知能と複雑系 110－1
(1998. 1. 22)

上昇型定理証明器 Hyper Tablaux への関連性試験の導入

太田 和彦　　岩沼 宏治

山梨大学工学部電子情報工学科

〒400 甲府市 武田 4-3-11 山梨大学工学部電子情報工学科

あらまし

本研究では，上昇型定理証明器 Hyper Tablau 演繹をさらに効率的にするため、関連性試験を導入する。先ず Hyper Tablau 演繹 を SATCHMORE の上昇型推論を行なうモデル生成部と下降型推論による反駁を行なう部分に分けて再構成する．更に SATCHMORE で用いられている関連性試験を組み込み，その完全性を示す．

キーワード　上昇型定理証明器，Hyper Tablaux，関連性試験

Hyper Tablaux with relevancy check

Kazuhiko OOTA　　Kouji IWANUMA

Department of Electrical Engineering and Computer Science,
Faculty of Engineering, Yamanashi University
4-3-11 Takeda, Koufu, Yamanashi 400, Japan

Abstract

In this paper, we make up relevancy check with Hyper Tablaux theorem prover using SATCHMORE's. Then we show its compleetness.

key words　bottom-up theorem prover, Hyper Tablaux, relevancy check

1 はじめに

本研究では，上昇型定理証明器 Hyper Tablau 演繹を効率的にするため、SATCHMORE で用いられている関連性試験を導入する。

代表的な一階論理式に対する上昇型定理器には，Manthey らによって作られた SATCHMO[4]，その拡張である SATCHMORE[3] などがある．これらの定理証明器は，上昇型証明によるモデル生成と，生成されたモデルに対する下降型証明による反駁を自然な形で表現しており，Prolog を用いて実装されている．Prolog の高速な推論を有効に利用したコンパクトで高速な定理証明器になっている．更に，SATCHMORE では，関連性試験を導入することにより，モデル生成に使用される節から，モデルに対する反駁で不要る節を排除している．しかし扱うことができる論理式が，領域限定なものに制限されており，一般の一階論理式の場合には，領域述語を導入して，一旦，領域限定式に変換した後に証明を行なわなければならない．領域述語を用いることにより，証明中でエルブラン領域の元の無駄な列挙が行なわれることになり，証明が非効率になる．

一方，Hyper Tablau 演繹 [1] は節形式を扱うタブロー [2] 形式の上昇型定理証明器である．SATCHMO に代表される上昇型定理証明器が扱う論理式を領域限定なものに限定しているのに対して，Hyper Tablau 演繹では，扱う論理式を領域限定でない一般の一階論理式に拡張している．これは，領域限定式の性質のうち，上昇型定理証明における部分モデルの構成時に必要となるもののみを，実行時に動的に用いることによって可能となっている．

本研究では，先ず Hyper Tablau 演繹を SATCHMORE の上昇型推論を行なうモデル生成部と下降型推論による反駁を行なう部分に分けて再構成する．更に SATCHMORE で用いられている関連性試験を組み込み，その完全性を示す．

本論文は，以下の構成になっている．先ず，第 2 節で，本論文中で使用する定義と記法を与える．第 3 節で，Hyper Tableau を定義する．この定義は，Hyper Tableau と SATCHMORE の関連性試験を外した版の定義を合わせたものになっている．第 4 節では，SATCHMORE の関連性試験と，その性質を紹介する．第 5 節で，Hyper TTableau に関連性試験を組み込み，第 6 節でその完全性を示す．第 7 節は，まとめと今後の課題である．

2 準備

本論文中では，一階述語論理と論理プログラミングの知識を仮定する．扱う論理式は，全て節形式とする．節集合 X に対して，X^g は X の元の全ての基礎例からなる集合とする．

タブローはノードにリテラルをラベル付けした順序木とする [1]．タブローの経路 b と，経路上の各ノードに付けられたリテラルを根から順番に並べた列，及び列の元からなる集合を同一視し，全て b で表す．また，節 C が b のモデルにおいて真であることを $b \models C$ と表す．

経路 b_1, b_2 に対して，b_1 が b_2 の部分経路であることを $b_2 \geq b_1$ で表す．

本研究で扱う定理証明器は節集合 S を扱う．S は予め，S 中の一部の Horn 節からなる集合 BC と $FC = S - BC$ に分割されているとする．但し，BC は，任意のリテラル集合 L に対して，$BC \cup L$ からの (Prolog による) 演繹の停止性が保証されているものとする．BC は空集合としても構わない．また BC, FC の元を，各々 BC 節と FC 節と呼ぶ．SATCHMORE 及び本研究の定理証明器では，FC 節はモデル生成に使用し，BC は生成したモデルの反駁に使用する．

定義 2.1 節 $C = A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ の，全ての異なる A_i, A_j が共通変数を持たないとき，C は純粋 (pure) であるという．また，節を純粋にする代入を純化代入と呼ぶ．

定義 2.2 BC,FC を与えられた BC 節集合と FC 節集合とする．また，$R(\subseteq FC)$ とする．T をタブロー，b を T の経路とする．このとき，$BC \cup b$ が矛盾を含めば，経路 b が閉じていると定義する．また，全ての経路が閉じているとき，タブロー T は閉じているという．

