タグ付き環境によるATMSの探索範囲絞り込み方式

古賀明彦† 増位庄一††

人工知能の問題解決では，多くの仮説の組合せから目的を達成する組合せを探索する．このような仮説管理の手法としてATMSが提唱されている．従来のATMSは，横型探索であり，矛盾する仮説の組合せの刈り取りだけで探索範囲を絞り込むため，(1) 割り当て問題などの許容解が多く存在する問題に対して，すべての許容解を生成し，組合せの爆発を起こす，(2) 横型探索以外の探索戦略を組み込むことが難しいので，解決戦略があらかじめ分かっている場合でもその戦略の適用が困難である，などの欠点を持つ．本論文では，これらの欠点を解消する新しい方式を提案する．これは (a) 仮説の組合せにタグを付け，探索の範囲をあるタグに限定することにより組合せ爆発を防ぐ，(b) タグを動的に変更することにより，ATMSの探索を横型以外にも制御できる方式である．この実現のため，(c) ラベルを伝播するものとしないものに分け，またタグから仮説の組合せにリンクを張ることにより，タグの切り替え時にラベルを効率的に再伝播するアルゴリズムを開発した．提案方式を時間割り作成問題に適用した結果，従来，組合せ爆発を起こし，解を求めることができなかった問題の解を求められるようになった．また，時間割り作成問題を一般化した問題に対して，本方式により最良探索の探索戦略の組み込みが可能となることを明らかにした．

Focussing the Search Space in the ATMS with Tagged Environment

AKIHIKO KOGA† and SHOICHI MASUI††

Problem solving in artificial intelligence means searching combinations of assumptions that satisfy the given goal. The ATMS is an assumption maintenance mechanism. The traditional ATMS offers the breadth-first search strategy and a way of restricting the search space based on contradiction. It has the following difficulties: (1) Combinatorial explosion occurs when applied to problems with many solutions, such as an assignment problem; (2) It is difficult to use a know-how with strategies other than the breadth-first. This paper proposes a solution to these: (a) we mark each combination with a tag and limit the search space to certain tagged combinations, (b) we control the ATMS by changing the tags dynamically. (c) We implemented the mechanism that satisfies (a) and (b) by developing an algorithm that maintains combinations in the two classes: active and inactive combinations, and uses the link from a tag to combinations to transfer them between the classes efficiently when tags are changed. With the method, a time-table construction problem is solved without combinatorial explosion. A best-first search strategy is put in the tagged ATMS to solve the generalization of the timetable construction problem. These indicate the difficulties (1) and (2) are solved.

1. はじめに

人工知能における問題解決では，解決手順は非決定論的に行われることが多い．すなわち，いくつもの仮説の組合せから，ゴールを満たす組合せを見つけ出し，解を構築していく．

ATMSは，de Kleer[1),2)]によって仮説の組合せを管理するための効率的なメカニズムとして提唱された．ATMSは，問題解決器が推論を行う際に得られるデータ間の依存関係（例えば，「データAとデータBからデータCが導かれる」など）を蓄えることにより，仮説の組合せとその組合せから導かれるデータを結び付けて記録する．このメカニズムにより，同じ推論を何度も繰り返さないですませることができる．また，仮説の組合せが矛盾したときは，それを含む組合せを探索範囲から刈り取ることができる．

しかしながら，ATMSではデータ間の依存関係の増加，仮説の個数の増加に従い，データとそのデータが成立する仮説の組合せの対の個数は急速に増加する．したがって，割り当て問題のように多くの仮説を扱い，

† (株) 日立製作所システム開発研究所
Systems Development Laboratory, Hitachi Ltd.
†† (株) 日立製作所ソフトウエア開発本部
Software Works, Hitachi Ltd.

かつ，組合せを刈り取れる矛盾する組合せがない問題では，実現不可能な個数の組合せを生成すること，すなわち，組合せ爆発の困難があった．

また，ATMSが仮説の組合せを生成する順番は，少ない個数の仮説の組合せを先に調べる横型探索であり，調べる順序を制御できない．このことは，例えば，「ある評価値が大きい要素から先に割り当てて行けば，ほとんどの場合解を得ることができる」などのノウハウがあらかじめ分かっている問題に対して，そのノウハウを実現する探索制御が行えないことを意味する．

