Vol. 38 No. 3 情報処理学会論文誌 Mar. 1997

定理証明器SATCHMOREの効率化に関する一手法

何 立風[†] 島尻 優香[†] 巣 宇燕[††]
世木 博久[†] 伊藤 英則[†]

本稿では，定理証明器SATCHMOREに有効性の概念を組み込んだ$\mathcal{A}$-SATCHMOREを提案する．前向き推論に用いる非ホーン節を選ぶ基準として「関連性」の概念を導入したSATCHMORE[8]が提案されている．しかし，実際には証明に貢献しないアトムも関連があるとしてしまうことにより，探索空間の爆発を起こしてしまうこともある．そこで，$\mathcal{A}$-SATCHMOREでは，関連のあるアトムの中でも，さらに証明に有効であるもののみに注目し，証明に用いる非ホーン節をさらに制限することによって探索空間を絞りこんでいる．また，本方法に基づく定理証明器を計算機上に実装し，実験を行い効果を確認した．

Making Theorem Prover SATCHMORE More Efficient

LIFENG HE,[†] YUKA SHIMAJIRI,[†] YUYAN CHAO,[††] HIROHISA SEKI[†]
and HIDENORI ITOH[†]

We present a refinement of SATCHMORE, called $\mathcal{A}$-SATCHMORE, by incorporating availability checking on relevancy. We show that any relevant atom need not be marked unless it is available and no non-Horn clause need be selected unless all of its consequent atoms are marked availably relevant, i.e., it is totally availably relevant. In this way, $\mathcal{A}$-SATCHMORE is able to further restrict the use of non-Horn clauses (therefore reduce the search spaces) and makes the proof more goal-oriented. Furthermore, $\mathcal{A}$-SATCHMORE can be simply implemented in Prolog. We discuss how to incorporate availability checking on relevancy, describe our improvement, present the implementation. We also prove that our theorem prover is sound and complete, and provide examples to show the power of our availability approach.

1. はじめに

定理証明器は，エキスパートシステムや，自然言語処理システムなど多くの人工知能システムにおいて重要な要素技術となっている．定理証明器についてさまざまな研究がなされているが（たとえば，文献6), 12), 9), 13)），それらの中で有名なものの1つに，SATCHMO（SATisfiability CHecking with MOdel generation）[5]がある．

SATCHMOは，後向き推論，前向き推論の両方を用いた混合型の定理証明器である．ホーン節に対しては後向き推論（Prolog）を用い，非ホーン節に対しては，前向き推論を適用している．その原理（モデル生成法）は単純で実装が容易であるため，MGTP（Model Generation Theorem Prover）[3]などモデル生成法に基づいたシステムが開発されている．ただし，すべての違反節（つまり，本体が充足され，頭部が充足されていない節）を推論に用いるのは，証明に関連のない非ホーン節が存在する場合の証明が非効率になることがある．この問題に対して，SATCHMORE[8]では，**関連性**という概念を導入し，推論に用いる非ホーン節を選ぶためのある基準を与えることによって，証明の効率化を実現している．

しかるに，SATCHMOREは矛盾が導かれるか否かを調べるとき，および，ある非ホーン節が違反節であるかを調べるときに現れた関連のあるアトムにすべて印をつける．そのとき，その印をつけたアトムが推論を進めたときに本当に矛盾を導くのに貢献するかどうかは確認しないので，不必要なアトムも関連があるとして印をつけてしまう．そのため，SATCHMOREは無駄に探索空間を広げてしまう場合がある．このことは，さらに後節で例を用いて説明する．

本稿では，このようなSATCHMOREの問題点を解決するために，関連性の概念にさらに**有効**

† 名古屋工業大学
Nagoya Institute of Technology
†† 名古屋大学
Nagoya University

性（availability）の概念を組み込んだ定理証明器$\mathcal{A}$-SATCHMOREを提案する．$\mathcal{A}$-SATCHMOREでは，頭部のアトムがすべて関連があり，かつその後の推論の過程で矛盾を導くのに貢献するアトムを頭部に持つ非ホーン節のみを用いる．このように，用いる非ホーン節をさらに制限することによって，探索空間を縮小している．

以下に論文の構成を示す．2章ではSATCHMOとSATCHMOREについて説明する．3章では，有効性を解析する必要性を表す例と，有効性に関する定義を示す．4章では，提案する方法を組み込んだ定理証明器$\mathcal{A}$-SATCHMOREについて説明し，そのプログラムを示す．5章で，その正当性を示し，6章で実験結果により提案する方法の有効性を示す．

2. SATCHMOとSATCHMORE

本稿では，文献5)，8)と同様，領域限定（range-restricted）☆である節 $C_1;\ldots;C_n \leftarrow A_1,\ldots,A_m$（$A_i$, C_j はアトム，$m,n \geq 0$）の集合を対象とする．節の本体 $A_1,\ldots,A_m$ はアトム A_i の連言，節の頭部 $C_1;\ldots;C_n$ はアトム C_j の選言を表すとする．また，与えられた節集合 S は，ホーン節集合 BC と非ホーン節集合 FC に分けて扱う．さらに，ホーン節集合 BC はどのような問合せに対しても，決定可能（decidable）である（実際には，ホーン節集合に再帰節がを含まれなければ，この条件を保証できる）と仮定している．最後，$S \vdash A$ はアトム A が論理的に S から証明できることを表し，$\bot$ は矛盾を表す記号とする．

SATCHMOが与えられた節集合 $S = BC \cup FC$ の充足可能性を判定するときの流れを以下に示す．

I は基底アトムの集合（初期状態は空集合）を表す．

(1) $BC \cup I \vdash \bot$ が成り立つとき，$BC \cup FC \cup I$ は充足不能である．

(2) $BC \cup I \nvdash \bot$ であるときは，FC から本体が充足され，頭部が充足されていない違反節 K を選ぶ．違反節が存在しない場合，$BC \cup I$ の最小モデルは $BC \cup FC$ のモデルである．

(3) 違反節 K に対して，基底化された節 $K\theta$ の頭部に現れる各アトム $C_i\theta$ に関して，後向き推論のための節集合 $BC \cup I'$ $(I' = I \cup \{C_i\theta\})$ と FC を引数として，この手続きを再帰的に実行する．各 $C_i\theta$ に関して，$BC \cup FC \cup I'$ が充足不能であるとき，$BC \cup FC \cup I$ は充足不能である．

☆ 節が領域限定であるとは，節の頭部に現れる変数はすべて本体にも現れているということである．文献5)に指摘されたように任意の節集合が領域限定の形に変換できる．

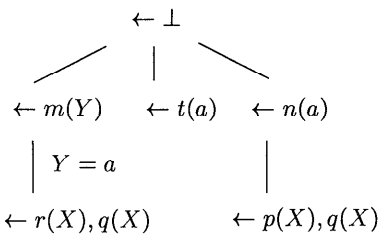


図1 ← ⊥ の失敗木 $FT_{\bot}^{BC}$
Fig. 1 The failure tree $FT_{\bot}^{BC}$.

