人 工 知 能 70－5
（1990．5．9）

論理制約利用による高速仮説推論システム

伊藤 史朗* 石塚 満

東京大学生産技術研究所

不完全な知識を扱う次世代知識ベースシステムの基礎となる仮説推論システムの高速化について述べる．従来のシステムでは，仮説の生成，検査をPrologの単純なバックトラック上で行なっていた．我々は，Prologによるシステムの問題点を分析し，効率的な推論のためには，仮説推論に内在する論理制約を能動的に利用することが重要であると考えた．そこで，論理式から作るネットワーク上で仮説を合成するシステム(KICK-SHOTGAN)を作成した．KICK-SHOTGANは，まず推論に必要なパスを張り，次にパスを用いて矛盾の検査を行いながら仮説を合成する．この方法によって，従来のシステムに比べ1000倍以上という非常に高速な推論性能を得ることができた．

A Efficient Hypothetical Reasoning System using Logical Constraints

Fumiaki ITO Mitsuru ISHIZUKA

Institute of Industrial Science, University of Tokyo

7-22-1 Roppongi, Minato-ku, Tokyo 106, JAPAN

The handling of incomplete knowledge is a key technology for next-generation knowledge-base systems. A hypothetical reasoning system can deal with incomplete knowledge as hypothesis. Most of hypothetical reasoning systems have been built so far based on SLD resolution with depth-first search strategy utilizing the function of Prolog. These systems are not efficient because they use logical constraints for the hypotheses only in passive way.

This thesis describes a hypothetical reasoning system, KICK-SHOTGAN, which utilizes the logical constraints in active way. KICK-SHOTGAN first generates a path for proving a given goal by compiling the inference network and then synthesizes consistent hypotheses through the path. As a result, the reasoning speed of KICK-SHOTGAN is thousands times faster than that of existing systems on Prolog.

---

* 現在，キヤノン(株)

1．はじめに

次世代知識ベースシステムでは，不完全な知識を扱える枠組みが不可欠である．不完全な知識とは，例外をもつ知識や矛盾を含む知識などのことである．人間の持っている知識の多くは，このような不完全なものである．不完全な知識を扱えるようになると，人間の持っている知識をそのままシステムに与えることができるので，人間に対する親和性が増す[1].

不完全な知識を扱う次世代知識ベースシステムの基礎となるものが仮説推論システムである．仮説推論システムでは，問題領域毎に与える知識が矛盾を含んでいてもよい．問題解決の際に与えられる観測事象に応じて，矛盾のある知識を使い分けるのである．

仮説推論システムの最大の問題点は推論速度が遅いことである．推論速度を向上する手段には，ヒューリスティックな知識の利用があるが，問題領域毎にヒューリスティックな知識を得ること自体が難しいという問題点がある．従って，付加的な知識をあてにせず，本来の知識だけでも問題解決ができる枠組みが必要である．このような理由から，付加的な知識なしで仮説推論システムの高速化を図る手法を開発すること，これを本研究の目的とする．

2．論理に基づく仮説推論システム

仮説推論は，PooleらがTheorist[2]によって提示した推論法で，矛盾を含む知識を仮説として扱うことにより，不完全な知識を扱うことができるものである．　仮説推論システムに予め与えられる知識は，二つに分かれている．一つは，矛盾を含んでいない知識の集合である．これらの知識を事実と呼び，その集合を事実知識集合と呼ぶ．推論にあたって，事実は常に正しいものとして扱われる．もう一つは，矛盾を含む知識の集合である．これらの知識の一つ一つを仮説素と呼び，その集合を仮説知識集合と呼ぶ．推論にあたって，仮説素は常に正しいとは限らないものとして扱われる．

問題解決の際は，仮説推論システムに観測事象を与える．観測事象とは，ある問題領域で観測可能な事象で，与えられている知識から推論可能なものである．一般には，事実知識集合だけからでは観測事象を推論できない．また，仮説知識集合中の仮説素を全て使うと矛盾を引き起こす．そこで，仮説知識集合の部分集合として仮説を作る．この仮説のうち，事実知識集合と合わせたときに，無矛盾でかつ観測事象を推論できるものを解仮説と呼ぶことにする．仮説推論の目的は，観測事象に対する解仮説を求めることである．解仮説は観測事象の説明と考えることができる．

ここで，”観測事象が推論できる”といったときの推論として，論理の推論を用いるものが，論理に基づく仮説推論である．論理の推論は，プロダクションシステムによる推論に比べて次の点が異なる．