ATMSの組合せ爆発を押さえる試みとしては，飛鳥井のSituated ATMS[3]，de Kleerのdependency-directed ATMS[4),5]，山之内ほか[6]がある．Situated ATMSでは，最初に状況（situation）の集合を与えて，仮説の組合せに状況を付け，伝播の処理の際に共立する状況を持っている仮説の組合せどうしだけしか組み合わせない方法で組合せ爆発を回避している．de Kleerのdependency-directed ATMSは，最初に依存性指向の探索を行い，仮説の組合せを一つ定め，それと矛盾しない組合せだけを探索する．これら二つの方式はいずれも，推論の前に探索方法を決めてしまうために，動的に得られた情報を用いて探索を制御することが難しい．山之内[6]は，仮説の組合せを一時的に矛盾すると宣言することにより，探索する組合せを制限する方法をとっている．この方式は，本来矛盾でない組合せを矛盾と同等に扱う点，また，矛盾する組合せのスーパセットになっている組合せをすべて探索範囲から取り去ってしまう点で，自由な探索制御が難しい．本論文では，矛盾集合を正しく保ち，一度推論したデータを保存するというATMSの特徴を保ったまま，探索空間の任意の部分を動的に指定できる新しい方式を提案する．すなわち，これは，

（1） 仮説の組合せにタグを付け，そのタグによって伝播する仮説の組合せを限定する

（2） タグを動的に変更することによって，ATMSの探索の範囲を自由に制御する

方式である．また，これらのタグによる探索範囲切り替えを効率的に実現するため，仮説の組合せを伝播するものとしないものに別けて管理し，タグから仮説の組合せにリンクを張ることにより，タグの切り替えを効率的に行うアルゴリズムを開発している．第2章では，ATMSの基本メカニズムを整理した後，ATMSは，ある種の問題に対して組合せ爆発を起こすことを示す．第3章では，組合せ爆発を抑えるためのタグ付きATMSの構成，動作，およびタグ制御の効率的な実現方式を述べる．第4章では，具体的な時間割問題を使って，提案方式により効率的に解決ができること，および，本方式を使って最良探索などの探索制御をATMSに組み込めることを示す．

2. ATMSとその問題点

2.1 ATMSの基本メカニズム

ATMSは図1に示すように，推論されたデータを蓄積するデータベースの役割を果たす．

問題解決器は，推論を行い，あるデータが仮説であることや，**正当化**と呼ばれるデータ間の依存関係をATMSに教える．ATMSは，それらを内部に蓄え，各データに対し，それがどのような仮説の組合せの元に成立するかを計算し，それを問題解決器の要求に従って渡す．問題解決器は渡されたデータを元に，再度，推論を行い，あらたなデータの依存関係をATMSに教える．このサイクルを繰り返しながら，問題解決器は，目的を達成するための可能な仮説の組合せをATMS内部に作成する．

以下，本論文で使用する用語の定義をしておく．データ X1, X2, ..., Xm が成立するとき常にデータ X が成立するという言明を**正当化**と言い，

$$X1, X2, \ldots, Xm \Rightarrow X$$

のように書く．データのうち，いくつかは区別され，**仮説**と呼ばれる．仮説の直感的な意味は，成立するかどうか分からないデータという意味である．仮説の集合を**環境**という．正当化の集合Jと環境Eに対して，データYが，

$$E, J \vdash Y$$

ここで，A1, ..., An ⊢ Bは，論理式 A1, ..., An から，Bが論理的に導出できることを表す；を満たせば，Yは，E，Jのもとに**成立する**と言う．このようなデータYのすべての集合を，Jの元でのEの**コンテキスト**と呼ぶ．データの中で，特に矛盾を表すデータFALSEを設ける．

$$E, J \vdash \mathrm{FALSE}$$

ならば，Eは**Jのもとで，矛盾した環境**であると言う．

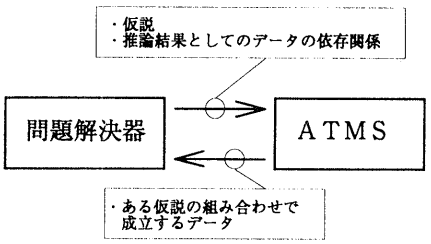


図1 ATMSと問題解決器のインタフェース
Fig. 1 Interface between ATMS and problem solver.

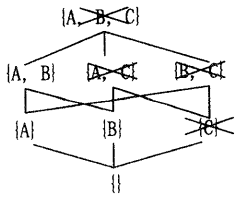


図2 ATMSが管理する束構造．環境 {C} が矛盾するならば，そのスーパセットである環境 {B, C}，{A, C}，{A, B, C} は，矛盾するので，解の探索範囲から外すことができる．
Fig. 2 Lattice of assumptions in ATMS.