SATCHMOの最大の問題点はすべての違反節を前向き推論に使ってしまうことである．文献8)で指摘されたように，証明に関連ないの違反節が存在する場合は，証明が非効率になってしまう．この問題に対して，SATCHMOREは関連のある違反節のみを前向き推論に使うことによって，効率化を実現している．

次に，SATCHMORE[8]に関する定義とアルゴリズムを概説する．

定義2.1[8] BC をホーン節集合，$\leftarrow G$ を $BC \nvdash G$ なるゴール（G はアトムの連言）とする．BC における $\leftarrow G$ のSLD木[4]を BC における $\leftarrow G$ の失敗木と呼び，FT_G^{BC} で表す．

例2.1 S を以下のような節集合とする．

BC : $\bot \leftarrow m(Y).$ $\bot \leftarrow t(a).$ $\bot \leftarrow n(a).$
$s(a).$ $s(b).$ $q(c).$
$m(a) \leftarrow r(X), q(X).$
$n(a) \leftarrow p(X), q(X).$

FC : $p(X); r(X) \leftarrow s(X).$
$m(a); n(a) \leftarrow s(a).$

ゴール $\leftarrow \bot$ の失敗木 $FT_{\bot}^{BC}$ を図1に示す．

定義2.2[8] BC をホーン節集合，$\leftarrow G$ を $BC \nvdash G$ なるゴールとする．ある基底アトム R が，FT_G^{BC} に現れるゴールの最左アトムであり，かつ $BC \nvdash R$ であるとき，R は BC において $\leftarrow G$ に関連があるという．また，アトム R が FT_G^{BC} に現れるゴールの最左アトムであるとき，R は BC において $\leftarrow G$ に拡張関連があるという．

例2.2 例2.1と同じ節集合について考えると，BC において $\bot$ に関連があるアトムは，$m(a), n(a), r(a), r(b), r(c), p(a), p(b), p(c), t(a)$ となる．また，拡張関連アトムは，$m(Y), r(X), n(a), t(a), p(X)$ となる．

SATCHMOREが与えられた節集合 $S = BC \cup FC$ の充足可能性を判定するときの流れを以下に示す．

```
?- op(1200,xfx,'→').

unsatisfiable:-
    bottom.
unsatisfiable:-
    not satisfiable.

satisfiable:-
    is_relevant(A,C),
    is_violated(A,C), !,
    satisfy(C),
    satisfiable.
satisfiable.

is_violated(A,C):-
    A, not C.

is_relevant(A,C):-
    retract(new_mark),
    (A→C), each_marked(C).
is_relevant(A,C):-
    new_mark, is_relevant(A,C).

satisfy(C):-
    component(X,C),
    (retract(marked(_)), fail; true),
    assertz(X),
    write('Asserting:'), write(X), nl,
    on_backtracking(retract(X)),
    not bottom.

component(X,(Y;Z)):-
    !, (X=Y; component(X,Z)).
component(X,X).

on_backtracking(X).
on_backtracking(X):-
    X, !, fail.

each_marked((C_1;CRest)):-
    !, marked(C_1),
    each_marked(CRest).
each_marked(C):-
    marked(C).

mark_unique(X):-
    marked(Y), is_instance(Y,X),!;
    subsume_and_mark(X).

subsume_and_mark(X):-
    marked(Y), is_instance(Y,X),
    retract(marked(Y)), fail.
subsume_and_mark(X):-
    (new_mark, !; asserta(new_mark)),
    assertz(marked(X)).

is_instance(X,Y):-
    not(X=Y), !, fail.
is_instance(_,Y):-
    var(Y), !.
is_instance(X,Y):-
    nonvar(X),
    functor(X,F,N), functor(Y,F,N),
    inst_args(X,Y,N).

inst_args(_,_,0):-!.
inst_args(X,Y,N):-
    arg(N,X,Ax), arg(N,Y,Ay),
    is_instance(Ax,Ay),
    N1 is N-1, inst_args(X,Y,N1).
```

図2 SATCHMOREプログラム
Fig. 2 SATCHMORE program.

I は基底アトムの集合（初期状態は空集合）を表す.

(1) $BC \cup I \vdash \bot$ が成り立つとき，$BC \cup FC \cup I$ は充足不能である.
（$\bot$ が導かれるかどうか調べている途中で現れた拡張関連アトムに印をつける.）

(2) $BC \cup I \not\vdash \bot$ であるときは，FC から頭部のアトムがすべて拡張関連である節（これを**拡張関連節**と呼ぶ）K を1つ選ぶ. 拡張関連節が存在しない場合，$BC \cup I$ の最小モデルは $BC \cup FC$ のモデルである.

(3) 節 K が違反節でないとき，(2)へ戻り別の拡張関連節を選ぶ.
（K の本体 A が充足されているかどうか調べている途中で現れた $\leftarrow A$ に拡張関連があるアトムに印をつける.）

(4) K が違反節のとき，その基底代入例 $K\theta$ の頭部はすべて関連のあるアトムである. ここで，基底化された節 $K\theta$ の頭部に現れる各アトム $C_i\theta$ に関して，後向き推論のための節集合 $BC \cup I'$ $(I' = I \cup \{C_i\theta\})$ と FC を引数として，この手続きを再帰的に実行する. 各 $C_i\theta$ に関して，$BC \cup FC \cup I'$ が充足不能であるとき，$BC \cup FC \cup I$ は充足不能である.

