定理証明器SATCHMOに関する新しい効率化手法

何 立風[†] 巣宇 燕[††]
中村 剛士[††] 伊藤 英則[††]

定理証明器 SATCHMO ではすべての違反節を前向き推論に用いるため，関連がない非ホーン違反節が存在する場合は探索空間の爆発を起こしてしまうこともある．その問題の解決には関連がある違反節だけを前向き推論に用いる方法がいくつか提案されている．本稿では，SATCHMO の効率化を図るための新しい手法を組み込んだ I-SATCHMO を提案する．I-SATCHMO では，SATCHMO と同様，最初に発見した違反節を前向き推論に用いる．ただし，実際の推論に貢献した違反節の頭部アトムを有用アトムとしてマークする．また，前向き推論に用いられる違反節が証明に必要とされるのはその節のすべての頭部アトムが有用であるときに限られる．そのため，I-SATCHMO は処理中の違反節のある頭部アトムが有用でないと判明したとき，その節に関する残りの処理をカットする．このように，無駄な推論を防止することによって，証明の効率化を実現している．

A New Improving Method for SATCHMO

Lifeng He,[†] Yuyan Chao,[††] Tsuyoshi Nakamura[††]
and Hidenori Itoh[††]

We present a new improving method for SATCHMO's model generation theorem priving approach. We use the violated non-Horn clauses in the same way as SATCHMO, but record the consequent atoms as useful whenever they have contributed to our reasoning. We show that a splitting of a non-Horn clause is necessary only if all of its consequent atoms are useful for our reasoning. As soon as one of the consequent atoms of a violated non-Horn clause used for forward chaining is found not to be useful, the remaining splitting over its consequence can be immediately pruned away. In this way, much of redundant search space can be eliminated.

1. まえがき

定理証明器 SATCHMO[6] の原理（モデル生成法）は単純で実装が容易であるため，自動推論の分野に注目されている[2),3),5)]．ただし，すべての違反節☆を推論に用いるため，証明に関連のない非ホーン違反節が存在する場合には，探索空間が爆発的に増大することがある．

証明に関連がある違反節だけを推論に用いれば，無駄な推論を防げることが明らかである．そのため，違反節を選ぶ基準として，「関連性」（relevancy）の概念を導入した SATCHMORE[5]，さらに「有用性」（availability）の概念を組み込んだ $\mathcal{A}$–SATCHMORE[3] が提案されている．

しかるに，それらの基準によって事前に関連があると認識された違反節は実際の推論と関連がないことがある．そのとき，探索空間は無駄に広げられてしまう．このことは，さらに後章で例を用いて説明する．

本稿では，SATCHMO に新たな効率化手法を組み込んだ定理証明器 I-SATCHMO（an Improvement of SATCHMO）を提案する．I-SATCHMO では，SATCHMO と同様，最初に発見した違反節を前向き推論に用いる．ただし，その違反節の各頭部アトムが推論に貢献するかどうかをその後の推論によってチェックし，推論に貢献しない頭部アトムを持つ違反節は証明に関連がないと判断する．そのため，I-SATCHMO は推論に用いられた違反節のある頭部アトムが推論に貢献しないと判明したとき，その節に関する残りの処理をカットする．このように，無駄な推論を防止し，証明の効率化を実現している．

† 愛知県立大学情報科学部
Faculty of Information Science and Technology, Aichi Prefectural University
†† 名古屋工業大学知能情報システム学科
Department of Intelligence and Computer Science, Nagoya Institute of Technology

☆ 違反節とは本体が充足され，頭部が充足されていない節をいう．

以下に論文の構成を示す．2 章では SATCHMO，SATCHMORE および$\mathcal{A}$–SATCHMORE について概説する．3 章では，新しい手法の必要性を明らかにする例を示し，I-SATCHMO を概説する．4 章で，I-SATCHMO の妥当性を証明し，5 章で実験結果により新しい手法の有効性を示す．

2. SATCHMO，SATCHMORE および $\mathcal{A}$-SATCHMORE

本稿では，文献 5)，6) と同様，領域限定（range-restricted）☆である節 $A_1,\ldots,A_m \rightarrow C_1;\ldots;C_n$ (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 （本体アトムという）の連言，節の頭部 $C_1;\ldots;C_n$ はアトム C_j （頭部アトムという）の選言を表すとする．また，$n = 0$，つまり，節の頭部が空であるとき，その節を $A_1,\ldots,A_m \rightarrow false$ で表し，負節という．一方，$m = 0$，つまり，節の本体が空である場合，その本体を $true$ で表す☆☆．さらに，与えられた節集合 S は，すべての負節からなる節集合 NC（Negative Clauses）と残りの節の集合 FC（Forward chaining Clauses）に分けて扱う．最後に，$S \vdash A$ は A が論理的に S から証明できることを表し，$A \in I$ は A が集合 I に属することを意味する．

SATCHMO，つまり，モデル生成法が与えられた節集合 $S = NC \cup FC$ の充足可能性を判定するために，空集合から S の各節を順番に満足させることを試しながら，S のモデルを求めていく．S のすべての節を満足するモデルを見つけることができれば，S が充足可能であり，そうでなければ，S が充足不能である．SATCHMO のアルゴリズムを以下に示す．