Jが明らかな場合は，単に**矛盾した環境**であると言う．

環境Eは，

E, J ⊢ Y を満たし，かつ E′, J ⊢ Y を満たすようなEの真部分集合E′が存在しない

とき，**極小である**という．データYをコンテキストとして含む矛盾しない極小な環境の集合を，Yの**ラベル**という．矛盾した極小な環境の集合，すなわち，FALSEのラベルを**ノーグッド**と言う．ラベルやノーグッドは，正当化の左辺のデータのラベルから正当化の右辺のラベルを計算する処理を繰り返すことによって求める．この処理を**ラベルを伝播する**と言う．

ATMSでは，環境は，集合の包含関係を使った束構造として管理される．**図2**は，仮説A，B，Cを使ったすべての環境の集合の包含関係を表している．ある環境で成立するデータは，そのスーパセットの環境でも成立する．したがって，あるデータが特定の環境で成立するかどうかは，そのデータが成立する極小の環境の集合を記録しておけば，集合の包含関係のチェックだけで判定することができる．また，ある環境がノーグッドならば，それを包含する環境はすべて矛盾し，解の候補から除外することができる．このようにATMSは，ノーグッドを含む環境を束構造の中の探索範囲から刈り取ることによって探索の効率化を達成している．ATMSのデータは，**ノード**と呼ぶ次のような3項組として管理される．

⟨data, label, justifications⟩

ここで，dataはデータ，labelは，dataのラベル，justificationsは，dataを右辺に持つ正当化の集合である．また，ノーグッドは，

⟨FALSE, nogood, justifications⟩

のように管理される．

2.2 組合せ爆発と探索制御の問題

ATMSのメカニズムでは，多くの仮説が与えられ，かつ，束構造の中の環境を刈り取るための有効なノーグッドがない問題に対しては，莫大な量の環境を計算，蓄積する．このような問題の例としては，1週間の各講義に勤務可能な先生の中から担任の先生を割り当てていく時間割作成問題がある．時間割作成問題では，しばしば制約が緩やかであり，非常に多くの部分解や解が存在する．ATMSでは，生成された部分解や解をすべて記録するため，実際上解を求めることができない．したがって，解を得るためには部分解の組合せ方を何らかのノウハウを使って制限する必要がある．この種のノウハウは，解の存在範囲を厳密に限定するものではないので，その範囲に解が無い場合は，動的に得られた情報を使って動的に探索範囲を限定する必要がある．また，ユーザが持つノウハウを表現しやすくするためには，絞り込みのためのATMSの操作が直感的なモデルを持ち，そのモデルがユーザのノウハウと密接に結び付く必要がある．これらをまとめると，ATMSをこのような問題に適用するためには次の要求を満たす必要がある．

(1) 探索範囲の絞り込み手段の提供
(2) 探索範囲の動的な制御可能性
(3) ユーザにとって理解容易なモデルの存在

本論文では，これらを満足するために，ATMSに探索範囲を絞り込むタグ付きATMSを提案する．

3. タグ付きATMSによる探索範囲の切り替え

3.1 タグ付きATMSの考え方

2.2節で述べた要求をみたし，効率的な探索制御を行うために，推論の目的を表すような印（タグ）をユーザが環境とATMS管理ルーティンに付け，ATMSが扱う環境を同じ目的のタグを持つ環境だけに限ることによって絞り込みを達成する．また，タグ変更のインタフェースをユーザに開放することにより，扱う環境を動的に変更できるようにする．

タグ付きATMSでの**環境**とは，2項組

⟨T, E⟩

である．ここで，Tは記号の集合であり，これを環境の**タグ**と呼ぶことにする．Eは仮説の集合である．Tの要素を**トピックス**と呼ぶ．また，タグ付きATMSは，ATMSの状態として**現在のタグ**と呼ばれるトピックスの集合を一つ持つことにする．現在のタグをCTで記す．タグの直感的なモデルは次の通りである．

[タグのモデル]

ユーザは，問題解決過程の各時点において何らかの関心事を持っている．例えば，現在，ある部分解を求めているならば，その部分解にユーザが付けた名前「部分解1」が関心事であり，また，ある特徴を持つ解を求めているならば，その特徴の名前が一つの関心事であ

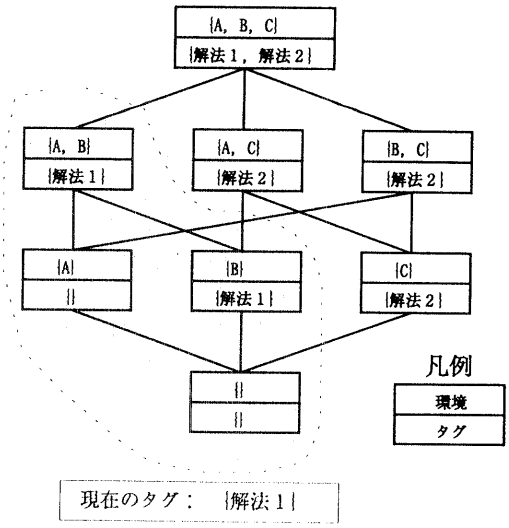


図3 タグ付きATMSの概念図
Fig. 3 ATMS with tags.