3 Hyper Tablau 演繹

本節では，Hyper Tablau 演繹を FC,BC 節集合を扱える形で定義する．

定義 3.1 節集合 $S = BC \cup FC$ とタブローから経路を選択する選択関数 f が与えられたとき，f を用いた S からの Hyper Tableau を以下で再帰的に定義する．

1. 単一のノードからなる $\top$ をラベル付けされたタブローは S からの Hyper Tableau である．このタブローを初期 Hyper Tableau と呼ぶ．
2. 以下により，新たな Hyper Tableau を生成する．
   (a) T が閉じていない Hyper Tableau なら，閉じていない経路 $b = f(T)$ を選択する．
   (b) $C \in FC$ を選択する．$C = A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ とする．
   (c) σ を $b \models (A_1 \wedge \cdots \wedge A_n)\sigma$ となる最も一般的な代入，π を $C\sigma$ に対する純化代入とする．
   (d) 新しい Hyper Tablau T' は T 中の b の葉ノードに $B_1\sigma\pi, \ldots, B_m\sigma\pi$ に対応する m 本の枝を伸ばしたタブローとする．

T から T' を生成する操作を Hyper 拡張と呼ぶ．

初期 Hyper Tableau T_0 から Hyper 拡張を繰り返して生成される Hyper Tableau の列 $T_0, T_1, \ldots$ に対して，

$$T_0 \vdash_{b_0, C_0, \sigma_0, \pi_0} T_1 \cdots T_n \vdash_{b_n, C_n, \sigma_n, \pi_n} T_{n+1} \cdots$$

を Hyper Tableau 演繹と呼ぶ．($b_i, C_i, \sigma_i, \pi_i$ は適宜省略して記述する．)

4 関連性試験

以下，Hyper Tableau 演繹に，SATCHMORE で使用されている関連性試験を導入する．関連性試験を用いると，モデル生成時に冗長な節の使用を削減することができる．

定義 4.1 b を経路，FC を FC 節集合，$R \subseteq FC$ とする．原子論理式 L が次のどちらかの条件を満たすとき，b と R で関連するという．

1. $b \cup BC$ から反駁を試みた SLD 失敗木の最左失敗アトムとして出現する．
2. ある $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \in R$ の A_i に対して，$b \cup BC$ から A_i の導出を試みた SLD 失敗木の最左失敗アトムとして出現する．

又，節 $C = A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ とする．全ての $B_1, \ldots, B_m$ が b と R で関連するとき，C は b と R で関連するという．

定義 4.2 Hyper Tableau 演繹 D 中に現われる経路 b に対して，$b_i \geq b$ かつ $b_{i+k} \geq b_i (k > 0)$ なる b_i の無限列 $b_0 (= b), b_1, b_2, \ldots$ を無限経路と呼ぶ．

定義 4.3 b を経路，BC を BC 節集合とする．$C = A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ を節とする．このとき，ある代入 σ が存在して，以下の条件を満たすとき，C は新規であるという．

1. $b \cup BC \models (A_1 \wedge \cdots \wedge A_n)\sigma$
2. $b \cup BC \not\models (B_1 \vee \cdots \vee B_m)\sigma$

定義 4.4 BC,FC を与えられた BC 節集合と FC 節集合とする．また，$R (\subseteq FC)$ とする．T をタブロー，b を T の経路とする．このとき，b が終結しているとは以下のどちらかを満たすときとする．

1. $FC - R$ 中に b と R で関連する節が存在しない．
2. 新規である節が存在しない．

定義 4.5 Tableau 演繹 D に対して，D 中のそれぞれの Hyper Tableau T_i で関連しかつ新規な節 C が，必ず，ある $T_j (j > i)$ で新規でなくなっているならば，D は公平であるという．

関連性試験については以下の補題が成り立つ．

補題 4.6 BC,FC を与えられた BC 節集合と FC 節集合，T をタブロー，b を T の経路とする．$R (\subseteq FC)$ を b の関連節のうち新規でないものの集合とする．$BC \cup b$ が充足可能かつ $BC \cup FC$ が充足不能であるならば，$FC - R$ 中に b と R で関連する節が存在する．

証明

$FC - R$ 中に b と R で関連する節が存在しないと仮定して矛盾を導く．

$BC \cup FC$ を構成する原子論理式の集合を $\mathcal{A}$ とし，エルブラン解釈 M を以下のように定める．

$$M = \{L \in \mathcal{A}^g \mid L \text{ は } b \text{ と } R \text{ で関連しない}\}$$

このとき，R 中の任意の節 $C = A_1, \ldots, A_n \to B_1, \ldots, B_m$ に対して，$M \models C$ が成り立つ．なぜならば，$M \not\models C$ とすると，代入 σ が存在して，$A_1, \ldots, A_n \in M$ かつ $B_1, \ldots, B_m \notin M$ となるが，これは C が関連節であることに反する．

また，$FC - R$ 中の任意の節 $C = A_1, \ldots, A_n \to B_1, \ldots, B_m$ に対して $M \models C$ である．なぜなら，仮定より C は b と R で関連しないので，ある B_i が b と R で関連しないことになり，$M \models B_i$ となる．よって $M \models C$.