I は与えられた節集合のモデル候補（初期状態は空集合）を表す．

(1) 負節集合 NC にモデル候補 I に関する違反節 $A_1,\ldots,A_m \rightarrow false$ が存在する場合，$I \vdash A_1,\ldots,A_m$ のため，$NC \cup I \vdash false$ が成り立つ．そのため，$NC \cup FC \cup I$ は充足不能である．

(2) 負節集合 NC にモデル候補 I に関する違反節が存在しないときは，FC から I に関する違反節 $A_1,\ldots,A_m \rightarrow C_1;\ldots;C_n$ （つまり，$I \vdash A_1,\ldots,A_m$ かつ $I \nvdash C_1;\ldots;C_n$）を探す．違反節が存在しない場合，I は S のモデルであるため，S が充足可能である．違反節を見つけた場合，違反節の頭部に現れる各アトム C_i $(1 \leq i \leq n)$ に関して，モデル候補 I を $I' = I \cup \{C_i\}$ に拡張し，この手続きを再帰的に実行する．各 C_i に関して，$NC \cup FC \cup I'$ が充足不能であるとき，$NC \cup FC \cup I$ は充足不能である．

SATCHMO の推論過程は以下に定義される証明木で直観的に表現できる．

定義 2.1 与えられた節集合を $S = NC \cup FC$ とする．以下の操作によって構成される木を SATCHMO による証明木と呼ぶ．

(1) 根ノードを空ノードとし，ϕ で表す．

(2) D（基底アトム）を生成された証明木に現れるノードとし，I_D を証明木の根ノードからノード D までの枝に現れるすべての基底アトムの集合（モデル候補）とする．

(a) NC において I_D に関する違反節 $A_1,\ldots,A_m \rightarrow false$ が存在する場合，$NC \cup I \vdash false$ が成り立つため，ノード D の下に葉ノード $false$ を生成し，その枝に対する処理を終了させる．

(b) NC において I_D に関する違反節が存在しないとき，FC から I_D に関する違反節を探す．違反節が存在しない場合，ノード D の下に SAT という葉ノードを生成し，証明木の構成手続きを終了させる．一方，違反節 $A_1,\ldots,A_m \rightarrow C_1;\ldots;C_n$ を前向き推論に用いるとき，ノード D の下に子ノード $C_1, \cdots, C_n$ を生成する．

証明木の任意の内部ノード（つまり，葉ノード以外のノード）D において，そのノードのすべての枝が葉ノード $false$ で終了した場合，$S \cup I_D$ は充足不能であることを意味する．つまり，I_D は与えられた節集合のモデルまでに拡張できないことを表す．これから，説明を簡単にするために，そのようなノードが**充足不能**であるという．また，ノード D の下に葉ノード SAT が生成された場合，I_D が与えられた節集合のモデルであることを示す．最後に，証明木のすべての枝が葉ノード $false$ で終了したときのみ，与えられた節集合は充足不能であることを表す．

例 2.1 S を以下の節集合とする．

$$NC: \quad c, g \rightarrow false. \quad d \rightarrow false. \quad e \rightarrow false. \quad f \rightarrow false.$$

☆ 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 6) 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．節の本体が基底アトムによって満足されれば，その節の頭部アトムが必ず基底アトムになるのは，領域限定である節が持つ最大の特徴である．

☆☆ 任意のモデルにおいて，$true$ が真，$false$ が偽である．

$$FC: \quad true \rightarrow a;\ b.$$
$$true \rightarrow c;\ d.$$
$$true \rightarrow e;\ f.$$

SATCHMO による証明木を**図 1** に示す．証明木のすべての枝は $false$ で終了したため，与えられた集合は充足不能であることが証明された．

Prolog で実装した SATCHMO のプログラムを**図 2** に示す．証明は深さ優先探索によって行う．

SATCHMO の最大の問題点はすべての違反節を前向き推論に用いるということである．証明に必要のない非ホーン違反節が存在する場合は，証明が非効率的になってしまう．たとえば，例 2.1 では違反節 $true \rightarrow a;\ b$ のどの頭部アトムでも $false$ の証明に役立たないため，この違反節を前向き推論に用いるのは探索空間を広げる原因となっている．

この問題に対して，証明に必要である違反節だけを前向き推論に用いる方法がいくつか提案されている．SATCHMORE[5] は関連のある違反節のみを前向き推論に使う．関連のある節とはすべての頭部アトムが**関連アトム**である節をいう．そのとき，関連アトムはゴール $false$，および関連でありかつ違反でない節の本体の導出に貢献できるアトムである．いい換えれば，関連アトムは関連の前向き推論の推進にただちに貢献できるアトムである．例 2.1 では S から $false$ を導出する試みの結果，‘c, g’，‘d’，‘e’ および ‘f’ がサブゴールとされている．そのとき，関連アトムは c，d，e および f である．そのため，違反節 $true \rightarrow a;\ b$ は関連でないことが分かり，推論に使うことがない．SATCHMORE による証明木を**図 3** に示す．そのサイズは図 1 に示された証明木のサイズの半分になっていることが分かる．