る．タグ付きATMSのトピックスはその関心事を表現する．現在のタグは，ユーザが問題解決の過程のある時点で関心を持っているトピックスの集合を表す．環境のタグは，その環境が関係するトピックスの集合を表す．ユーザは問題解決の際に，そのとき関心を持っている環境だけを計算する．ユーザが扱う環境が切り替わるのはユーザの関心がさらにほかのトピックスを含むように広がった場合や，まったく別のトピックスの集合に移った場合である．

このモデル化の元にタグ付きATMSでは，タグが現在のタグに含まれる環境だけを伝播の対象にする．例えば，**図3**では現在のタグは，{解法1}であるので，それに含まれるタグを持つ環境{}，{A}，{B}，{A, B}だけが伝播の対象となる環境である．

タグ付きATMSの問題解決器とのインタフェースは，図1に，問題解決器からATMSに対するタグの操作を加えたものである．タグ付きATMSで，ある環境を伝播の対象から外すためには，現在のタグCTに含まれないトピックをその環境に加えるか，あるいは，現在のタグからその環境に含まれるトピックを取り去ればよい．タグの操作としては，次のようなものを提供している．

(1) add_topic(Topic, ⟨T, E⟩)
環境⟨T, E⟩のタグTにトピックスTopicを追加する．このとき，Eのスーパセットとなる環境にもトピックスTopicを同時に追加する．

(2) delete_topic(Topic, ⟨T, E⟩)
環境⟨T, E⟩のタグTからトピックスTopicを削除する．このとき，Eのスーパセットとなる環境からもトピックスTopicを同時に削除する．

(3) add_current_topic(Topic)
現在のタグCTにトピックスTopicを追加する．したがって，Topicがあるために伝播の対象から外されていた環境は伝播の対象になる．

(4) delete_current_topic(Topic)
現在のタグCTからトピックスTopicを削除する．したがって，タグにTopicを持つ環境は伝播の対象から外される．

3.2 タグ付きATMSの構成と動作

まず，ATMSのノードを

⟨data, label1, label2, justifications⟩

のように4項組に拡張する．label1は，現在のタグに含まれるタグを持ち，伝播の対象となる環境の集合である．label2は，現在のタグに含まれないトピックスを含むために伝播の対象から外された環境の集合である．label1を**表ラベル**，label2を**裏ラベル**と言うことにする．正当化追加のときの処理は，ATMSの追加処理におけるラベル伝播の処理を表ラベルだけに限定したものである．正当化jが加えられたときのラベル伝播処理ADD_TAG_JUSTIFICATIONの厳密な動作を以下に記述する．

ADD_TAG_JUSTIFICATION(j)
jは，正当化 X1, ..., Xn ⇒ X とする
(1) jをXのノードに追加する．
(2) Xがノーグッドでないなら
UPDATE_TAG_LABEL(X)
を呼び，正当化追加によって新たに生成されるラベルを伝播する．そうでなければ，
UPDATE_TAG_NOGOOD()
を呼び，新たに生成されたノーグッドで環境の刈り取りを行う．

UPDATE_TAG_LABEL(X)
Xは，データとする．
(1) Xの正当化の集合をJとする．
(2) NEW_LABEL :=
COMPUTE_TAG_LABEL(J)
(3) NEW_LABELがXのいまの表ラベルlabel1と等しいならば，このUPDATE_TAG_LA-

BEL(X) の処理を終わる.

(4) X のラベルを NEW_LABEL とする.

(5) X を条件部分に含む正当化の帰結部分に現れるデータの集合を {Y1, Y2, ..., Yn} とし, 各 Yi に対し, UPDATE_TAG_LABEL(Yi) を呼ぶ.

COMPUTE_TAG_LABEL(J)

J は, 正当化の集合

(1) L を空集合にする.

(2) J の各正当化 X1, ..., Xn ⇒ X すべてに対して,

(2.1) X1, ..., Xn の表ラベルを L1, ..., Ln とする.

(2.2) $L' := \underset{i=1,\ldots,n}{\{\cup Ei\}} \quad \underset{Ei \in Li}{}$

ここで, タグ付き環境 E1 = ⟨T1, Env1⟩ と E2 = ⟨T2, Env2⟩ の和は,

E1 ∪ E2 = ⟨ T1 ∪ T2, Env1 ∪ Env2⟩

とする.

(2.3) L := L ∪ L'

(3) L の中から無矛盾で, 極小な環境だけを集めたものを L'' とし, L'' のなかから, 現在のタグに含まれないタグを持つものを裏ラベルに追加し, L'' から取り除いて返す.