図2 に SATCHMORE のプログラム[8] を示す. ホーン節は，Prolog プログラムで与えられるとし，非ホーン節は，$A \rightarrow C$ で表す. 非ホーン節の頭部に現れるすべてのアトム $p(\vec{X})$ に関して，$p(\vec{X})$:- `mark_unique(`$p(\vec{X})$`), fail.` がプログラムに加えられているとする.

3. $\mathcal{A}$-関連性

推論過程の効率化のためには証明に不要な非ホーン節を極力用いない方がよい. なぜなら，非ホーン節を用いることにより，場合分けが生じ探索空間が倍増するためである. この推論に用いるべき非ホーン節を選ぶ基準に関してはいくつかの研究がされている（たとえば，文献2), 10), 13)）.

SATCHMOREでは，効率化のために前向き推論に用いる非ホーン節を関連がある違反節のみに限定している. この効率化方法は，関連があるとされたアトムが実際に矛盾を導くのに貢献した場合，非常に有効に働く. しかし，関連があるとされても導かれることの

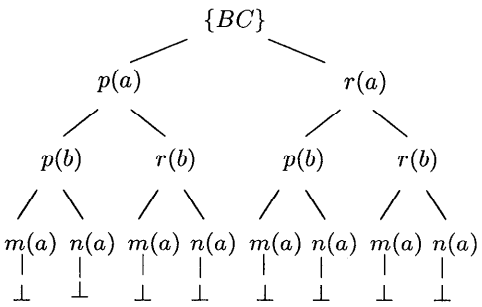


図3 SATCHMOREによる証明木
Fig. 3 The case tree made by SATCHMORE.

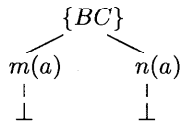


図4 最小の木
Fig. 4 The minimal tree.

ないアトムが存在していた場合，不必要な計算をしてしまう場合がある．以下で例を用いて説明する．

例 3.1 $S = BC \cup FC$ を例 2.1 と同じ節集合とする．SATCHMOREによる証明木を**図3**に示す．$r(X)$ と $p(X)$ が拡張関連アトムであるので，$X = a$ および $X = b$ のとき非ホーン節 $p(X); r(X) \leftarrow s(X)$ は，拡張関連節であり違反節である．しかしながら，この例題において，$r(X)$ と $p(X)$ のインスタンスは $\perp$ の証明の役に立たないばかりでなく，探索空間を広げる原因となっている．実際，$q(a)$ と $q(b)$ は節集合から導き出せないので，$r(a), r(b), p(a), p(b)$ は $\perp$ を証明するのには貢献しない．したがって，$X = a$ および $X = b$ という代入に関して，節 $p(X); r(X) \leftarrow s(X)$ を証明に用いる必要はない．そこで，その節を用いずに証明木を生成すると，**図4**のような最小の木が得られる．

SATCHMOREでは，頭部のすべてのアトムが関連があるときに，その節は全関連（totally relevant）であるといい，推論にはこの全関連である節のみを用いている．しかし，上述のように関連性の解析のみでは不十分な場合がある．そこで，本稿ではより強い条件である**$\mathcal{A}$-関連性**と呼ぶ制御を提案し，$\mathcal{A}$-関連性による制御でアルゴリズムの完全性が失われないことを示す．

ここで，$\mathcal{A}$-関連性に関する定義を示す．

定義 3.1 FC を与えられた節集合 S に含まれるすべての非ホーン節の集合とする．このとき，FC 中の節の頭部に現れるすべてのアトムの集合から，最も一般的な☆アトムを取り出して得られるアトムの集合を $\mathcal{S}_{ca}$ とする．

たとえば，例 2.1 の節集合について考えると，$\mathcal{S}_{ca} = \{p(X), r(X), m(a), n(a)\}$ となる．

定義 3.2（拡張失敗木） $S = BC \cup FC$ を節集合，$\leftarrow G$ を $BC \nvdash G$ を満たすゴール（G はアトムの連言）とする．以下の操作によって得られる木を $\leftarrow G$ の S における**拡張失敗木**と呼び，EFT_G^S で表す．

(1) BC における $\leftarrow G$ のSLD失敗木 FT_G^{BC} を初期状態とする．

(2) $g = \leftarrow A_1^*, \ldots, A_{j-1}^*, A_j, A_{j+1}, \ldots, A_j$ $(n \geq 1, j \geq 1)$ を生成された木に現れるゴールとする．

O_1) $H\sigma = A_j\sigma$ なる最汎単一化置換（mgu）σ が存在するすべてのホーン節 $H \leftarrow C_1, \ldots, C_l$ に関して，g のサブゴール $\leftarrow A_1^*\sigma, \ldots, A_{j-1}^*\sigma, C_1\sigma, \ldots, C_l\sigma, A_{j+1}\sigma, \ldots, A_n\sigma$ を生成する．

O_2) $B\sigma = A_j\sigma$ なる mgu σ が存在するアトム $B \in \mathcal{S}_{ca}$ に関して，g のサブゴール $\leftarrow A_1^*\sigma, \ldots, A_{j-1}^*\sigma, A_j^*\sigma, A_{j+1}\sigma, \ldots, A_n\sigma$ を生成する．

O_3) g に対して O_1 および，O_2 のいずれの操作も適用できないとき，ゴール g に関する処理を終了する．このとき，g は$\mathcal{A}$-失敗であるという．

(3) 生成された木に対して(2)のどの操作も適用できないとき，処理は終了する．このとき，すべての葉は$\mathcal{A}$-失敗であるか，あるいはサブゴールに含まれるすべてのアトムに $*$ がついた状態（これを$\mathcal{A}$-成功という）となっている．

直観的には，$*$ がついたアトムは FC から導かれる可能性があることを示す．

例 3.2 $S = BC \cup FC$ を例 2.1 と同じ節集合とする．このとき，$S_{ca} = \{r(X), p(X), m(a), n(a)\}$ であり，$\leftarrow \perp$ の S における拡張失敗木 $EFT_\perp^S$ は**図5**のようになる．