$\mathcal{A}$–SATCHMORE[3] は，SATCHMORE の関連性概念にさらに有効性の概念を取り込んだ証明器である．$\mathcal{A}$–SATCHMORE では**有効**なサブゴールの最も左のアトムのみを関連アトムとしてマークする．そのとき，有効なサブゴールはそのサブゴールのすべてのアトムが与えられた節集合から導出可能なサブゴールである．そのため，関連アトムの数が減少することによって必要でない節の使用を防止できる．たとえば，例 2.1 では S から $false$ を導出する試みの結果，‘c, g’，‘d’，

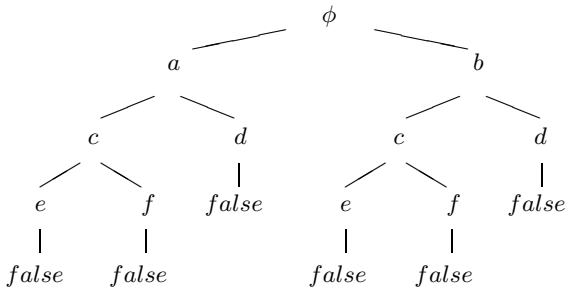

図 1 SATCHMO による証明木
Fig. 1 The proof tree constructed by SATCHMO.

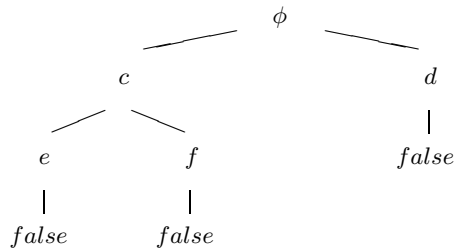

図 3 SATCHMORE による証明木
Fig. 3 The proof tree constructed by SATCHMORE.

```
:-op(1200,xfx, --->).
satisfiable:-
   (A ---> false),A,!,fail.

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

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

```
satisfy(C):-
   component(X,C),
   asserta(X),
   on_backtracking(retract(X)).

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

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

図 2 SATCHMO のプログラム
Fig. 2 The SATCHMO program.

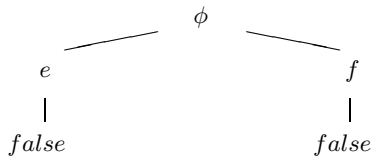

図 4 $\mathcal{A}$-SATCHMORE による最小証明木
Fig. 4 The proof tree constructed by $\mathcal{A}$-SATCHMORE.

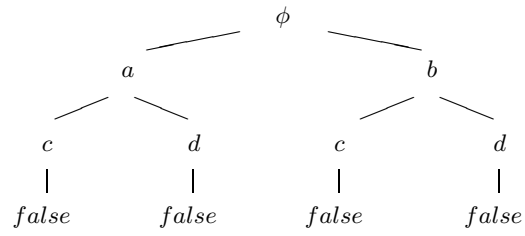

図 5 $\mathcal{A}$-SATCHMORE による証明木
Fig. 5 The proof tree constructed by $\mathcal{A}$-SATCHMORE.

'e' および 'f' がサブゴールとされている．しかし，アトム g は S のいかなる節の頭部にも現れないため，明らかに S から導出可能でないアトムである．仮にアトム c が導出されても g が導出されないので 'c, g' は証明に有効なサブゴールでないことが分かる．そのため，関連アトムは d, e および f に減少し，関連違反節は $true \rightarrow e;\ f$ だけとなる．$\mathcal{A}$–SATCHMORE による証明木は最小となり，**図 4** に示される．

3. I-SATCHMO の概略

ある基準に基づいて選んだ違反節だけを前向き推論に用いることによって探索空間を制限するアプローチには 2 つの問題がある．1 つは基準を満たす違反節を見つけ出す解析計算にかかるコストを無視できないことである．一般的には，その基準は厳しければ厳しいほどコストが高い．もう 1 つはそれらの基準で必要とされている違反節が実際の証明に必要でないことがある．そのとき，無駄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 を以下のような節集合とする．$\mathcal{A}$-SATCHMORE による証明木を**図 5** に示す．

$$
\begin{array}{lll}
NC: & a,\ e \rightarrow false. & b,\ f \rightarrow false. \\
 & c \rightarrow false. & d \rightarrow false. \\
FC: & true \rightarrow a;\ b. & \\
 & true \rightarrow c;\ d. & \\
 & true \rightarrow e;\ f. &
\end{array}
$$

S から $false$ を導出するとき，すべてのサブゴール 'a, e', 'b, f', 'c' および 'd' は有効である．その結果，違反節 $true \rightarrow a;\ b$ が必要であると判断され，前向き推論に用いられてしまう．しかしながら，この例題において，この節の頭部アトム a または b は実際の推論に使われていない．したがって，この違反節は証明に必要がない．

例 3.1 のような単純な場合であっても，節 $true \rightarrow a; b$ を推論から簡単に排除できない．このような場合には証明が無駄な探索をしてしまう．

違反節の頭部アトムが実際の推論に有用かどうか，その違反節を前向き推論に用いる前に判断するのは困難であるが，その違反節を前向き推論に用いた後に頭部の各アトムが実際の推論に使われたかどうかを確認するのは容易である．たとえば，例 3.1 では，節 $true \rightarrow a; b$ が前向き推論に用いられた後，頭部のアトム a と b のどちらも推論に使われていなかったことが明白である．