1）論理の推論には完全性，健全性の保証された推論法が存在している．論理を基盤とした仮説推論も完全性，健全性を保証しやすい．

2）論理式は，”～ならば～である”という知識をそのままの形で表現していて，制御情報などの付加的な知識を持たない．これは，自然な形で知識を作成できることにつながる．

これらの点は，次世代知識ベース技術の基礎として望ましいものであるので，本研究では論理に基づく仮説推論システムを対象とし，以後単に仮説推論というときは論理に基づく仮説推論のことをさすこととする．

仮説推論は，次のように定式化される．事実知識集合Fと仮説知識集合Hが与えられているとき，観測事象Oに対して，次の三つの条件

$$h \subseteq H \qquad (1)$$

$$F,\ h \models O \qquad (2)$$

$$F,\ h \text{ is satisfiable} \qquad (3)$$

を満足する解仮説hを求めることが仮説推論の目的である．

論理に基づく仮説推論システムとしては，Theoristなどの一階述語論理を対象としたものと，赤間のHYPOSE[3]など，対象をホーン節に限ったものとがある．現実的に扱う知識は，ホーン節で表現で

きる範囲で十分であるので，効率のよい推論器が存在するホーン節に対象を絞ることは有意義な制限である．本研究の対象もホーン節とする．

仮説推論の目的は，観測事象に対する解仮説を求めることであった．実際の問題解決において，解仮説はどのような意味を持つのか？仮説推論が適用される代表的な問題である診断問題と設計問題で考えてみる．

診断問題では，原因が仮説となり症状が観測となる．解仮説は，ある症状の原因を示すものである．ある症状に対して，複数の原因が考えられることがある．このときは，解仮説が複数存在するので，これらを全て求めることが妥当である．これを全解探索と呼ぶ．全解探索で，二つの解仮説h1とh2があるとき，h1がh2の部分集合であればh2はh1に対して冗長である．冗長な解仮説を求める必要はない．

設計問題では，設計事項を仮説とし，仕様を観測とする．解仮説は，要求される仕様を満足する設計結果となる．設計では，解仮説は一つ見つかればよい．仕様を満たす設計結果が得られればそれで十分だからである．これを，単解探索と呼ぶ．

いずれにも共通していえることは，知識の形が”原因→結果”，”設計→仕様”というように，問題領域の自然な知識でよいことである．例えば，仮説推論を使わずに故障診断の知識を作るとなると，”結果→原因”という形が必要で，この知識は故障診断にしか用いることができない．

仮説推論の計算の複雑さは，理論的には明らかにされていない．しかし，デフォルト論理で，ある論理式が含まれる拡張の存在を調べる問題の複雑さが，ほとんどのケースで指数オーダーであることから[4]，仮説推論の複雑さもほとんどの場合指数オーダーであると我々は考えている．実際，既存の仮説推論システムの計算時間の実測値では，仮説の増加にともなって計算時間が指数オーダーで増えている．

しかし，指数オーダーとなるのは単純に仮説推論を行なった場合であって，学習など高次の制御メカニズムを導入すると，計算量を多項式オーダーにもっていくことが可能だと考える．そこで，仮説推論システムとして必要な当面の速度向上の技術は，指数オーダーのなかにあって，定数を考えにいれた計算時間を短縮すること，及び計算の平均時間を減らすことであると我々は考えた．

本研究では，扱う論理式の範囲を命題論理のホーン節とする．現実の知識のほとんどは関数を使わない述語論理式で表現できるので，命題論理に展開可能である．さらに，方法によっては命題論理での枠組みを効率的に述語論理にも拡張可能である．そこで，基礎的な研究としては，まず対象を命題論理に絞ることは妥当である．

3．仮説推論における論理制約の利用と推論効率

我々は，仮説推論を仮説に対する制約問題としてとらえた．仮説に対する制約を満足するようなものが解仮説であると考えるのである．この制約は論理式で記述されるので，これを仮説に対する論理制約と呼ぶことにする．

それでは，仮説に対する論理制約としては何があるのか？それは，

1）解仮説は，事実知識集合と合わせると観測事象を推論できるものでなければならない．

2）解仮説は，事実知識集合と合わせても無矛盾でなければならない．

という二つの制約である．1）を観測による制約，2）を矛盾による制約と呼ぶ．仮説は任意個の仮説素の組合せであるから，仮説素の数をnとすると2^n個存在する．この中で，上記二つの制約を満足する仮説が解仮説となるのである．冗長な仮説を除く場合は，更に解仮説は減る．