図5中のゴール $\leftarrow r(X), q(X)$ について考える．まず，$r(X)$ に対して処理 O_2 を適用し，$\leftarrow r^*(X), q(X)$ が得られる．次に，$q(X)$ に対して処理 O_1 を適用し，mgu $\sigma = \{c/X\}$ より，$\leftarrow r^*(c)$ が得られる．このとき，葉 $\leftarrow r^*(c)$ は$\mathcal{A}$-成功である．

節集合 $S = BC \cup FC$ とゴール $\leftarrow G$ に関して，$BC \nvdash G$ であるとき，BC が決定可能であり，$\mathcal{S}_{ca}$ が

☆ あるアトム A_i が，アトム集合において最も一般的であるとは，その集合に含まれる他のどのアトムも A_i をインスタンスとしないということである．

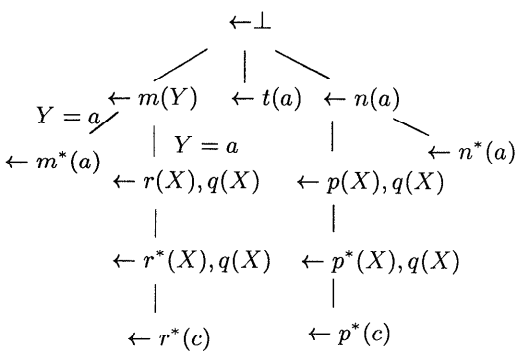


図5 ← ⊥ の拡張失敗木
Fig. 5 The extended failure tree $EFT_{\perp}^{S}$.

有限であることより，有限の大きさを持つ EFT_G^S が生成できる．この$\mathcal{A}$–成功葉に含まれるアトムは，非ホーン節により導かれる可能性があり，かつ矛盾を導くのに貢献するアトムである．本方法でも，SATCHMOREと同様，矛盾を導くために貢献する$\mathcal{A}$–成功葉の最左アトムに注目する．

定義 3.3 $S = BC \cup FC$ を与えられた節集合，G を $BC \nvdash G$ なるゴールとする．あるアトム B が BC において G の**$\mathcal{A}$–関連アトム**であるとは，B が，EFT_G^S の$\mathcal{A}$–成功葉の最左アトムから $*$ を取り除いたアトムの基底代入例であり，かつ $BC \nvdash B$ を満たすことをいう．

例 3.3 $S = BC \cup FC$ を例 2.1 と同じ節集合とする．BC における ⊥ の$\mathcal{A}$–関連アトムは，$m(a), n(a), r(c), p(c)$ となり，例 2.2 で示した関連アトムよりも少なくなっている．

しかしながら，$\mathcal{A}$–関連アトムを正確に計算するのは非常に複雑な処理である．なぜなら，EFT_G^S 中の$\mathcal{A}$–成功葉の最左アトムのインスタンスの中でも BC から導かれるものは$\mathcal{A}$–関連アトムではないので，それを取り除く処理が必要だからである．たとえば，与えられた節集合 S にホーン節 $r(a)$ が存在し，$r^*(X), p^*(X)$ が $EFT_{\perp}^S$ 中の$\mathcal{A}$–成功葉であるとすると，$BC \vdash r(a)$ より $r(a)$ は S において ⊥ の$\mathcal{A}$–関連アトムではない．

そこで，手続きの簡単化のため，まず以下に示す拡張$\mathcal{A}$–関連アトムを計算し，その後で$\mathcal{A}$–関連アトムであるかを確かめるという方法をとる．

定義 3.4 $S = BC \cup FC$ を与えられた節集合とし，G を $BC \nvdash G$ なるゴールとする．あるアトム B が S において G の**拡張$\mathcal{A}$–関連アトム**であるとは，B が EFT_G^S の$\mathcal{A}$–成功葉の最左アトムから $*$ を取り除いたアトムと共通の代入例を持つことである．

最後に，$\mathcal{A}$–関連節（拡張$\mathcal{A}$–関連）の定義を示す．ここでは，非ホーン節の一部は違反節でないことが検査してあるとする．

定義 3.5 I を基底アトムの集合，M を違反節でないことが検査済みの非ホーン節の集合とする（$M \subseteq FC$）．節 K が (M, I) に関する$\mathcal{A}$–関連節（拡張$\mathcal{A}$–関連節）であるとは K の頭部のアトムがすべて (1) $BC \cup I$ において ⊥ に$\mathcal{A}$–関連（拡張$\mathcal{A}$–関連）があるか，(2) $BC \cup I$ において M 中のある節の本体 A に$\mathcal{A}$–関連（拡張$\mathcal{A}$–関連）があることをいう．

以下では簡単のため，文脈より I および M が明らかなときは，単に$\mathcal{A}$–関連節（拡張$\mathcal{A}$–関連節）という表現を用いる．

4. $\mathcal{A}$–SATCHMOREアルゴリズム

本章では，我々の提案する$\mathcal{A}$–SATCHMOREのアルゴリズムについて説明する．$\mathcal{A}$–SATCHMOREでは，⊥ が導かれるかを調べるとき，および，節 K が違反節であるかどうかを調べるときに拡張$\mathcal{A}$–関連アトムに印をつける．そして，$\mathcal{A}$–関連で違反節である非ホーン節 K のみを前向き推論に用いる．

文献 8) に述べられているとおり，SATCHMOREではSATCHMOに比べ関連のあるアトムを計算するための処理時間が必要となる．本方法では，さらに詳しい解析を行うためよりその処理が大きくなってしまう．また，その解析の際には同じ計算が何度も繰り返されている．そこで，その繰返しを抑えるために，以下の操作を組み込んだ．

BC からすべての負節を取り除き，$EFT_{\perp}^{BC \cup FC}$ のすべての$\mathcal{A}$ 成功葉からすべての $*$ を取り除き，それを負節として加えたものを BC' とする．このとき，$BC \cup FC \vdash \perp$ が成り立つとき，また，その場合に限り，$BC' \cup FC \vdash \perp$ が成り立つ．そこで，負節の代わりに，$EFT_{\perp}^{BC \cup FC}$ の$\mathcal{A}$–成功葉を加えた BC' を用いることによって，$EFT_{\perp}^{BC \cup FC}$ の計算の一部を省略することができる．