本稿では，前向き推論に貢献した違反節の頭部アトムを**有用アトム**という．そして，前向き推論に用いられた違反節の頭部アトムが実際の推論に貢献しないとき，その違反節が証明に必要でないと判断し，その節に関する残りの処理をカットすることによって効率化する方法を提案する．なお，この方法は前に紹介した改良法と完全に独立するため，SATCHMO，SATCHMORE および $\mathcal{A}$–SATCHMORE のいずれにも取り込むことができる．ここでは，説明を簡単にするために，SATCHMO に基づいて説明する．

SATCHMO は充足不能性における健全性と完全性を持つ証明器である．そのため，これから与えられた節集合は充足不能であると仮定する．SATCHMO はある違反節 $A_1, \ldots, A_m \rightarrow C_1; \ldots; C_n$ を処理する際，深さ優先探索による推論を行い，左から右への順で違反節の頭部アトムを処理していく．ある頭部アトム C_i に対して，証明木にはノード C_i を生成する（つまり，C_i を推論データベースに挿入する）．そして，ノード C_i のすべての枝が $false$ で終了することを証明したとき，証明手続きがバックトラックし，C_i を推論データベースから取り除き，次の頭部アトム C_{i+1} を選んで同様の処理を行う．したがって，C_i が推論データベースから取り除かれるまでに，推論に使われることがない場合，C_i は推論に貢献しないことが分かる．

一方，推論データベースに挿入された違反節の頭部アトムが推論に用いられるのは違反節の本体を導出する場合のみである．そのため，違反節を推論に用いるとき，その違反節のすべての本体アトムを有用アトムとしてマークすればよい．

I-SATCHMO では，与えられた節集合の充足可能性を判定する流れは基本的に SATCHMO と同じである．ただし，推論に用いられる違反節の本体の導出に貢献した頭部アトムを有用なアトムとしてマークする．一方，バックトラックによって推論データベースから取り除かれるときも有用アトムとしてマークされていない頭部アトムは有用でないと認識する．そして，ある頭部アトムが証明に有用でないと判明すれば，そのアトムを持つ違反節に関する残りの処理をカットする．

以下に，I-SATCHMO のアルゴリズムの概略を示す．

I は与えられた節集合 $S = NC \cup FC$ のモデル候補（初期状態は空集合）とする．

(1) NC にモデル候補 I に関する $I \vdash A_1, \ldots, A_m$ が成り立つ違反節 $A_1, \ldots, A_m \rightarrow false$ が存在する場合，$NC \cup I \vdash false$ のため，$NC \cup FC \cup I$ は充足不能である．そこで，すべての A_i（$1 \leq i \leq m$）を有用アトムとしてマークする．

(2) NC にモデル候補 I に関する違反節が存在しないとき，FC から I に関する違反節を探す．違反節が存在しない場合，I は $NC \cup FC$ のモデルであるため，$NC \cup FC$ が充足可能である．逆に，違反節 $A_1, \ldots, A_m \rightarrow C_1; \ldots; C_n$ を見つけたとき，すべての A_i（$1 \leq i \leq m$）を有用アトムとしてマークする．そして，その違反節の各頭部アトム C_j $1 \leq j \leq n$（左から右への順に）に関して，モデル候補 I を $I' = I \cup \{C_j\}$ に拡張し，この手続きを再帰的に実行する．ただし，C_j が推論データベースから取り除かれるとき，有用でない（つまり，有用アトムとしてマークされていない）と判明されたとき，または，すべての C_j に関して，$NC \cup FC \cup I'$ が充足不能であるとき，$NC \cup FC \cup I$ は充足不能である．

I-SATCHMO による証明木の生成は基本的に SATCHMO と同じである．ただし，違反節 $A_1, \ldots, A_m \rightarrow C_1; \ldots; C_n$ の頭部アトム C_j が有用でないと判明したとき，$C_{j+1}, \cdots, C_n$ に関する処理をカットする．

例 3.2 S を例 3.1 と同じ節集合とする．I-SATCHMO による証明木を**図 6** に示す．そこで，× はそこから先の処理をカットしたことを表す．

I-SATCHMO は違反節 $true \rightarrow a; b$ の頭部アトム a が推論に使われていなかったので，この節が証明に必要でないと判断し，頭部アトム b に対する処理をカットした．

I-SATCHMO のプログラムを**図 7** に示す．証明に貢献した頭部アトムを有用アトムとしてマークするために，推論に用いられる違反節の頭部アトム X を推論データベースに挿入すると同時に，atom(X, 0) も挿入する．ここで，フラッグ 0 は X が推論にまだ使用されていないことを示す．そして X が推論に貢献したことが確認できたとき，atom(X, 0) を atom(X, 1) に取り替える．そのとき，フラッグ 1 は X が推論に貢献したことを表す．その取り替えの作業は手続き mark_each によって行う．

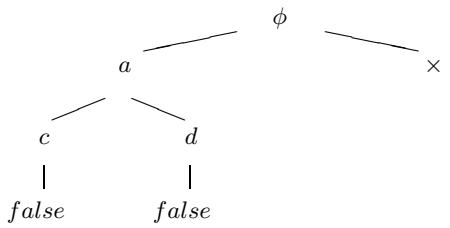

図 6 I-SATCHMO による証明木
Fig. 6 The proof tree constructed by I-SATCHMO.