解仮説を求める方法は，generate&testである．これは，仮説を生成しては，それが制約を満足しているかどうかを－解仮説であるかどうかを－，検査する方法である．仮説の生成には，インクリメンタルな仮説の生成を用いる．インクリメンタルな仮説の生成では，空集合である仮説－前提が事実知

識集合だけでできているということーから始めて，仮説素を一つずつ加えながら新しい仮説を生成していく．

generete&testの効率を上げるためには，制約を検査だけに使うのではなく，できるだけ生成段階で制約を使って，仮説の生成を絞り込むことが必要である．このような制約の使い方を制約の能動的利用という[5]．仮説に対する論理制約を能動的に利用すると，全ての仮説を生成することなしに解仮説を得ることができる．生成する仮説をできるだけ抑えることが，効率的な仮説推論につながる．

制約の能動的利用を考える前に，推論アトム列という概念を導入する．結論（g．とする）の推論アトム列とは，次の条件を満足するアトムの有限列p1，p2，...，pn（＝g）のことである．

（1）p1は前提中のファクト型の節に現れるアトムである．

（2）1＜k≦nであるpkは，

（a）前提中のファクト型の節に現れるアトムである，または

（b）前提中のルール型の節で，ボディ部のアトムが全てpj（1≦j≦k－1）のいずれかであるような節のヘッド部のアトムである．

ホーン節では，前提から結論を推論できることと，推論アトム列が存在することが同値となる．簡単にいえば，推論アトム列は，結論を推論するのにそれ自身が真であることを要求されるアトムの列である．

観測による制約を満足するためには，推論アトム列中に現れる仮説素を仮説に加える必要がある．インクリメンタルな仮説の生成では，観測事象の推論アトム列中に現れる仮説素を加えていくことにより，観測による制約を能動的に利用できる．

矛盾の検査法には次の二通りがある．

1）推論器を使って，矛盾を検出する方法．

2）矛盾を起こす仮説素の集合を登録して，それとの集合関係で矛盾を検出する方法．

2）は，ＡＴＭＳ[6]で用いられている方法である．矛盾である仮説の上位集合は矛盾であるから，これらを始めから生成しなければよい．これが実現されると，矛盾による制約を利用して仮説の生成を絞り込むことになるので，矛盾による制約を能動的に利用しているといえる．矛盾による制約を能動的に利用するには，仮説の生成を自由に行えることが必要である．矛盾による制約により，ある仮説を生成してはいけないことがわかっても，その仮説の生成を中止できなかったら意味がない．ゴール駆動の探索では，探索を進めていくときに，どのような仮説が生成されるかわからないので，ある仮説の生成を中止することができない．それに対し，データ駆動の探索では，仮説を決めてから探索を行なうので仮説の生成の中止は簡単である．従って，矛盾による制約の能動的利用のためには，データ駆動の探索による仮説生成が必要である．

4．Prologによる仮説推論システムとその問題点

Prologによる仮説推論システムは，Prologのメタインタプリタとして実現される．まず，事実知識集合の節と仮説知識集合の節をPrologプログラムの形で両者の区別がつくように保持する．推論開始時は仮説を空集合とし，前提Γを事実知識集合だけで構成する．その上で，観測事象をゴール節としてサブゴールへの展開を行なう．サブゴールへの展開は，Γに含まれている節だけを使って行なわれる．その時点でのΓだけでは展開を行えないとき，新たな展開を行える仮説素θを前提に追加して（Γ'＝Γ＋θ）新しい仮説を生成する．そして，この新しい前提Γ'が無矛盾であるかどうかを，inconsistent述語をゴールとする別の導出を行なって調べる．この結果，矛盾であるときはバックトラックにより仮説素を入れ換えて他の仮説を生成する．このようにしてインクリメンタルな仮説の生成を行いながら，空節が導出されるまで導出を進める．空節が導出されたときの仮説が解仮説の一つとなる．