以下に，$\mathcal{A}$–SATCHMOREアルゴリズムの概略を示す．

(1) $BC \vdash \perp$ が成り立つとき，$BC \cup FC$ は充足不能である．
（⊥ が導かれるかどうか調べている途中で現れた拡張$\mathcal{A}$–関連アトムに印をつける．）

(2) $BC \nvdash \perp$ であるとき，BC からすべての負節を取り除き，$EFT_{\perp}^{BC \cup FC}$ のすべての$\mathcal{A}$–成功葉を加えたものを BC' とする．

(3) $BC' \cup FC$ の充足可能性を調べる．I は基底アトムの集合（初期状態は空集合）を表す．

```
?- op(1200,xfx,'→').
unsatisfiable:-
   bottom.
unsatisfiable:-
   not  satisfiable.
bottom:-
   n_clause(A),
   is_available(A,B),
   (B==[ ]; asserta(new_n_clause(B),
     first(B,X), mark_unique(X), fail).
satisfiable:-
   is_relevant(A,C),
   is_violated(A,C), !,
   satisfy(C),
   satisfiable.
satisfiable.
is_relevant(A,C):-
   retract(new_mark),
   A → C, each_marked(C).
is_relevant(A,C):-
   new_mark, is_relevant(A,C).
is_violated(A,C):-
   is_available(A,B),
   (B==[ ], not  C;
    first(B,X), mark_unique(X), fail).
is_available([ ],[ ]).
is_available([X|A],B):-
   X, is_available(A,B).
is_available([X|A],B):-
   b_clause(D,X),
   append(D,A,E),
   is_available(E,B).
is_available([X|A],[X|B]):-
   ca_set(S_ca), member(X,S_ca),
   is_available(A,B).
satisfy(C):-
   component(X,C),
   (retract(marked(_)), fail; true),
   asserta(X),
   on_backtracking(retract(X)),
   not  new_bottom.
new_bottom:-
   new_n_clause(A),
   new_is_available(A,B),
   (B==[ ]; first(B,X), mark_unique(X), fail).
new_is_available([X|A],B):-
   X, new_is_available(A,B).
new_is_available(A,A).
first([X|_],X).
```

図6 $\mathcal{A}$-SARTCHMOREプログラム
Fig. 6 $\mathcal{A}$-SATCHMORE program.

(a) $BC' \cup I \vdash \bot$ が成り立つとき，$BC' \cup FC \cup I$ は充足不能である．
（$\bot$ が導かれるかどうか調べている途中で現れた拡張$\mathcal{A}$-関連アトムに印をつける．）

(b) $BC' \cup I \not\vdash \bot$ であるときは，FC から拡張$\mathcal{A}$ 関連節 K を1つ選ぶ．拡張$\mathcal{A}$ 関連節が存在しない場合，$BC' \cup I$ の最小モデルは $BC' \cup FC$ のモデルである．

(c) 節 K が違反節でないとき，(b) へ戻り別の拡張$\mathcal{A}$-関連節を選ぶ．
（K の本体 A が充足されているかどうか調べている途中で現れた $\leftarrow A$ に拡張$\mathcal{A}$-関連があるアトムに印をつける．）

(d) K が違反節のとき，その基底代入例 $K\theta$ は$\mathcal{A}$-関連である．基底化された節 $K\theta$ の頭部に現れる各アトム $C_i\theta$ に関して，後向き推論のための節集合 $BC' \cup I'$ ($I' = I \cup \{C_i\theta\}$) と FC を引数として，この手続きを再帰的に実行する．各 $C_i\theta$ に関して，$BC' \cup FC \cup I'$ が充足不能であるとき，$BC' \cup FC$ は充足不能である．

(4) $BC' \cup FC$ が充足不能のとき，$BC \cup FC$ が充足不能である．さもなければ充足可能である．

ここで，**図6** に$\mathcal{A}$-SATCHMOREのプログラムを示す．ただし，**mark_unique(X)**，**is_component(C)**，**on_backtracking(X)**，**each_marked(C)** はSATCHMOREと同様であるので省略してある．

負節 $A \rightarrow \bot$（A はアトムの連言）は **n_clause(A)**，その他のホーン節 $A \rightarrow H$（H はアトム）は **b_clause(A,H)**，非ホーン節は $A \rightarrow C$（C はアトムの連言）という形式で与えるとする．また，$\mathcal{S}_{ca}$ を **ca_set(**$\mathcal{S}_{ca}$**)** としてあらかじめ与えておく．

$\mathcal{A}$-SATCHMOREは目標 **unsatisfiable** で実行する．まず，目標 **bottom**（i.e., $\bot$）を呼ぶ．ここでは，負節 **n_clause(A)** を1つ選び，$EFT_A^{BC \cup FC}$ の$\mathcal{A}$-成功葉 B を求める（**is_available(A,B)**）．A が証明されない（B が空でない）とき，**new_n_clause(B)** をデータベースに付け加え，その最左アトムに印をつけ（**mark_unique**），目標 **is_available(A,B)** は失敗する．ここで，BC のみで充足不能であることが示されれば目標 **bottom** は成功し，そうでない場合は有限失敗する．

次に，目標 **satisfiable** が実行される．まず，目標 **is_relevant(A,C)** が呼ばれ，頭部のアトムがすべて拡張$\mathcal{A}$-関連である非ホーン節 $A \rightarrow C$ を求める．

目標 `is_relevant(A,C)` が成功するとその節 $A \rightarrow C$ が違反節であるかどうかを調べる(`is_violated(A,C)`). また, SATCHMOREと同様, 違反節であるかのチェックによって**実際に** $\mathcal{A}$-関連であるかを確認することができる. ただし, SATCHMOREと違い, 節の本体 A が証明されなかったとき, $\mathcal{A}$-SATCHMOREでは目標 `is_available(A,B)` により求められる$\mathcal{A}$-成功葉の最左アトムのみに印をつける.

