシステムにおいても組合せ爆発の問題に対しては解決策を与えるには至っていないが、最良優先探索はその可能性を与えている。

・最良優先探索 (best-first search)

ＡＯ＊ [28] のような最良優先探索を EXPAND あるいは CHECK で組み込んだ場合、バックトラッキングは必ずしも矛盾に出会ったときにのみ起きるわけではない。他に良さそうな候補解がある場合の context switcing に相当するバックトラックも考えられる。従って、バックトラックする前の path を取っておく必要がある。また、評価関数の値が更新されることがあるとすればそれは更なるコンテキストの切換えを誘引する。

・評価関数を用いた動的な枝刈り

一般に TMS では 制約条件 (constraints) を矛盾の検出に使用しているが、これを不要な仮説の枝刈りのための限定要因 (bound) として使用することにより、人工知能研究の基礎的な成果である B&B や $\alpha$-$\beta$ といった探索のアルゴリズムが利用できる。すなわち、制約式の動的な変化で不要な仮説の枝刈りを行う。例えば、ジョブショップ・スケジューリングにおける工程時間制限は得られた計画の満たすべき基準として矛盾の検出に使用できるが、これを問題解決の過程で得られている最短工程時間によって、その時間内で終了しないことが判った計画を枝刈りするために使用することにすれば、一層の探索の効率化を計ることが可能である。

・ヒューリスティック探索

評価関数が、解が無矛盾であるための十分条件を有するとき、その条件を満足する評価値を heuristics として用いる。 このことにより、解に到達しやすいと思われる仮説から選定が行われることになる。仮説選択順序のより良い指定が効率を左右する。

・評価関数として適切なものが得られない場合

評価関数を使う代りに'dominance' (preference) を使う。ADB : A の方が B よりも良い (と思われる)。このうち最も簡単な場合は全仮説間に全順序が与えられている場合であり、仮説選択の順序として使えるため、基本的に (1) の手続きで良い。より難しいのは部分的に半順序関係が与えられる場合である。

4.3 制約充足向きの問題解決

前節で与えたコンテキスト探索は対象とする問題領域に応じた推論戦略と合わせて適宜使って行く必要がある。一例として制約充足 (CSP) 向きの問題解決器を考える。CSP は基本的に Generate and Test (G&T) によって解くことができるが、GSEARCH アルゴリズムにおいて部分的に制約式を適用することで階層的に問題解決が行え、より効率的になる (hierarchical G&T) 。CSP では、仮説はある変数への値の割り当てとみる。但し、中間レベルでの仮説はある中間の部分問題における状態などを表わし、それ自体は値を持たないこともある。いずれのレベルにおいても、あるコンテキストの基である観測が

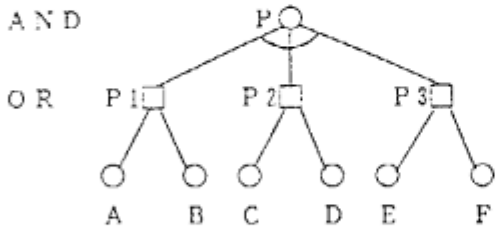
発見された時点でその観測あるいは否定を支持する仮説を求め、現在のコンテキストとの無矛盾性を調べる。以下に［9］で使われた単純な例について考察する。

例題 問題Pは部分問題P1, P2, P3の積であり、各部分問題はパラメータに対する値の決定であり、次の値の中から選択する（Exclusive OR の関係）。

P1∈{a, b}, P2∈{c, d}, P3∈{e, f}.

また、制約式として以下のものがある。

c∧e⇒false, a∧d⇒false, b⇒false.




本問題に対して総コンテキスト数は 27 個あり、ATMSでは 16 個、DDB + environment lattice では 15 個、本 GSEARCH アルゴリズムでは 13 個のコンテキストを探索する。探索過程は下図を参照のこと。ADDCLAUSE は各 GSEARCH のループで追加される論理式を、D, NOGOOD, OPEN, SOLUTION は各ループでの状態を各々表わす。但し本例では最下位レベルにしか制約が働かないために必ずしも GSEARCH の利点は発揮されていない。

| | CONTEXT | ADDCLAUSE (Σ, X) | D | NOGOOD | OPEN | SOLUTION |
|---|---|---|---|---|---|---|
| 0 | {} | P | | | {P} | |
| 1 | {P} | P≡P1∧P2∧P3 | | | {P1} | |
| 2 | {P1} | P1≡A∨B | | | {A} {B} | |
| 3 | {A} | A⊃a | {A} | | {P2, A} {B} | |
| 4 | {P2, A} | P2≡C∨D | {A} | | {C, A} {D, A} {B} | |
| 5 | {C, A} | C⊃c | {A, C} | | {P3, C, A} {D, A} {B} | |
| 6 | {P3, C, A} | P3≡E∨F | {A, C} | | {E, C, A} {F, C, A} {D, A} {B} | |
| 7 | {E, C, A} | E⊃e, ¬c∨¬e | {A, C} {E} | {C, E} | {F, C, A} {D, A} {B} | |
| 8 | {F, C, A} | F⊃f | {A, C, F} {E} | {C, E} | {D, A} {B} | {A, C, F} |
| 9 | {D, A} | D⊃d, ¬a∨¬d | {A, C, F} {E} {D} | {C, E} {A, D} | {B} | {A, C, F} |
| 10 | {B} | B⊃b, ¬b | {A, C, F} | {E} {D} {B} | φ | {A, C, F} |

## 4.4 まとめ

本章では、動的知識を管理するための汎用の TMS と、対象とする問題領域に依存する問題解決器との間のインタフェースとして、推論制御を行うための DDS の機能を持つ汎用の問題解決手法について述べた。その特徴としては、問題解決制御を AND/OR 木探索により行っていること、及び ATMS のようにすべての仮説を並列に扱うことをせず、コンテキストに沿ってインクリメンタルに仮説の追加が可能なことが挙げられる。この枠