Prologによるシステムには，推論速度を遅くする次のような問題点がある．

1）推論アトム列に現れない仮説素を前提に加えてしまい，そのような仮説素を含む仮説を生成してしまう．

Prologでは，導出木上に枝別れがあるとき縦型探索を行なう．導出木の葉は必ずしも空節とはなら

ないので，その葉に到る枝状で生成した仮説は無駄に終わるのである．

2）仮説の変更の際に，変更とは関係ない部分木の探索も行なってしまう．

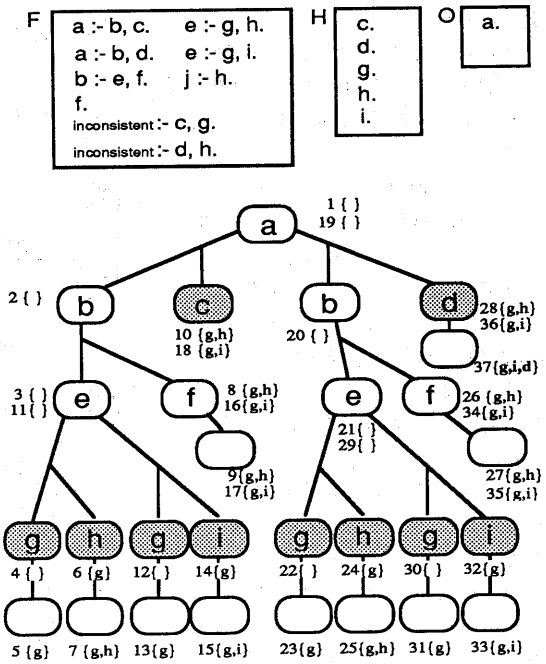
単純なバックトラックでは，仮説の変更の度に関係のない部分木の探索まで繰り返してしまう．Fig.1の例では，サブゴールｆに対する枝は，仮説の変更に関係ないにもかかわらず，仮説の変更毎に探索が行なわれている．

3）導出木上に同じ部分木が二つ以上現れる．

これはProlog自体の問題である．仮説推論では，仮説の変更を導出木上のバックトラックで行なうため，通常の推論に比べバックトラックの回数が増える．従って，この問題の影響は重大である．Fig.1の例では，サブゴールｂの下の部分木が２回探索されている．

4）既に矛盾とわかった仮説の上位集合を生成してしまう．

どのような仮説を生成するかは，導出木の探索順だけで決定される．従って，既に矛盾とわかった仮説の上位集合を生成すべきでないにもかかわらず，それを避けることができない．矛盾による制約の能動的利用が行なわれていないのである．Fig.1の例では，一度矛盾であるとわかった仮説｛c，g，h｝がもう一度生成されている．



Subtree below b is searched twice and inconsistent hypotheses {c, g, h } is generated twice.

Fig. 1 An example of hypothetical reasoning on Prolog.

5）矛盾による制約の判定自体が非効率である．

矛盾の判定は，仮説を含めた前提からinconsistentアトムが推論できるかどうかを調べることで行なわれる．これは大変効率の悪い方法である．

これらの問題は，一つはProlog自身の問題に起因する．もう一つは，推論アトム列を求めながら仮説を生成してしまっている点に起因する．そこで，我々はProlog以外の推論法をベースにして仮説推論システムを構築することを考えた．KICK-SIMは，数理計画法を使って反例を利用した仮説推論システムである[7][8]．しかし，推論アトム列を用いること自体は，反例を利用する方法よりも優れていることがわかったので，次に推論アトム列を完全に求めてから仮説を生成する方法を我々は考えた．これが，次に述べるKICK-SHOTGANである．

5．ネットワークを利用した仮説合成による仮説推論システム

KICK-SHOTGAN (Knowledge-base system handling incomplete knowledge - by synthesizing hypotheses through generated path on network) は，ネットワークで真理値を伝播しながら推論を行なう手法[9]を基盤にしている．そこで，まずこの方法を説明する．

この手法で用いるネットワークのノードは各アトムに対応するが，その他にtrueとfalseの二つのノードがある．リンクは有向で，節の型に応じて次のように張られる．

1）ルール型の節

ボディ部の各アトムに対応するノードそれぞれから，ヘッド部のアトムに対応するノードに向けてリンクを張る．これらのリンクには全て同じ番号（節番号）を付ける．

2）矛盾型の節

各アトムに対応するノードそれぞれからから，falseノードに向けてリンクを張る．これらのリンクには全て同じ番号（節番号）を付ける．

3）ファクト型の節

trueノードから節中のアトムに対応するノードに向けてリンクを張る．

各ノードは，真と偽の二つの真理値のうちのどちらかをとる．trueノードは始めに真に設定されており，他の全てのノードは偽に設定されている．その上で，同一番号の有向リンクの元のノードが全て真であるとき，それらのリンクの先のノードを真に変える．新たに真になるノードがなくなるまで，この操作を繰り返す．操作が終わったとき，falseノードが真であれば元の論理式は充足不能である．偽のときは充足可能である．