D を証明木に現れる葉以外のノード，I_D をノード D が生成されたときの推論データベースにあるすべての頭部アトムの集合（つまり，証明木の根ノードからノード D までの枝に現れるすべてのアトムの集合）とする．I-SATCHMO はノード D における推論を行う際，satisfiable を実行する．まず，負節集合 NC から $I_D \vdash A$ のような違反節 $A \rightarrow false$ を求める．見つかれば，mark_each(A) で A のすべてのアトムを有用なアトムとしてマークし，satisfiable を失敗させる．一方，見つからない場合，is_violated(C) を呼んで FC から違反節を求める．

FC から違反節が見つからなかった場合，satisfiable が成功し，与えられた節集合が充足可能であることを示す．逆に，違反節 $A \rightarrow C$ が見つかった場合，mark_each(A) で A のすべてのアトムを有用なアトムとしてマークする．そして，satisfy(C) を呼び，C からアトム X を component(X, C) で選ぶ．そのとき，useless がデータベースに存在しない場合，atom(X, 0) をデータベースに挿入する．それは証明木においてノード D の下にノード X を生成することを意味する．そして，ノード X に対し，satisfiable が再帰的に呼び出される．

ノード X において satisfiable が失敗したとき，つまり，ノード X が充足不能であると証明したとき，is_necessary(X) を呼び，X および atom(X, Y) を推論データベースから取り除く．そのとき，フラッグ Y が依然 0 のままであれば（つまり，X が推論に用

```
:-op(1200,xfx,--->).
satisfiable:-
   (A ---> false), A,
     mark_each(A),!,fail.

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

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

satisfy(C):-
   (component(X,C);
     retract(useless),!,fail),
   (retract(useless),!,fail;true),
   asserta(X),asserta(atom(X,0)),
   on_backtracking(is_necessary(X)).

mark_each((X,A)):-
   retract(atom(X,_)),
   asserta(atom(X,1)),
   mark_each(A).
mark_each(X):-
   X=true;
   retract(atom(X,_)),
   asserta(atom(X,1)).

is_necessary(X):-
   retract(X),retract(atom(X,Y)),
     (Y=0,asserta(useless);true).

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

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

図 7 I-SATCHMO プログラム
Fig. 7 I-SATCHMO program.

いられたことがないならば），useless がデータベースに挿入され，X が有用でないことを示す．すると，次の component が成功するかどうかにかかわらず，retract(useless) で useless をデータベースから除去し，satisfy(C) を強制的に失敗させる．それによって，ノード D においてのコール satisfiable も失敗してしまう．一方，フラッグ Y が 1 であるとき（つまり，X が有用であるとき）の処理は SATCHMO とまったく同じように行う．

4. 正 当 性

本章では，I-SATCHMO の健全性と完全性を示す．まず，I-SATCHMO において必要がないとされる違反節の処理をカットすることが健全性に影響しないことを証明するための補題を示す．

補題 4.1 $S = NC \cup FC$ を充足不能の節集合，D を証明木のノード，P を D の親ノードとする．そのとき，頭部アトム D が証明に有用でなければ，D の親ノード P が充足不能である．

証明： I-SATCHMO のアルゴリズムによって，頭部アトム D が証明に有用であるかどうかという判断は，ノード D が充足不能であるのを証明した直後に行う．そのため，頭部アトム D が証明に有用でないと判別されたとき，証明木ではノード D のすべての枝が $false$ で終了することが分かる．つまり，すべての枝において，$NC \cup I_P \cup \{D\} \cup J_x \vdash false$ が成り立つ．ここでは，I_P は根ノードからノード P までの枝に現れる頭部アトムの集合，J_x はノード D から $false$ までのある枝に現れる頭部アトムの集合を表す．D が有用でないことから，D が J_x のどの要素または $false$ の導出にも貢献しなかったことが分かる．そのため，頭部アトム D がデータベースに存在していなくても，ノード D 以下の推論に影響しない．そうすると，ノード D における推論がそのままノード P で行える．したがって，ノード P が充足不能である．

I-SATCHMO の健全性は以下の定理によって示される．

定理 4.2 I-SATCHMO の目標 satisfiable が失敗するとき，与えられた節集合は充足不能である．

証明： 定理 4.2 は SATCHMO の健全性と補題 4.1 を用いて証明できる．

I-SATCHMO は SATCHMO と同じ順序に違反節を選ぶことによって，一部の枝がカットされること以外，I-SATCHMO による証明木と SATCHMO による証明木がまったく同じであることが分かる．そのため，証明木のあるノードに対して，I-SATCHMO は SATCHMO と同様に処理すれば，SATCHMO の健全性から I-SATCHMO の処理も健全であることが明