$\mathcal{A}$-関連かつ違反節である節が見つかると目標 `satisfy(C)` が呼ばれる. 目標 `satisfy(C)` は C からアトム C_i を選び, 部分解釈 I を I' に拡張する. ここで, 目標 `new_bottom` を呼び, 矛盾が導かれるかどうかを調べる. SATCHMOREが $BC \cup I' \vdash \bot$ が成り立つかどうかを調べるのに対し, 本方法では, $BC' \cup I' \vdash \bot$ が成り立つかどうかを調べる. ここで, $BC' \cup I'$ が充足可能である(つまり, 目標 `new_bottom` が失敗する)場合, 目標 `satisfy` は成功し, $BC' \cup I'$ に対して目標 `satisfiable` が再帰的に呼び出される. このとき, 負節 `n_clause(A)` ではなく, $EFT_{\bot}^{BC \cup FC}$ の$\mathcal{A}$-成功葉 `new_n_clause(A)` を呼ぶことによって, $EFT_{\bot}^{BC \cup FC}$ の計算を省略することができる. new また, $BC' \cup I'$ が充足不能である場合, C_i から別のアトムを選択して I を拡張する. C に現れるどのアトムで拡張した部分解釈においても目標 `satisfy` が失敗したとき, 目標 `satisfiable` は失敗し, 目標 `unsatisfiable` が成功する.

5. 正 当 性

本章では, $\mathcal{A}$ SATCHMOREの健全性と完全性を示す.

定理 5.1(健全性) $\mathcal{A}$-SATCHMOREの目標 `unsatisfiable` が成功するとき, 与えられた節集合は充足不能である.

証明: $\mathcal{A}$-SATCHMOREで生成される証明木は違反節の選ぶ順序を変えることによって, SATCHMOでも得られること, および, SATCHMOの健全性より証明される. □

次に, $\mathcal{A}$-関連性を用いることによって完全性が失われないことを証明するための補題を示す.

補題 5.2 I を基底アトムの集合, W を (I, W) に関して$\mathcal{A}$-関連のある違反節でない基底非ホーン節の集合とする. $BC \cup FC$ が充足不能な基底節集合であり, $BC \cup I$ が充足可能な基底ホーン節集合であるとき, (I, W) に$\mathcal{A}$-関連である節が $FC - W$ 中に存在する.

証明: 補題 5.2 が偽, つまり, $FC - W$ 中に$\mathcal{A}$-関連節が存在しないと仮定する. この仮定に基づいて $BC \cup FC$ にモデルが存在することを示し, $BC \cup FC$ が充足可能であることと矛盾することを示す. $\mathcal{M}$ を (I, W) に関して$\mathcal{A}$-関連のあるアトムと, $BC \cup I \not\vdash A$ かつ $\mathcal{S}_{ca}$ 中のどのアトムともmguが存在しないアトム A に偽を割り当てた解釈とする. このとき, 解釈 $\mathcal{M}$ において $BC \cup FC$ のすべての節が充足されていることを示す.

(1) W 中の節は $\mathcal{M}$ において充足されている. なぜならば, W 中の各節は違反節ではない, つまり, 本体に少なくとも1つ$\mathcal{A}$-関連アトムか, あるいは $BC \cup I \not\vdash A$ かつ $\mathcal{S}_{ca}$ 中のどのアトムともmguが存在しないアトムが存在し, そのアトムは仮定より $\mathcal{M}$ において偽となっている.

(2) $FC - W$ 中の節は $\mathcal{M}$ において充足されている. なぜならば, $FC - W$ 中に$\mathcal{A}$-関連節が存在しないという仮定より, 各節の頭部は, $\mathcal{M}$ において充足されている.

(3) BC 中の各節は $\mathcal{M}$ において充足されている. なぜならば, 頭部のアトムが偽であるとすると, その節は$\mathcal{A}$-関連でも非ホーン節から導かれることもないアトムを少なくとも1つ含む失敗木を持つので, その本体は充足されてないからである. □

$\mathcal{A}$-SATCHMOREの完全性は以下の定理によって示される. 証明は, 補題 5.2 と文献 1), 7) の手法を用いて文献 8) と同様に示せるので, ここでは省略する.

定理 5.3(完全性) BC はすべての問合せに対して決定可能(decidable)である(Prologで計算が有限終了である)とする. このとき, $BC \cup FC$ が充足不能な節集合であれば, $\mathcal{A}$-SATCHMOREの目標 `unsatisfiable` が成功し, それは与えられた節集合が充足不能であることを示す.

6. 実 行 例

例 2.1 で, 関連性はあるが$\mathcal{A}$-関連ではないアトムによって探索空間が増大する場合があることを示した. ここでは, 本稿で提案する$\mathcal{A}$-関連を用いた定理証明器の動きと有効性の概念を組み込むことの効果を確かめるためのいくつかの例を示す.

まず, SATCHMO, SATCHMOREと$\mathcal{A}$-SATCHMOREの性能を比較するための例を示す.

例 6.1 以下の節集合について考える.

BC: $\bot \leftarrow p, m, h.$ $\bot \leftarrow q, n, h.$
$\bot \leftarrow s.$ $\bot \leftarrow t.$ $r.$

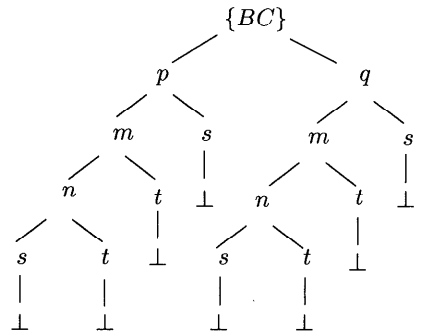


図7 SATCHMOによる証明木
Fig. 7 The case tree made by SATCHMO.

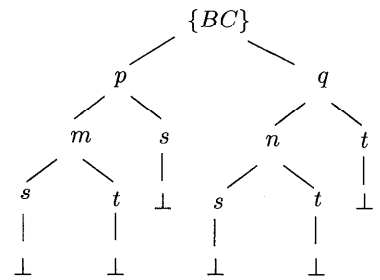


図8 SATCHMOREによる証明木
Fig. 8 The case tree made by SATCHMORE.

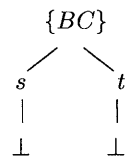


図9 $\mathcal{A}$-SATCHMOREによる証明木
Fig. 9 The case tree made by $\mathcal{A}$-SATCHMORE.