UPDATE_TAG_NOGOOD()

(1) FALSE のノードを調べて, その正当化を使って COMPUTE_TAG_LABEL() と同様にノーグッドを伝播させ, 新しいノーグッド集合 NEW_NOGOODS を計算する.

(2) NEW_NOGOODS が古いノーグッド集合と同じであれば, 処理を終わる. そうでなければ, すべてのラベルからノーグッドのスーパセットになっている環境を取り除く.

3.3 探索範囲切り替えの効率的実現方式

タグ付き ATMS のノードは,

⟨data, label1, label2, justifications⟩

であり, タグ切り替えが起こらない限り, label1 だけに限定した通常の ATMS の伝播処理が利用できる. タグの切り替えが起こり, 伝播対象だった環境が伝播対象から外れた場合は, それらを label1 から label2 へ移せばよい. また, 伝播対象でなかった環境が伝播対象になった場合は, それらを label2 から label1 へ移し, 再伝播する. 一度作成した環境は伝播対象から外れても裏ラベルにとっておかれるので, 再度作成するオーバヘッドはない.

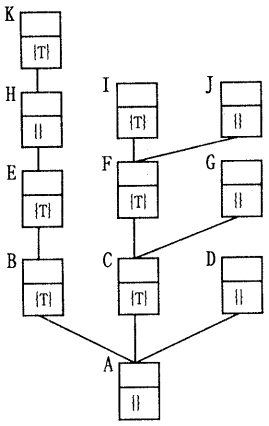


図4 タグと環境のリンク構造
Fig. 4 Link from tags to environments.

タグ切り替えを実現する上で, つぎのような非効率性が考えられる.

(1) 現在のタグ CT にトピックスを追加, 削除した場合, 全環境を調べ, 新たに表ラベルへ移す環境と裏ラベルへ移す環境を検索しなければならない.

(2) 一つの環境にトピックスを追加, 削除した場合, その環境のスーパセットである環境すべてに同様にトピックスを追加, 削除する必要がある.

(1) に対しては, トピックスから, それをタグに持つ環境へのポインタを張ることにより, トピックスの追加削除が影響する環境を効率的に調べるようにした. このとき, すべての環境にポインタをつけるのではなく, 環境が束をなすことを利用して, メモリ効率を向上した. すなわち, 環境束内の部分ネットワークを

T.added: トピックス T を含む環境で, その直接の親 (その環境に含まれる最大の環境) は, トピックス T を含まないものの集合.

T.deleted: 直接の親の環境はトピックス T を含むが, 自分自身は T を含まない環境の集合.

の二つの集合で上と下から挟むことによってトピックスを持つ環境の集合を表す. 例えば, 図 4 では, トピックス T を持つ環境の集合は

T.added = {B, K, C}, T.deleted = {H, G, J}

で表現する.

環境のタグの更新関数 add_topic(),delete_topic() の呼び出しの時, これらの集合を次のように更新する.

(a) 環境 E にトピックス T を加えたとき,

$$\text{T.added} := \text{T.added} - \{E' \in \text{T.added} \mid \text{env(E)} \subseteq \text{env(E')}\}$$

$$\text{T.deleted} := \text{T.deleted} - \{E' \in \text{T.deleted} \mid \text{env(E)} \subseteq \text{env(E')}\}$$

if (E のどの親も T を含まない) then
T.added := T.added + {E}

(b) 環境 E からトピックス T を取り去ったとき,
T.added := T.added − {E′ ∈ T.added | env(E) ⊆ env(E′)} − {E}
T.deleted := {E′ ∈ T.deleted | env(E) ⊆ env(E′)}
if (E の親環境のどれかが T を含んでいる) then
T.deleted := T.deleted + {E}

(2) に対しては，タグをビット列表現することにより，メモリ効率向上だけは対処した．トピックスの追加削除処理は指定された環境のスーパセットである環境すべてに対して行っている．

4. タグ付き ATMS の評価

4.1 トレーナ時間割作成問題による性能評価

この章では小さな規模の割当問題を使ってタグ付き ATMS の効果を評価する．まず，ATMS を用いてこの問題を解く知識（ルール）を示し，生成される環境の個数を評価し，次に同じ知識を使って，タグ付き ATMS ではより少ない個数の環境しか生成されないことを示す．ここで扱う問題は次のものである．

スポーツクラブ A では，1 週間毎日，トレーニングのクラスを開いている．毎日のクラスでは，2 名のトレーナを割り当てている．次のような条件を満たす時間割を作成せよ．

トレーナは，
a さん，ランク = 3，休日 = 日曜 金曜
b さん，ランク = 2，休日 = 月曜 金曜
c さん，ランク = 1，休日 = 火曜 木曜
d さん，ランク = 2，休日 = 水曜 土曜
がいる．