組みは、仮説推論システム ＡＰＲＩＣＯＴ への導入により実現する予定である。課題としては、4.3 (2) のようにある種の heuristics を用いた場合に論理的な完全性と計算量の効率とのトレード・オフをどのように設定するかという点が挙げられる。

## 5 参考文献


1. Bose, P.K. and Rao Padala, A.M., Reasoning with Incomplete Knowledge in an Interactive Personal Flight Planning, Proc. Avignon 87 (1987) 1077-1092.
2. Davis, E., Constraint Propagation with Interval Labels, Artificial Intelligence 32 (1987) 281-331.
3. Dechter, R., Learning while Searching in Constraint-Satisfaction-Problems, Proc. AAAI-86 (1986) 178-183.
4. de Kleer, J., An Assumption-based TMS, Artificial Intelligence 28 (1986) 127-162.
5. de Kleer, J., Extending the ATMS, Artificial Intelligence 28 (1986) 163-196.
6. de Kleer, J., Problem Solving with the ATMS, Artificial Intelligence 28 (1986) 197-224.
7. de Kleer, J., Doyle, J., Steel, G.L. and Sussman, G.J., Explicit Control of Reasoning, MIT AI Lab., Memo No. 427 (1977).
8. de Kleer, J. and Williams, B.C., Diagnosing Multiple Faults, Artificial Intelligence 32 (1987) 97-130.
9. de Kleer, J. and Williams, B.C., Back to Backtracking: Controlling the ATMS, Proc. AAAI-86 (1986) 910-917.
10. Doyle, J., A Truth Maintenance System, Artificial Intelligence 12 (1979) 231-272.
11. Doyle, J., Some Theories of Reasoned Assumptions: An Essay in Rational Psychology, CMU Computer Science Dep., Technical Report CMU-CS-83-125 (1983).
12. Etherington, D.W., Formalizing Nonmonotonic Reasoning Systems, Artificial Intelligence 31 (1987) 41-85.
13. Finger, J.J. and Genesereth, M.R., RESIDUE: A Deductive Approach to Design Synthesis, Stanford HPP, Memo HPP-85-1 (1985).
14. Forbus, K.D., The Qualitative Process Engine: A Study in Assumption-based Truth Maintenance, Qualitative Physics Workshop Abstracts (1987).
15. Goodwin, J.W., WATSON: A Dependency Directed Inference System, Proc. AAAI Workshop on Non-monotonic Reasoning (1984) 103-114.
16. Hanks, S. and McDermott, D., Nonmonotonic Logic and Temporal Projection, Artificial Intelligence 33 (1987) 379-412.
17. 飯島，井上，ＥＳＰによるＡＴＭＳ－第１版－，ICOT Technical Memo No. TM-467 (1988).
18. 井上，仮説推論に対する一考察，ICOT Technical Memo No. TM-289 (1987).
19. 井上，導出を用いた仮説探索，情報処理学会第３５回全国大会論文集 (1987) 1567-1568.
20. 井上，仮説推論による探索木の枝刈りについて，日本ソフトウェア科学会第４回大会 (1987) 131-134.
21. Inoue, K., Pruning Search Trees in Assumption-based Reasoning.

ICOT Technical Report No. TR-333 (1988).
22. Inoue, K., On the Semantics of Hypothetical Reasoning and Truth Maintenance, ICOT Technical Report No. TR-356 (1988).
23. McCarthy, J., Circumscription—A Form of Non-monotonic Reasoning, Artificial Intelligence 13 (1980) 27-39.
24. McDermott, D.V., Contexts and Data Dependencies: A Synthesis, IEEE Trans. Pattern Anal. Machine Intelligence 5 (1983) 237-246.
25. McDermott, D.V. and Doyle, J., Non-monotonic Logic I, Artificial Intelligence 13 (1980) 41-72.
26. Morris, P., Curing Anomalous Extensions, Proc. AAAI-87 (1987) 437-442.
27. Morris, P.H. and Nado, R.A., Representating Actions with an Assumption-Based Truth Maintenance System, Proc. AAAI-86 (1986) 13-17.
28. Pearl, J., Heuristics, Addison-Wesley (1984).
29. Poole, D., Default Reasoning and Diagnosis as Theory Formation, Technical Report CS-86-08, Dept. of Computer Science, Univ. of Waterloo (1986).
30. Reiter, R., A Logic for Default Reasoning, Artificial Intelligence 13 (1980) 81-132.
31. Reiter, R., A Theory of Diagnosis from First Principle, Artificial Intelligence 32 (1987) 57-95.
32. Reiter, R. and de Kleer, J., Foundations of Assumption-based Truth Maintainance Systems: Preliminary Report, Proc. AAAI-87 (1987) 183-188.
33. Stallman, R.M. and Sussman, G.J., Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, Artificial Intelligence 9 (1977) 135-196.
34. Sussman, G.J., and McDermott, D., From PLANNER to CONNIVER—A Generic Approach, Proc. AFIPS FJCC (1972) 1171-1179.
35. Sussman, G.J., and Steele, G.L., CONSTRAINTS—A Language for Expressing Almost-Hierarchical Descriptions, Artificial Intelligence 14 (1980) 1-39.
36. Williams, C., Managing Search in a Knowledge-based System, Inference Corporation, unpublished (1985).
37. Wos, W. et al., Automated Reasoning, Prentice-Hall (1984).
38. Zabih, R., McAllester, D. and Chapman, D., Non-Deterministic Lisp with Dependency-Directed Backtracking, Proc. AAAI-87 (1987) 59-64.