FC: $p;q \leftarrow r.$ $\quad m;s \leftarrow r.$
$\quad n;t \leftarrow r.$ $\quad s;t \leftarrow r.$

SATCHMO, SATCHMORE, $\mathcal{A}$-SATCHMOREにより生成される証明木をそれぞれ図**7**, 図**8**, 図**9**に示す.

図より, この例題ではSATCHMOはすべての非ホーン節を証明に用いていることが分かる. SATCHMOREは一部無駄な計算を省いている. $\mathcal{A}$-SATCHMOREはhが非ホーン節からも導かれないことが分かるので, 最後の非ホーン節のみを証明に用いている. この例に関しては, $\mathcal{A}$-SATCHMOREが最小な証明木を生成している.

次に, 単純な例を用いてモデル生成法において有効性を調べる必要性があることを示す.

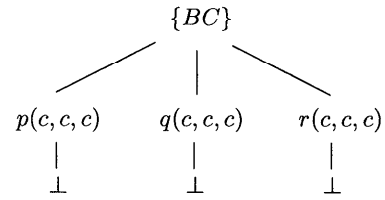


図10 $\mathcal{A}$-SATCHMOREによる証明木
Fig. 10 The case tree made by $\mathcal{A}$-SATCHMORE.

例 6.2 以下の節集合について考える.

BC: $p(X,Y,Z), t(X,Y,Z) \rightarrow \bot.$
$\quad q(X,Y,Z), t(Y,Z,X) \rightarrow \bot.$
$\quad r(X,Y,Z), t(Z,X,Y) \rightarrow \bot.$
$\quad s(a). \quad s(b). \quad s(c). \quad t(c,c,c).$
FC: $s(X), s(Y), s(Z)$
$\quad \rightarrow p(X,Y,Z); q(Y,Z,X); r(Z,X,Y).$

明らかに, $X = Y = Z = c$ のときのみ, 与えられた節集合から矛盾が導ける. SATCHMOREは有効性を調べないため, $p(X_1,Y_1,Z_1)$, $q(X_2,Y_2,Z_2)$, $r(X_3,Y_3,Z_3)$ を関連があるアトムとして印を付ける. よって, $X = Y = Z = a$ のとき, 非ホーン節は関連があり, かつ違反節になってしまう. しかし, $p(a,a,a)$, $q(a,a,a)$, $r(a,a,a)$ はまったく矛盾を導くのには貢献しないばかりでなく, 枝分かれをおこし探索空間を広げてしまう. 証明のために, SATCHMORE (SATCHMOとMGTP同様) では 3^{26} 個のモデル候補が生成されてしまう.

$\mathcal{A}$-SATCHMOREは有効性チェックによって, $p(c,c,c)$, $q(c,c,c)$, $r(c,c,c)$ のみを$\mathcal{A}$-関連として印を付ける. そのため, この例において$\mathcal{A}$-SATCHMOREは最小の証明木 (図**10**) を生成し, 0.02秒以下の実行時間ですむ.

最後, Schubert's Steamroller問題[11]を用いてSun SPARCstation5/85 MHzの上でいくつかの実験を行った. この例題は, SATCHMOREの関連性による枝刈りの効果を示すのに用いられているが, この例題に少し変更を加えると, その効果は現れなくなる. ここではその変更した例題においても, $\mathcal{A}$-関連性を用いることによる効果が得られることを示す.

例 6.3 以下の節集合 (Schubert's Steamroller) について考える.

$\bot \leftarrow wolf(X), fox(Y), eats(X,Y).$
$\bot \leftarrow wolf(X), grain(Y), eats(X,Y).$
$\bot \leftarrow bird(X), snail(Y), eats(X,Y).$
$\bot \leftarrow animal(X), animal(Y), eats(X,Y),$
$\qquad\qquad grain(Z), eats(Y,Z).$

表1 実験結果
Table 1 The experimental results for different provers.

	Steamroller	Steamroller + the clause (2)	Steamroller + the clauses (3) and (2)
SATCHMO	0.04 sec	$> 2.8 \times 10^3$ sec	> 600 sec
SATCHMORE	2.35 sec	2.37 sec	$> 4.5 \times 10^4$ sec
$\mathcal{A}$-SATCHMORE^1	2.87 sec	2.90 sec	3.7 sec
$\mathcal{A}$-SATCHMORE^2	3.10 sec	3.15 sec	4.20 sec

$wolf(w)$. $fox(f)$. $bird(b)$.
$snail(s)$. $caterpillar(c)$. $grain(g)$.
$animal(X) \leftarrow wolf(X)$. $animal(X) \leftarrow fox(X)$.
$animal(X) \leftarrow snail(X)$. $animal(X) \leftarrow bird(X)$.
$animal(X) \leftarrow caterpillar(X)$.
$plant(X) \leftarrow grain(X)$. $plant(i(X)) \leftarrow snail(X)$.
$plant(h(X)) \leftarrow caterpillar(X)$.
$smaller(X,Y) \leftarrow caterpillar(X), bird(Y)$.
$smaller(X,Y) \leftarrow snail(X), bird(Y)$.
$smaller(X,Y) \leftarrow bird(X), fox(Y)$.
$smaller(X,Y) \leftarrow fox(X), wolf(Y)$.
$eats(X,Y) \leftarrow bird(X), caterpillar(Y)$.
$eats(X,i(X)) \leftarrow snail(X)$.
$eats(X,h(X)) \leftarrow caterpillar(X)$.

$$eats(X,Y); eats(X,Z) \leftarrow animal(X), animal(Y), smaller(Y,X), plant(W), eats(Y,W), plant(Z). \quad (1)$$

元の例題においては，単純に違反節を用いるだけで最小の証明木が生成される．SATCHMMOは充足不能であることを証明するのに0.04秒しかかからない．しかし，節：

$$quicker(X,Y); quicker(Y,X) \leftarrow animal(X), animal(Y). \quad (2)$$

が節 (1) の前に付け加えられたとすると，SATCHMOは 2.8×10^3 秒以上の計算時間がかかる．

SATCHMOREは元の例題に関して2.35秒かかるが，節 (2) を付け加えた場合でもほぼ同じ時間しかかからない[8]．ただし，さらに節：

$$eats(X,Y) \leftarrow quicker(X,Y), smaller(Y,X). \quad (3)$$

が節 (2) の前に付け加えられたとする．このとき，SATCHMOとSATCHMOREは同じ証明木を生成するが，SATCHMOREは関連のあるアトムを計算し印をつけるための時間が必要な分，SATCHMOよりも多く時間がかかってしまう．