また BC 中の節 $C = A_1, \ldots, A_n \to B_1, \ldots, B_m$ は，b と R との関連性の定義より，ある B_i が関連リテラルならば，必ず関連リテラル A_j が存在し，$M \not\models A_i$ となるので，$M \models C$ となる。

以上より，M は $BC \cup FC$ のモデルとなり，充足不可能性と矛盾する．

5 Hyper Tablaux への関連性試験の組み込み

Hyper Tableau 演繹を次のように変更し関連性試験を組み込む．

定義 5.1 節集合 $S = BC \cup FC$ とタブローから経路を選択する選択関数 f が与えられたとき，f を用いた S からの関連性試験付き Hyper Tableau を以下で再帰的に定義する．

1. top をラベル付けした，単一のノードのみからなるタブローは S からの Hyper Tableau である．
2. $R = \emptyset$ とする．以下により，新たな Hyper Tableau を生成する．
   (a) T が閉じていない Hyper Tableau なら，閉じていない経路 $b = f(T)$ を選択する．
   (b) b と R で関連する節 $C \in FC$ を選択する．
   (c) $C = A_1, \ldots, A_n \to B_1, \ldots, B_m$ とする．σ を $b \models (A_1 \wedge \cdots \wedge A_n)\sigma$ となる最も一般的な代入，π を $C\sigma$ に対する純化代入とする．σ が存在しないなら，R を $R \cup C$ として，(a) の選択を行なう．
   (d) C が新規であるかなら，新しい Hyper Tablau T' は T 中の b の葉ノードに $B_1\sigma\pi, \ldots, B_m\sigma\pi$ に対応する m 本の枝を伸ばしたタブローとする．

6 完全性

Hyper Tablaux の完全性定理は，次のように拡張できる．公平性や終結の定義が，Hyper Tablaux と異なっているところに注意する必要がある．

定理 6.1 S を節集合，D を公平で終結する S からの (無限) 演繹列とする．このとき，D 中の全てのタブローが閉じていないならば，S は充足可能である．

証明

簡単のため，$BC = \emptyset$ とする．

演繹列 D が有限の場合．$D = T_0 \vdash_{b_0} T_1 \vdash \cdots T_k$ とする．仮定より T_k は終結して閉じていない経路を持つ．このとき，終結の定義より，全ての $C \in S$ に対して，$b \models C$. よって b は S のモデルとなり，充足可能である．

演繹列 D が無限の場合．$D = T_0 \vdash_{b_0} T_1 \vdash \cdots T_n \vdash \cdots$ とする．演繹列中の無限系列 $p = b_0, b_1, \ldots$ を考える．ここで，$\mathcal{A}(p) = \bigcup_{binp} b$ とする．

$\mathcal{A}(p) \models S$ であることを示す．エルブランの定理より，$\mathcal{A}(p) \models S$ と $\mathcal{A}(p) \models S^g$ は同値であるので，ここでは $\mathcal{A}(p) \not\models S^g$ を仮定して背理法を用いて証明する．

仮定より，$\mathcal{A}(p) \not\models C'$ なる $C' \in S^g$ が存在する．ここで，$C \in S, C' = A_1, \ldots, A_n \to B_1, \ldots, B_m$, $C' = C\theta$ とする．このとき，$b_i \models A_1, \ldots, A_n$ なる b_i が存在する．

ここで，ある $b_j(\geq b_i)$ において，$B_1, \ldots, B_m$ が関連リテラルになるなら，公平性の定義より，$\mathcal{A}(p) \models S^g$ となり矛盾する．全ての $b_j(\geq b_i)$ において，$B_1, \ldots, B_m$ が関連リテラルにならない場合を考える．補題 4.6 より，関連性試験を用いないオリジナル Hyper Tableau を使用しても，C' をを用いた反駁を構成することはできない．よってオリジナル Hyper Tableau の完全性より，$\mathcal{A}(p) \models S^g$.

7 おわりに

本研究では，上昇型定理証明器 Hyper Tablaux に関連性試験を導入し，その完全性を示した．今後，実験を通して，Hyper Tablaux に対して，どの程度の効率化が為されたかを検証する必要がある．

参考文献

[1] Baumgartner, P., Furbach, U., Niemela, I. :Hyper Tablaux, *Proc. of European Workshop on Logics in Artificial Intelligence (JELIA '96)* (1996), pp.1–17.

[2] Fitting, M. :*First-Order Logic and Automated Theorem Proving, 2nd Edition* (1996) Springer-Verlag

[3] Loveland, D., Reed, D., Wilson, D. :SATCHMORE: SATCHMO with RElevancy, *J. Automated Reasoning* 14 (1995), pp.325–351.

[4] Manthey, R., Bry, F. :SATCHMO: atheorem prover implemented in Prolog, *Proc. of the Ninth Int'l Conf. on Automated Deduction*, 1988.

[5] Ramsay, A. :Generating Relevant Models *Journal of Automated Reasoning* 7 (1991), pp.359–368.