(1) 毎日，主担当と副担当を選ぶ．主担当のランクは副担当のランクより大きくなければならない．
(2) トレーナを休日に割り当てることはできない．
(3) トレーナを，2 日続けて主担当に割り当てることはできない．

ATMS を用いて解く場合，次のようになる．

曜日，日月火…を 0, 1, ..., 6 で表すことにする．

仮説の集合は，各曜日 $?X = 0, 1, \ldots, 6$ と各トレーナ $?Y = a, b, c, d$ に対して，

[?X] 曜日の主担当は，[?Y] である
[?X] 曜日の副担当は，[?Y] である

の 56 個を設ける．ルールは，次のようになる．

{ ペア割り当てルール }
if [?X] 曜日の主担当は，[?Y1] である
[?X] 曜日の副担当は，[?Y2] である
RANK(?Y1) > RANK(?Y2)
then [?X] 曜日の割り当てを行った

{ 休日に割り当てることを禁止するルール }
if ([?X] 曜日の主担当は，[?Y] である
or
[?X] 曜日の副担当は，[?Y] である)
[?Y] の休日は，[?X] 曜日である
then FALSE

{ 連続して主担当に割り当てることを禁止するルール }
if [?X] 曜日の主担当は，[?Y1] である
?X2 = (?X1 + 1) mod7
[?X2] 曜日の主担当は，[?Y1] である
then FALSE

{ すべての曜日に割り当てるルール }
if [0] 曜日の割り当てを行った
:
:
[6] 曜日の割り当てを行った
then すべての割り当てを行った

ATMS では，このようなルールをデータドリブンで発火させてできた正当化を入力してやれば，データ「すべての割り当てを行った」のラベルとして，この問題の解となる仮説の組合せが記録される．

次に生成される環境の個数を評価する．このとき，ノーグッドとして生成される環境は比較的少数で以下に述べるどの解法でもほぼ一定なので無視する．

各曜日に割り当て可能な組合せは，日曜日 2 通り，月曜日 3 通り，火曜日 2 通り，水曜日 3 通り，木曜日 2 通り，金曜日 1 通り，土曜日 3 通りであり，ルールで，すべての曜日に可能な割り当ての組合せが一度計算されるので，$2 \times 3 \times 2 \times 3 \times 2 \times 1 \times 3 = 216$ 個の環境が生成され，上で計算されたノーグッドにより，消去され，最終的には，8 個の環境データが「すべての割り当てを行った」のラベルとして残る．

以下，次のようなノウハウを使った解法をタグ付き ATMS で表現し，生成される環境の個数を評価する．

[ノウハウ]

主担当はランクの大きな人を先に割り当てる．
副担当はランクの小さな人を先に割り当てる．

ルールは，ATMS の解法と全く同じである．問題解決器は，特定の仮説を推論対象からはずすためのタグ OUT を設け，まず，すべての仮説のタグを {OUT}

に設定して，一旦伝播の対象から外し，ノウハウに従って調べる仮説の組合せのタグを空集合に設定することにより一つ一つ調べて行けばよい．この際，ノーグッドになる仮説は調べる必要がないので，バックトラックする．この手続き SOLVE() を以下に詳細に記述する．

SOLVE()

(1) 配列 mains に，トレーナをランクの大きな順に，配列 subs に，小さな順にソートしていれる．

(2) 現在のタグ CT を空集合とし，すべての仮説のタグを OUT だけからなる集合 {OUT} とする．

(3) SOLVE2(0, {}) を呼ぶ．

SOLVE2(n, env)

(1) n が 7 ならばすべての割当が終わっているから，データ「すべての割り当てを行った」のラベルを返して停止する．

(2) i を 1 から 4 まで増やしながら次の処理を行う．

(2.1)「[n] 曜日の主担当は，[mains[i]] である」のタグを空集合にして伝播の対象にする．

(2.2) 環境 env に「[n] 曜日の主担当は，[mains[i]] である」を加えた環境を env2 とする．

(2.3) ルールを発火させて，できた正当化をタグ付き ATMS に渡す．

(2.4) env2 がノーグッドでないなら，j を 1 から 4 まで，増やしながら次の処理を行う．

(2.4.1) 仮説「[n] 曜日の副担当は，[subs[j]] である」のタグを空集合にして，伝播の対象にする

(2.4.2) env2 に「[n] 曜日の副担当は，[subs[j]] である」を加えた環境を env3 とする．

(2.4.3) ルールを発火させて，できた正当化をタグ付き ATMS に渡す．

(2.4.4) env3 がノーグッドでないならば，

SOLVE2($n+1$, env3);

を再帰的に呼び出す．

この方式では環境が生成されるのは，(2.2) と (2.4.2) の箇所である．実際に動かしてみて生成される環境の個数をカウントすると，最初の解を求めるまでに，50 個の環境を生成する．

以上述べた方式で生成される環境の個数を評価すると**表 1** のようになり，生成される環境の個数を本方式のメカニズムで絞り込むことができることが分かる．

次に実際のマシン上で上記の問題でトレーナの数を増やして実験した結果を**表 2** に示す．ここで，使用したマシンは，CPU が 68040 25MHz のものである．ATMS を用いた解法では全解探索し，タグによるバックトラック制御では，最初の解だけを見つけた．トレーナ数 6 の場合は，ATMS では解くことができなかった．

表 1 解法別の生成される環境個数比較
Table 1 Comparison of number of environments.