真理値伝播を生じさせるノードの探索を，推論したいアトムのノードから，ゴール駆動の探索で行なうと，そのアトムが推論されるかどうかを調べることができる．そのアトムの真理値が真となれば，推論できることを意味する．この方法による推論例をFig.2に示した．この図では，trueノードにつながるノードの真理値を真に初期化することによって，trueノードを省いている．また，falseノードは現われない．アトムaを推論するためには，ノードaの真理値を調べるだけでよいのである．

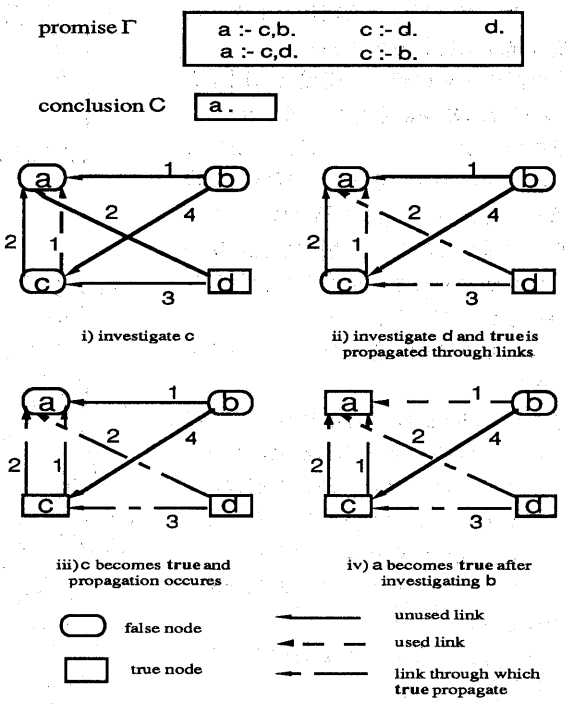


Fig. 2 An example of inference by truth value propagation on network.

KICK-SHOTGANでは，仮説推論をパス生成フェーズ(phase1)と仮説合成フェーズ(phase2)に分けた．パス生成フェーズは，推論アトム列を求めるフェーズである．仮説合成フェーズは，推論アトム列に現れる仮説素を合わせながら新しい仮説を生成していくフェーズである．

パス生成フェーズは，上記のネットワークを用いた真理値伝播による推論器を利用する．ただし，真理値を真と偽の２値ではなく，恒真，仮説の支持により真，恒偽の３値とし，各ノードの真理値を次のように初期化する．

1）事実知識集合中のファクト形式に現れるアトムに対応するノードを恒真とする．

2）仮説素に現れるアトムに対応するノードを仮説の支持により真とする．

3）その他のノードを恒偽とする．

そのうえで，観測事象をゴールとするゴール駆動の探索により，ネットワークに沿って真理値伝播を行なう．このとき，あるノードに入ってくるリンクがないか，もしくは，同一番号のリンクの元のノードの，少なくとも一つが恒偽であるようなリンクしかないときは，そのノードを恒偽のままとする．あるノードに入ってくる同一番号のリンクの元のノードが全て恒真のときは，そのノードを恒真とする．このいずれでもないときは，そのノードを仮説の支持により真とする．

観測事象を推論する仮説は，真理値伝播の際に，真理値が仮説の支持により真であったノードだけから生成される．そこで，これらのノード間を結ぶリンクだけが，次の仮説合成フェーズでは用いられるべきである．この基準に基づいて，仮説合成フェーズで用いるネットワークを切り出す．その結

果，あるノードから1本のリンクだけが出て，そのリンクが入るノードには他にリンクが入ってこないときは，この二つのノードを共通化する．Fig.3にFig.1と同じ例題でのパス生成を示した．こうして得られたネットワークでは，Prologでの問題点1）を防げる．

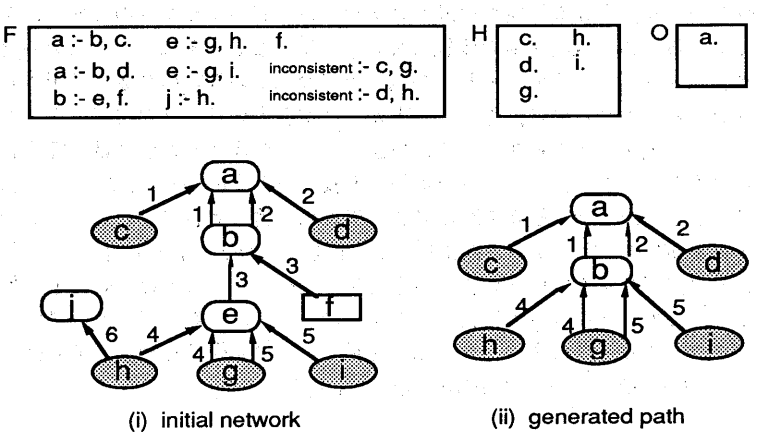


Fig.3 Examples of path generating on KICK-SHOTGAN .