らかである．一方，I-SATCHMO にカットされたノードに対して，補題 4.1 により，そのノードの親ノードが充足不能であるので，SATCHMO の完全性により，SATCHMO はその親ノードにおいて充足不能であることを証明できる．すなわち，証明木の任意のノードが I-SATCHMO により充足不能であれば，SATCHMO によるものも充足不能である．つまり，I-SATCHMO は与えられた節集合が充足不能であることを示すことができれば，SATCHMO もその節集合が充足不能であることを示すことができる．SATCHMO の健全性により，I-SATCHMO の健全性が証明される．

I-SATCHMO の完全性は以下の定理によって示される．

定理 4.3 S が充足不能な節集合であれば，I-SATCHMO の目標 `satisfiable` が失敗し，それは与えられた節集合が充足不能であることを示す．

証明： 定理 4.3 は SATCHMO の完全性により証明できる．I-SATCHMO は証明木上の任意のノードに対して SATCHMO と同様に処理するか，または直接にそのノードが充足不能であると判定する．そのため，SATCHMO は与えられた節集合が充足不能であることを示すことができれば，I-SATCHMO もその節集合が充足不能であることを示すことができる．

5. 実 行 例

本章では，本稿で提案する効率化方法の効果を確かめるためのいつかの例を示す．すべての実験は Intel PentiumIII/500 MHZ の上で行った．なお，実行時間の単位は秒である．

例 5.1 以下の節集合☆について考える．

$$
\begin{aligned}
NC:\quad & p(X,Y,Z),\ t(X,Y,Z) \rightarrow false.\\
& q(X,Y,Z),\ t(X,Y,Z) \rightarrow false.\\
& r(X,Y,Z),\ t(X,Y,Z) \rightarrow false.\\
& v(X,Y,Z) \rightarrow false.\\
FC:\quad & true \rightarrow s(a).\\
& true \rightarrow s(b).\\
& true \rightarrow s(c).\\
& true \rightarrow u(c,c,c).\\
& s(X),\ s(Y),\ s(Z) \rightarrow p(X,Y,Z);\\
& \qquad\qquad q(X,Y,Z);\ r(X,Y,Z).\\
& u(X,Y,Z) \rightarrow t(X,Y,Z);\ v(X,Y,Z).
\end{aligned}
$$

明らかに，$X = Y = Z = c$ になるときのみ，与えられた節集合から反駁が導ける．$\mathcal{A}$-SATCHMORE は $p(X,Y,Z)$，$q(X,Y,Z)$ と $r(X,Y,Z)$ を有効関連アトムとして印を付ける．そうすると，たとえば，$X = Y = Z = a$ のときは 1 番目の非ホーン節が関連かつ違反節になってしまう．しかし，$p(a,a,a)$，$q(a,a,a)$ と $r(a,a,a)$ のいずれもまったく矛盾を導くのに貢献しないので，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，$\mathcal{A}$–SATCHMORE（もちろん，SATCHMO，SATCHMORE が同様である）は $3^{27} \times 2$ 個のモデル候補を生成し，適切な時間内に答えが導き出せなくなる．

I-SATCHMO は推論データベースに挿入された $p(c,c,c)$ 以外の p アトムが推論に使われないことから，$p(c,c,c)$ 以外の p アトムが推論に有用でないと認識する．そして，それらのアトムを持つ違反節は推論に必要がないと判断し，それらの節に対する処理をカットする．そうすると，I-SATCHMO は与えられた節集合の充足不能を示すために，0.01 秒の計算時間もかからない．I-SATCHMO による証明木は**図 8** に示される．

次に，Schubert's Steamroller 問題[8]（TPTP[9] の PUZ031-1 問題）を用いていくつかの実験を行った．ここではこの例題およびこの例題の拡張したいくつかのバージョンを用いて SATCHMO，SATCHMORE，$\mathcal{A}$–SATCHMORE と I-SATCHMO の性能を比較し，I-SATCHMO の効果を示す．

例 5.2 Schubert's Steamroller という問題は以下の節集合で表される．

NC:

$$
\begin{aligned}
& wolf(X), fox(Y), eats(X,Y) \rightarrow false.\\
& wolf(X), grain(Y), eats(X,Y) \rightarrow false.\\
& bird(X), snail(Y), eats(X,Y) \rightarrow false.\\
& animal(X), animal(Y), eats(X,Y),\\
& \qquad grain(Z), eats(Y,Z) \rightarrow false.
\end{aligned}
$$

FC:

$$
\begin{aligned}
& true \rightarrow wolf(w). \qquad true \rightarrow fox(f).\\
& true \rightarrow bird(b). \qquad true \rightarrow snail(s).\\
& true \rightarrow caterpillar(c). \qquad true \rightarrow grain(g).\\
& wolf(X) \rightarrow animal(X).\\
& \quad fox(X) \rightarrow animal(X).\\
& snail(X) \rightarrow animal(X).\\
& \quad bird(X) \rightarrow animal(X).\\
& caterpillar(X) \rightarrow animal(X).\\
& grain(X) \rightarrow plant(X).\\
& snail(X) \rightarrow plant(i(X)).