たとえば，$quicker(w,w)$ は明らかに充足不能性の証明には貢献しないが，SATCHMOREは $quicker(w,w)$ も関連があるアトムとして印をつけてしまう．そのため，$X = Y = w$ のとき，節 (2) は関連があり，かつ違反節であるので証明に用いられる．このとき，$quicker(w,w)$ は，まったく矛盾を導くのには貢献しないばかりか，枝分かれをおこし探索空間を広げることになる．

$\mathcal{A}$-SATCHMOREでは $smaller(w,w)$ は$\mathcal{A}$-関連アトムではない．よって，節 (2) が選ばれることはなく，最小な証明木が生成される．

3つの定理証明器の実行時間を**表1**に示す．なお，$\mathcal{A}$-SATCHMORE^1 は与えられた負節を `new_n_clause(B)` に置き換える処理を行った場合の実行時間であり，$\mathcal{A}$-SATCHMORE^1 はその処理を行わなかった場合の実行時間である．

7. おわりに

本稿では，定理証明器SATCHMOREに有効性の概念を組み込んだ$\mathcal{A}$-SATCHMOREを提案した．それにより，関連のあるアトムの数が少なくなり，証明に用いるべき非ホーン節をさらに絞り込むことができるので，無駄な計算を省くことができるようになった．また，実際に計算機上に実装しいくつかの実験を行うことにより，モデル生成法における有効性の概念の必要性を示した．

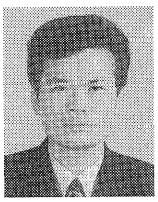
$\mathcal{A}$-SATCHMOREによって生成される証明木の各枝は，それぞれ独立に探索可能である．そこで，$\mathcal{A}$-SATCHMOREの並列実装が今後の課題として考えられる．

参考文献

1) Chang, C.L. and Lee, K.C.T.: *Symbolic Logic and Mechanical Theorem Proving,* Academic Press, New York (1973).
2) Demolombe, R.: An Efficient Strategy for Non-horn Deductive Databases, *Theoretical Computer Science*, Vol.78, pp.245–249 (1991).
3) Fujita, M., Slaney, J.K. and Hasegawa, R.: New Results in Mathematics by a Parallel Theorem Prover on a Parallel Inference Machine, *Technical Report*, ICOT-TR-1221, ICOT, Tokyo (1992).
4) Lloyd, J.W.: *Foundations of Logic Programming, 2nd edition*, Springer-Verlag, Heidelberg (1987).

5) Manthey, R. and Bry, F.: SATCHMO: A Theorem Prover Implemented in Prolog, *Proc. 9th Conference on Automated Deduction* (1988).
6) Loveland, D.W.: Mechanical Theorem Proving by Model Elimination, *J. ACM, 15*, pp.236–251 (1968).
7) Loveland, D.W.: *Automated Theorem Proving: A Logic Basis,* North-holland, Amsterdam (1978).
8) Loveland, D.W., Reed, D.W. and Wilson, D.S.: SATCHMORE: SATCHMO with RElevancy, *Journal of Automated Reasoning*, Vol.14, pp.325–351 (1995).
9) McCune, W.: OTTER 2.0 User' Guide', Technical Report, ANL-90/9, Mathematics and Computer Science Divsion, Argonne National Laboratory, Illinois (1990).
10) Ramsay, D.W.: Generating Relevant Models, *Journal of Automated Reasoning*, Vol.7, pp.359–368 (1991).
11) Stickel, M.E.: Schubert's Steamroller Problem: Formulations and solutions, *Journal of Automated Reasoning*, Vol.2, pp.89–101 (1986).
12) Stickel, M.E.: A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler, *Journal of Automated Reasoning*, Vol.4, No.4, pp.353–380 (1988).
13) Stickel, M.E.: Upside-down Meta-interpretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction, *Journal of Automated Reasoning*, Vol.13, pp.189–210 (1994).

(平成8年6月24日受付)
(平成9年1月10日採録)



何　立風（学生会員）

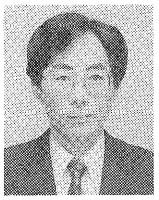
1982年中国西北軽工業学院自動制御学科卒業．1994年名古屋工業大学工学研究科博士前期課程電気情報工学専攻修了．現在，同大学院工学研究科博士後期課程在学中．論理プログラミング，定理証明，知識データベース，マルチエージェント協調計算，様相論理等に興味を持つ．

島尻　優香（正会員）

1994年名古屋工業大学工学部電気情報工学科卒業．1996年同大学院工学研究科博士前期課程電気情報工学専攻修了．現在，同大学知能情報システム学科助手．論理プログラミング，演繹データベース等に興味を持つ．人工知能学会会員．

巣　宇燕

1984年中国西北軽工業学院機械製造工学科卒業．現在，名古屋大学大学院人間情報文化工学研究科博士前期課程在学中．画像処理および図面理解，CAD，定理証明等に興味を持つ．人工知能学会学生会員．



世木　博久（正会員）

1979年東京大学工学部計数工学科卒業．1981年同大学院工学系研究科修士課程修了．同年4月より三菱電機（株）中央研究所に勤務．1985～1989年（財）新世代コンピュータ技術開発機構に出向．1992年4月より名古屋工業大学工学部知能情報システム学科助教授．工学博士．論理プログラミング，演繹データベース等に興味を持つ．電子情報通信学会，人工知能学会，ACM，IEEE Computer Society各会員．

伊藤　英則（正会員）

1974年名古屋大学大学院工学研究科博士課程電気・電子専攻満了．工学博士号取得．同年日本電信電話公社入社，横須賀研究所勤務．1985年（財）新世代コンピュータ技術開発機構出向．1989年より名古屋工業大学教授，現在知能情報システム学科所属，人工知能学会理事．これまでに，数理言語理論とオートマトン，計算機ネットワーク通信OS，知識ベースシステムなどの研究と開発に従事．電子情報通信学会，人工知能学会，ファジィ学会各会員．