仮説合成フェーズに引き渡されたネットワークの各ノードは，そのノード自身を推論するために必要な仮説（これをそのノードの支持仮説と呼ぶ．）の集合を保持する．初期状態では，どのノードも支持仮説をもたない．そのうえで，仮説素に対応するノードは，その仮説素を支持仮説とする．そして，あるノードに入ってくる同一番号のリンクの元のノードが全て支持仮説を持つならば，それらの支持仮説の和集合をそのノードの支持仮説とするのである．このように仮説を合成しながら新しい仮説を生成していく．新しく生成された仮説は，明らかにそのノードを推論するために必要な仮説である．ネットワーク上のノードは，観測事象の推論のために必要なノードであったので，新しく生成された仮説も必ず観測による制約を満足するのである．

生成された仮説は，合成時に，矛盾による制約を満足しているかどうかの検査を受ける．この検査は，後で示すようにPrologによる方法に比べ効率がよい．また，ここで矛盾であることがわかった仮説は，支持仮説集合から取り除かれるので，その上位集合は生成されない．このようしてに，Prologによるシステムの問題点4）と5）は解消される．

矛盾による制約の検査と同時に，支持仮説集合の中に冗長なものがあるかどうかを調べる．あるノードの支持仮説集合で冗長なものは，その後の合成の後も冗長であるので早めに支持仮説集合から取り除くべきだからである．

矛盾や冗長の検査を単純に行なうと，支持仮説集合内の全ての仮説の組合せを検査しなければならない．これでは効率が悪い．そこで，ステージと冗長モードの二つを用意し，これらの制約検査の高速化を図った．

各ステージSは，そのステージに対応する仮説素θsにより特徴付けられる．仮説素θsを対応するノードの支持仮説集合に加えて，このノードを発火させることからステージSは始まる．そして，仮説の合成を行うことのできる間は仮説の合成を続ける．仮説の合成がそれ以上できなくなったとき，ステージは終了する．一つのステージでは，一つの仮説素ノードの発火しか行なわない．このステージで合成される仮説は必ず仮説素θsを含む．

ところで，ステージS以前に生成された仮説は仮説素θsを含んでいないので，ステージSで生成された仮説に対して冗長になることはない．そこで，この仮説が冗長であるかどうかの検査は行なわなくてすむのである．

新しく矛盾である仮説が発見されたとき，今までに得られた支持仮説集合から，新しく矛盾であることがわかったものを除かなければならない．あるステージSで矛盾であることがわかった仮説は，そのステージを特徴付ける仮説素θsを必ず含んでいる．取り除かなければならない支持仮説は，この仮説の上位集合であるので，やはり仮説素θsを含んでいる．従って，現在のステージで生成された支持仮説だけを削除の対象とすればよいのである．

冗長モードは，冗長な仮説が生成されるケースはどのようなものであるかを検討した結果を利用したものである．ある仮説が他の仮説に対して冗長になるときは，その仮説の合成のもととなっている仮説が，ネットワーク中のいずれかの地点で二つ以上の仮説の合成に利用されたときだけである．このようなことが生じるのは，あるノードから２本以上のリンクが出ているときである．１本しかリンクが出ていないノードを経由してきた仮説だけから合成された仮説は，冗長になることも他の仮説を冗長にすることもない．このような仮説の冗長モードをINDモードとする．２本以上のリンクが出ているノードを経由した仮説から合成された仮説は，他と冗長になることがある．このような仮説の冗長モードをDEPモードとする．DEPモードの仮説とINDモードの仮説から合成された仮説は，INDモードの仮説の要素を含んでいるので，自分自身が冗長になることはあっても，他の仮説を冗長にすることはない．このような仮説の冗長モードをSUPモードとする．冗長モードは，仮説の合成の度に更新される．仮説間の冗長検査のときには，冗長モードの組合せを利用して冗長検査を削減できる．