\end{aligned}
$$

☆ この節集合は定理証明問題集 TPTP[9] の SYN009-1 問題の拡張である．

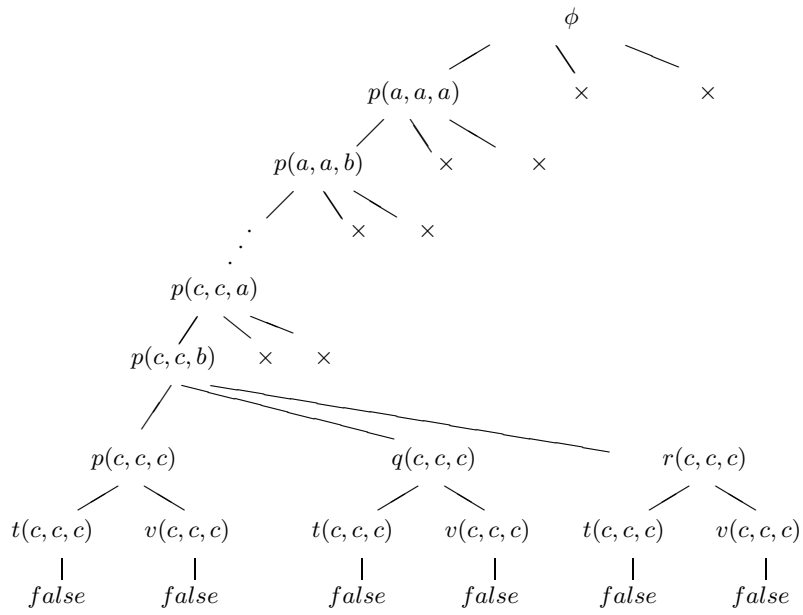

図 8 I-SATCHMO による証明木
Fig. 8 The proof tree constructed by I-SATCHMO.

$caterpillar(X) \rightarrow plant(h(X))$.
$caterpillar(X), bird(Y) \rightarrow smaller(X,Y)$.
$snail(X), bird(Y) \rightarrow smaller(X,Y)$.
$bird(X), fox(Y) \rightarrow smaller(X,Y)$.
$fox(X), wolf(Y) \rightarrow smaller(X,Y)$.
$bird(X), caterpillar(Y) \rightarrow eats(X,Y)$.
$snail(X) \rightarrow eats(X, i(X))$.
$caterpillar(X) \rightarrow eats(X, h(X))$.

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

元の例題を拡張するために以下のいくつかの節を用いる．

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

$$quicker(X,Y), stronger(X,Y) \rightarrow eats(X,Y). \quad (3)$$

$$animal(X), animal(Y), smaller(X,Y) \rightarrow stronger(Y,X);\ eats(Y,X). \quad (4)$$

いくつかの組合せを用いて実験を行い，その結果を**表 1** に示す．

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ため，SATCHMO は最も効率的に証明できる．SATCHMORE または $\mathcal{A}$–SATCHMORE は必要のない節集合の解析を行い，余分なコストがかかってしまう．I-SATCHMO は節が必要かどうかの解析にコストがあまりかからないため，計算スピードが SATCHMO と同じレベルにある．

しかし，節 (2) が節 (1) の前に付け加えられたとき，SATCHMO はこの節に対してあらゆる探索を行うため，68 秒以上の計算時間が要する．SATCHMORE または $\mathcal{A}$–SATCHMORE はこの節を選ばないので，計算時間がほとんど変わらない．また，いかなる quicker アトムも推論に使われないことから，I-SATCHMO は節 (2) による無駄な処理をカットし，必要な実行時間は前より少しだけ増えている．

さらに，節 (3) を節 (2) の前に付け加えるとすると，SATCHMORE は *quicker* アトムを関連アトムとマークしてしまうため，節 (2) のすべての基底例を前向き推論に用いる．解析に計算時間がかかるので，SATCHMORE は SATCHMO よりも非効率的になってしまう．$\mathcal{A}$–SATCHMORE はいかなる *stronger* アトムも導出可能でないと認識し，*quicker* アトムを関連アトムとマークしない．そのため，節 (2) が前向き推論から排除され，計算時間は節 (2) が付け加えられる前とほぼ同じである．一方，I-SATCHMO の実行時間はほとんど変わらない．

最後に，節 (4) が節 (2) の前に付け加えられたとき，すべての *stronger* アトムが導出可能となるので，$\mathcal{A}$–SATCHMORE も節 (2) を前向き推論に用いてしまう．より深い解析が行われたため，$\mathcal{A}$–SATCHMORE は SATCHMORE よりも多く時間がかかってしまう．

表 1 実験結果 I
Table 1 The experiment results I.

	NC+ FC+(1)	NC+FC +(2)+(1)	NC+FC+ (3)+(2)+(1)	NC+FC+(3) +(4)+(2)+(1)
SATCHMO	0.003	68	84	94
I-SATCHMO	0.007	0.009	0.012	0.31
SATCHMORE	0.05	0.05	5602	2445
$\mathcal{A}$-SATCHMORE	0.175	0.185	0.190	126143

表 2 実験結果 II
Table 2 The experiment results II.

	SATCHMO	SATCHMORE	$\mathcal{A}$-SATCHMORE	I-SATCHMO
MSC006-1	0.07	0.09	0.13	0.03
MSC007-1.008	19.90	71.70	76.48	20.75
PUZ-001-2	—	31	48	2.35
PUZ-018-2	—	0.05	0.08	1015
PUZ-023-1	1.02	0.26	0.35	0.03
PUZ-025-1	0.52	2.40	4.50	0.02
PUZ-030-1	0.59	0.57	0.81	0.23

ここでは，I-SATCHMO はいかなる *quicker* アトムも推論に使われていないから節 (2) のすべての基底例が推論に必要がないと判断し，それらの基底例における無駄な処理をカットしている．そのため，I-SATCHMO は依然効率的である．

我々はさらに定理証明問題集 TPTP[9] からいくつかの問題を用いて実験を行った．その実験結果を**表 2** に示す．なお，—は実行時間が 1 時間以上であることを表す．問題 MSC007-1.008 と問題 PUZ-018-2 以外，I-SATCHMO の効率は最も良い．問題 MSC007-1.008 には関連のない節が存在しないため，SATCHMO の性能は一番良い．I-SATCHMO の実行時間は SATCHMO の実行時間よりわずかに多く，SATCHMORE および $\mathcal{A}$-SATCHMORE は非効率になってしまう．また，充足可能な問題 PUZ-018-2 において，I-SATCHMO の効率は SATCHMORE および $\mathcal{A}$-SATCHMORE より効率が悪いが，SATCHMO より効率的である．