方式	環境の個数
ATMS	216
タグ付き ATMS	50

表 2 タグ付き ATMS の実測性能
Table 2 Performance of ATMS with tags.

トレーナ数	ATMS	タグによるバックトラック制御
4	2.5 sec	4.5 sec
5	29 sec	5.5 sec
6	—	7.5 sec

4.2 割当て問題用汎用制御のタグによる表現

タグ付き ATMS の機構を用いて，次の制約充足問題に対してドメインシェルを構築したので，その方法を示す．

制約充足問題

入力

(1) 変数の集合 $\{Xi|i=1,2,\ldots,n\}$

(2) 領域の集合 $\{Di|i=1,2,\ldots,n\}$
各領域 Di は，変数 Xi が取り得る値の有限集合

(3) 変数間の制約 $\{Cj|j=1,2,\ldots,m\}$

出力

Xi に領域 Di から値を割り当て，制約を満足するもの

ドメインシェルは，**図 5** に示すように，ユーザから，問題の定式化と問題解決のノウハウを与えられて，それをタグ付き ATMS の制御に翻訳するインタフェースを設けることにより構築する．このインタフェースは，ユーザのノウハウを解釈実行し，最良探索を行うようにタグ制御を行う．

問題の定式化は，次のようにして ATMS に渡す．

［変数およびその領域］

各変数 ?X およびその領域の各値 ?V に対して，仮説

変数 [?X] は，[?V] である

をタグ付き ATMS に渡すことにより設定する．

［制約］

(1) $\forall x.P(x)$ の形の制約

これは，

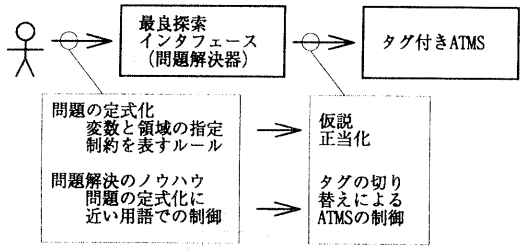


図5 制約充足問題ドメインシェルの構造
Fig. 5 Structure of domain shell.

if $\neg P(x)$ then FALSE

の形のルールで表現する.

(2) $\exists x.P(x)$

これは,最良探索インタフェースにあらかじめあるデータ「ゴール」を指定しておき,ユーザは,

if $P(x)$ then ゴール

の形のルールで表現し,最良探索インタフェースがすべての割り当てを終わった時点でゴールが生成されているかどうか調べることによって実現する.

[問題解決のノウハウ]

次のようなメソッドを用意し,

(1) 変数 [?X] の優先度を [?Priority] にする

与えられた変数に優先度を対応付けてドメインシェルの内部に記録し,探索の順序を決定する.

(2) 値 [?V] の優先度を [?Priority] にする

与えられた値に優先度を対応付けてドメインシェルの内部に記録し,この値によりタグを制御し,どの値から割り付けて行くかを決定する.

ルール

if 条件 then メソッドの呼び出し

の形でユーザに記述して貰う.

最良探索インタフェースの制御アルゴリズム

BACKTRACK() は,次に示すように変数の優先度や値の優先度に従って,各仮説データ

「変数 [?X] は,[?V] である」

のタグを {OUT} から {} に変更していき,伝播の対象とすることによって行う.

BACKTRACK()

(1) 割り当てる変数を優先度に従って選択する

(2) 割り当てる変数が残っていない場合には,ゴールのラベルがあるかどうか調べて,もしあれば,それを解として返す.ラベルがなければ,現在の割り当ては,解では無いので,BACKTRACK() を呼び出した側に戻る.

(3) その変数の領域の各値 ?V に対して,次の動作を繰り返す.

(3.1) ?V を変数に割り当てた仮説をタグ操作により,推論範囲に入るようにする.

(3.2) もし,現在までの割り当ての組合せが,矛盾として登録されているなら,(3) に戻る.

(3.3) 制約を表すルール群を起動する.この時,解決のためのノウハウも実行し,変数の優先度の変更,値に対する優先度の計算も行う.推論の結果,もし,現在の仮説の組合せがノーグッドであれば (3) に戻る.