KICK-SHOTGANのシステム構成をFig.4に示した．

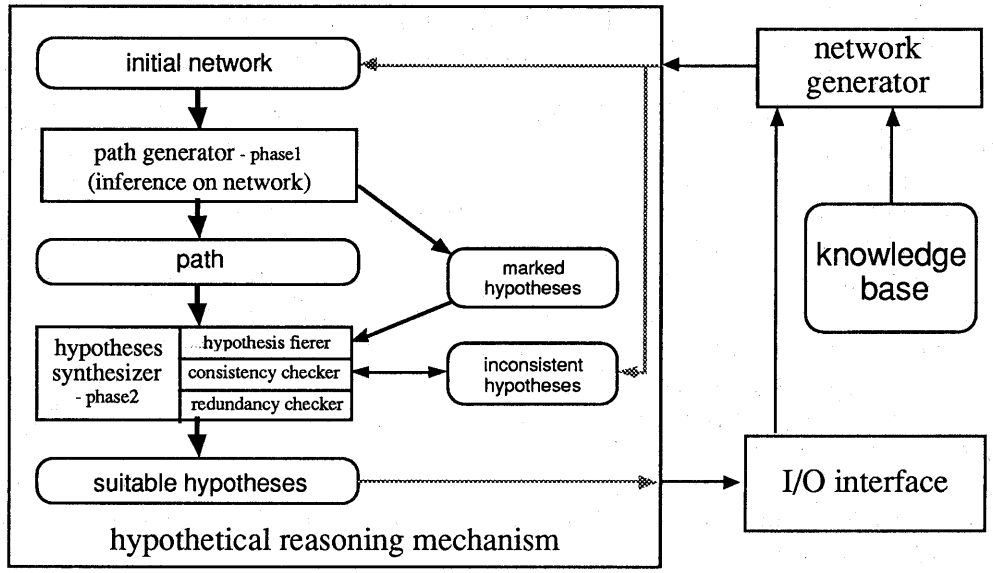


Fig.4 System configuration of KICK-SHOTGAN.

６．例題による推論速度の評価

KICK-SHOTGANの推論速度を，Prologによるシステムと比較して評価するため，例題を使っての推論速度の測定を行なった．例題としては，論理回路の故障診断を用いた．回路の規模を変えることにより，異なる規模の問題に対する計算量を測定できるようにした．推論時間は，仮説推論自体の探索にかかるcpu時間とした．事前の知識の形の変換にかかる時間は入れていない．これは，知識の形が本質ではないからである．測定はSUN4/260上で行なった．Prologによる仮説推論システムは，Sicstus Prologによってインプリメントされている．それをコンパイルして実行した．KICK-SHOTGANは，C言語でインプリメントされており，SUN4上のgccコンパイラによって最適化されている．

デコーダの故障診断での測定結果を，Fig.5に示した．KICK-SHOTGANは，prologによるシステムに比べ，いずれも1000倍以上の推論速度を出している．さらに，オーダーを示す指数関数の底も小さくなっている．

7．まとめ

仮説推論での論理制約の能動的利用を考えて作成したKICK-SHOTGANでは，従来のPrologによるシステムに比べ，高速の推論性能が得られた．KICK-SHOTGANでは，推論アトム列を事前に求め，そこに現れる仮説素を組み合わせた仮説だけを生成するという，最も観測による制約を能動的に利用できる手段が実現されている．また，仮説の生成はデータ駆動であるので，矛盾による制約の能動的利用も実現できている．さらに，ネットワークによる推論法自体が，Prolog自体の問題点を解決している．以上の理由により，高速な仮説推論が実現できるのである．

パス生成フェーズは，線形時間でホーン節命題論理の推論を行なうネットワークによる推論法を利用しているので，非常に高速に実行される．仮説合成フェーズは，ATMSのラベル更新アルゴリズムを利用したものである．ただし，KICK-SHOTGANでは，ステージと冗長モードの利用により，矛盾と冗長の検査を高速化できている．