6. おわりに

本稿では，定理証明器 SATCHMO の新しい効率化方法を提案した．論理節の解析による前向き推論に用いる違反節を選ぶという従来の効率方法と違い，本稿で提案した方法は前向き推論に用いられた違反節の頭部アトムが推論に貢献したかに注目する．推論に貢献しない頭部アトムを持つ違反節は推論に必要でないと判断し，その節に関する残りの推論を行わないことによって無駄な探索を防止することができる．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，本手法はモデル生成法における有効な手法であることを示した．

本稿で提案した手法はある基準による前向き推論に用いる違反節を選ぶ効率化手法と独立であるため，本手法は SATCHMORE または $\mathcal{A}$–SATCHMORE に取り込むことができる．その実現が容易であるため，ここではこれ以上の議論をしないこととする．

本手法を一般化すると，証明にいったん用いられる論理節は結局証明に関連がないと判明したとき，その論理節に関する推論をカットするということになる．この考え方は HARP 定理証明器[7] に取り入れられた proof condensation 手法，Hyper Tableaux 定理証明器に取り込まれた level cut 手法[1] と類似である．そのため，I-SATCHMO はこういった効率化手法の SATCHMO のモデル生成法への拡張と実装であると考えられる．

また，重複な推論を防止できる folding up[4] 手法も次のように I-SATCHMO に取り込むことができる．$NC \cup FC$ を与えられた節集合，D をその証明木上のノードとするとき，ノード D が充足不能であると証明された場合，$NC \cup FC \cup U_D \vdash false$ が成り立つ．ここでは，U_D はノード D 以下の部分木を構成したときに有用とマークされたアトムの集合を表す．そのため，U_D が lemma としてその後の証明に利用できる．そのような定理証明器の実装や評価が今後の課題として考えられる．

参 考 文 献

1) Baumgartner, P., Furbach, U. and Niemela, I.: Hyper Tableaux, *Proc. European Workshop: Logics in Artificial Intelligence*, *JELIA*, LNAI 1126, pp.1–17 (1986).
2) Bryand, F. and Yahya, A.: Positive Unit Hyperresolution Tableaux and Their Application to Minimal Model Generation, *Journal of Automated Reasoning*, Vol.25, pp.35–82 (2000).
3) He, L., Chao, Y., Simajiri, Y., Seki, H. and Itoh, H.: $\mathcal{A}$-SATCHMORE: SATCHMORE with Availability Checking, *New Generation Computing*, Vol.16, pp.55–74 (1998).
4) Letz, R., Mayr, K. and Goller, C.: Controlled Integration of the Cut Rule into Connection Tableau Calculi, *Journal of Automated Reasoning*, Vol.13, pp.297–337 (1994).
5) Loveland, D.W., Reed, D.W. and Wilson, D.S.: SATCHMORE: SATCHMO with RElevancy, *Journal of Automated Reasoning*, Vol.14, pp.325–351 (1995).
6) Manthey, R. and Bry, F.: SATCHMO: A theorem prover implemented in Prolog, *Proc. 9th Conference on Automated Deduction*, LNCS 310, pp.415–434 (1988).
7) Opacher, F. and Suen, E.: HARP: A Tableau-Based Theroem Prover, *Journal of Automated Reasoning*, Vol.4, pp.69–100 (1988).
8) Stickel, M.E.: Schubert's Steamroller Problem: Formulations and solutions, *Journal of Automated Reasoning*, Vol.2, pp.89–101 (1986).
9) Sutcliffe, G. and Suttner, C. http://www.cs.jcu.edu.au/~ tptp

(平成 13 年 9 月 5 日受付)
(平成 14 年 2 月 13 日採録)

何 立風（正会員）

1997 年名古屋工業大学工学研究科博士後期課程修了．博士（工学）．現在，愛知県立大学情報科学部助教授，マルチエージェント，定理証明，知識データベース等に興味を持つ．

巣 宇燕

2000 年名古屋大学大学院人間情報文化工学研究科博士後期課程修了．現在，名古屋工業大学研究員．博士（学術）．定理証明，画像処理および図面理解，CAD 等に興味を持つ．

中村 剛士

1998 年名古屋工業大学工学研究科博士後期課程修了．博士（工学）．現在，同大学知能情報システム学科助手．感性情報，ソフトコンピューティング等に興味を持つ．人工知能学会会員．

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満了．工学博士号取得．同年 NTT 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．感性情報，自動推論，マルチメディア，マルチエージェント等に興味を持つ．電子情報通信学会，人工知能学会，ファジィ学会各会員．