(3.4) このアルゴリズムを再帰的に呼び出して,残りの変数の割り当てを行う.

(3.5) 処理 (3.3) で設定された優先度を取り消す.

(4) このアルゴリズムを呼び出した側に戻る.

4.3 評価のまとめ

時間割問題では,タグによる探索範囲の制御を用いることにより探索の制御を組み込み,生成される環境の個数を絞り込むことができることを示した.また,実測のデータは,ATMS だけを使った場合,組合せ爆発で解を求めることができなかった問題に対しても,タグによる制御を組み込めば解を求めることができることを示している.また,実測データは,探索時間がトレーナ数に比例して増えており,これは,タグ制御のオーバヘッドが仮説の個数によらずほぼ一定で実現できていることを示す.

4.2 節で述べたバックトラックの組み込みでは,タグの切り替えを使ったバックトラック用の制御方式を記述した.タグによる切り替え方式は,探索空間の自由な領域を指定できるので,本論文で述べた変数の割り当てを行う制約充足問題に対しては,比較的小さな制御プログラムを記述するだけでドメインシェルが構築できた.

5. おわりに

de Kleer の提唱した仮説管理システム ATMS で,組合せ爆発を起こしてしまう問題があることに対処するために,ATMS の環境伝播範囲を制限するメカニズムとして,環境にタグを付けた ATMS を提案した.トレーナの割り当て問題を例として,生成される環境の個数を減少させることができることを示した.また,タグの切り替えを動的に行うことによって,制約充足問題に対して優先度を使った探索戦略を持つドメインシェルを構築した.

各データに対して,伝播対象になる表ラベルとタグが切り替わるまで伝播の対象にならない裏ラベルの二つを設け,伝播処理を表ラベルだけに限ることによる効率のよい伝播処理アルゴリズムを示した.タグの切

り替えに関しては，

（1）トピックスに対してそれらを含む環境の集合へのポインタを持ち，表ラベルから裏ラベルへ，あるいは，裏ラベルから表ラベルへ移動する環境を効率的に探索する；

（2）環境が半順序を成すことを利用し，上記環境の集合を上限，下限を押さえることにより記録する；

という手段で，その効率的な実現方法を考案した．

謝辞 タグ付きATMSに関する議論をしていただき，また，評価実験を手伝っていただいた加納隆氏他（日立西部ソフトウエア(株)），割り当て問題用ドメインシェルを実現してくださった米谷尚久氏，本論文を丁寧に読んでいただき，有益な指摘をしていただいた絹川博之博士（(株)日立製作所），渡邊担博士（電気通信大学）に感謝します．

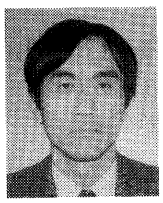
参考文献

1) de Kleer, J.: An Assumption-Based Truth Maintenance System, *Artif. Intell.*, Vol.28, pp.127–162 (1986).
2) de Kleer, J.: Choices without Backtracking, *Proc. AAAI-84*, pp.79–85 (1984).
3) 飛鳥井：Situated ATMS, 情報処理学会研究報告 89-AI-66, 1989年9月22日 (1989).
4) de Kleer, J.: Extending ATMS, *Artif. Intell.*, Vol.28, pp.163–196 (1986).
5) de Kleer, J. and Williams, B.C.: Back to Backtracking: Controlling the ATMS, *Proc. AAAI-86*, pp.910–917 (1986).
6) 山之内ほか：仮説の選択が可能なATMS, 第37回情報処理学会全国大会論文集, pp.1449–1450 (1988).

(平成6年2月4日受付)
(平成7年7月7日採録)

古賀 明彦（正会員）

1958年生．1982年京都大学理学部数学科卒業．1984年同大学数理解析研究所修士課程修了．修士での専門はプログラム変換．同年，(株) 日立製作所システム開発研究所に入社し，プログラミング環境，エキスパートシステム構築ツール，特に仮説推論機構，ヒューマンインタフェースの研究に従事．現在は，ネットワークを使った広域情報サービスのインタフェースを研究している．AIのCSCWへの適用に興味を持つ．日本ソフトウエア科学会会員．



増位 庄一（正会員）

1972年京都大学工学部卒業．1974年同大学院修士課程修了．同年（株）日立製作所入所．同社システム開発研究所にて制御システム，知識工学システム，オブジェクト指向システムなどの研究に従事．1981～82年米国カーネギーメロン大学客員研究員としてリアルタイムAIシステムの研究を行う．1994年以降，同社ソフトウエア開発本部市場開発センタ副センタ長．IEEE, AAAI, 人工知能学会などの会員．