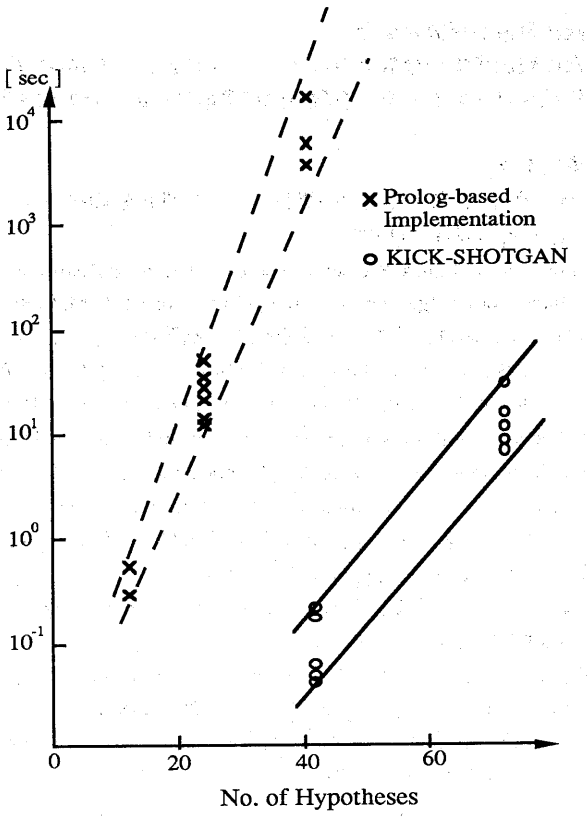


Fig.5 CPU time v.s. number of hypotheses in fault diagnosis for a decoder circuit.

仮説推論システム全体としてとらえると，KICK-SHOTGANではパス生成フェーズでゴール駆動のパス生成を行なうため，仮説合成をゴールに向かうものだけに絞ることができる．単純なATMSの利用では，理由付けを通して更新できるノード全てに対して，ラベルの更新を行なってしまう．この点でも，KICK-SHOTGANは優れている．

KICK-SHOTGANは，論理式に対する後ろ向き推論と，前向きの仮説合成を組み合わせていることにより，高速の仮説推論を実現している．この点で，従来の後ろ向きの仮説推論システムはもとより，飯島らのシステム[10]のような前向きのプロダクションシステムに対する仮説推論システムよりも優れている．

今後の課題としては，次のような点がある．

1）述語論理への拡張

KICK-SIM，KICK-SHOTGANとも現在のシステムは命題論理を対象としている．述語論理への拡張にあたっては，エルブラン領域への展開を行なえばよい．特に，KICK-SHOTGANはＳＬＤ導出と似た動きをしているのでユニフィケーションが使用可能であると考えている．

2）ステージの順序の決定

現在のKICK-SHOTGANでは，ステージの順序は知識が書かれた順序によって決定されている．知識の構造によっては，ステージの順序が推論効率に大きく影響するので，ステージの順序を推論効率がよくなるようにとることは重要である．

3）観測事象が複数の場合

複数の観測事象が複数であるときは，KICK-SHOTGANを繰り返し利用すればよい．しかし，これでは無駄な点もあるので，複数の観測事象に対応した方法を新たに考えることも重要である．

＜参考文献＞

[1] 石塚　満：不完全な知識の操作による次世代知識ベース・システムへのアプローチ，人工知能学会誌 3, pp.552-562 (1988)

[2] D.Poole, R.Aleliunas, R.Goebel: Theorist; A Logical Reasoning System for Defaults and Diagnosis, in the Volume Knowledge Representation, N.J.Cercone & G.McCalla (eds.), IEEE Press (1985)

[3] 赤間　清：論理的な制約表現を備えた拡張Prolog，人工知能学会誌 3, pp.581-589 (1988)

[4] H.Kautz, B.Selman: Hard Problems for Simple Default Logics, Proc. of 1st International Conference on Principles of Knowledge Representation and Reasoning, pp.189-197 (1989)

[5] M.Dincbas: Constraints, Logic Programming and Deductive Database, Proc. of France-Japan Artificial Intelligence and Computer Science Symposium 86, pp.1-27 (1986)

[6] J.de Kleer: An Assumption-based TMS, Artif. Intell. 28, pp.127-162 (1986)

[7] 伊藤史朗，石塚　満：数理計画法の適用による仮説推論システムの高速化，情報処理学会第38回全国大会論文集 2F-4, pp.422-423 (1989)

[8] 伊藤史朗，石塚　満：数理計画法を適用した仮説推論システム，人工知能学会研究会資料 SIG-FAI-8903-2, pp.11-20 (1989)

[9] W.F.Dowling, J.H.Gallier: Linear-time Algorithms for Testing the Satisfiability of Propositional Horn Formulae, J. of Logic Program. 3, pp.267-284 (1984)

[10] 飯島泰裕，成田良一，吉田裕之，泉　寛幸：仮説ネットワークを用いた仮説推論器，人工知能学会研究会資料 SIG-FAI-8701-2, pp.7-14 (1987)