ICOT Technical Report: TR-0860

TR-0860

Modal Propositional Tableaux in a Model Generation Theorem Prover

by
M. Koshimura & R. Hasegawa

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191~5

Institute for New Generation Computer Technology

Modal Propositional Tableaux in a Model Generation Theorem Prover

Miyuki Koshimura and Ryuzo Hasegawa
ICOT Research Center

Abstract

A simple method for making a tableaux-style theorem prover is introduced. The method is based on the similarity between tableaux-style and model generation theorem provers. A modal propositional tableaux is implemented in a model generation theorem prover, called MGTP. In the implementation, we consider a set of input clauses for MGTP as being a programming language, and use that language to make a modal propositional tableaux. Making a tableaux-style theorem prover with this method clearly distinguishes the inference rule and its inference mechanism. Each tableaux inference rule is described in an MGTP input clause, and the inference mechanism is implemented by making MGTP. Separating the inference rule and inference mechanism makes it easy to develop tableaux-style theorem provers.

1 Introduction

A theorem prover tries to prove a theorem with several inference rules. The inference rule vary from prover to prover. For example, a resolution-based theorem prover uses a 'resolution' rule as its inference rule. On the other hand, a tableaux-based[7] theorem prover uses 'expansion' rules. In many existing theorem prover implementations, a theorem to be proven is given as input data to the theorem prover and its inference rules are embedded into its implementation, i.e., the inference rules are programmed as a part of the theorem prover. In our method, the inference rules themselves are given as input data to the theorem prover. This is akin to making a universal Turing machine if the existing theorem prover is regarded as being a Turing machine. Here, the theorem prover MGTP(Model Generation Theorem Prover)[3] corresponds to a universal Turing machine, and a modal propositional tableaux[1, 2], imitated in MGTP, corresponds to a Turing machine.

MGTP performs a computation according to a set of input clauses and determines its satisfiability. MGTP can imitate tableaux-style theorem proving by representing tableaux inference rules by its input clauses. This is also regarded as a meta-programmed tableaux prover on MGTP. A merit of meta-programming is its capacity for flexible representation. Actually, it is easy to make a modal propositional tableaux for several modal systems such as K, $K4$, T, $S4$ and $S5$, by replacing the MGTP input clauses that represent modality.

For example, a modal propositional tableaux for K can be built by writing 12 MGTP clauses that represent tableaux rules. The simplicity comes from the clear distinction between the representation of tableaux rules and its computation mechanism. Combinatorial computation and pattern matching, which act as an industrial kernel in theorem proving, are performed in MGTP.

Thus, isolating the representation of tableaux rules from the implementation of its computation mechanism lower the barrier to tableaux implementation.

$C1:\quad p(X), s(X) \rightarrow false.$
$C2:\quad q(X), s(Y) \rightarrow false.$
$C3:\quad q(X) \rightarrow s(f(X)).$
$C4:\quad r(X) \rightarrow s(X).$
$C5:\quad p(X) \rightarrow q(X); r(X).$
$C6:\quad true \rightarrow p(a); q(b).$

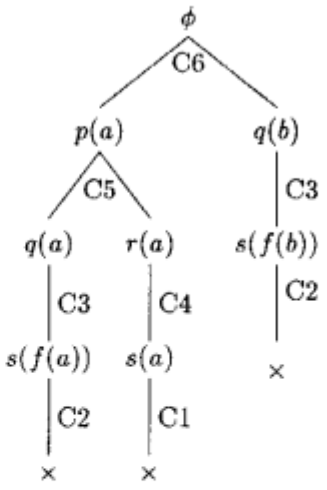

Figure 1: Sample MGTP Program and Proof Tree

2 MGTP

Throughout this paper, an MGTP clause is represented by an implicational form:

$$A_1, A_2, \ldots, A_n \rightarrow C_1; C_2; \ldots; C_m$$

where $A_i (1 \leq i \leq n)$ and $C_j (1 \leq j \leq m)$ are atoms; the antecedent is a conjunction of $A_1, A_2, \ldots, A_n$; the consequent is a disjunction of $C_1, C_2, \ldots, C_m$. A clause is said to be *positive* if its antecedent is $true (n = 0)$, *negative* if its consequent is $false (m = 0)$, and otherwise, *mixed* $(m \neq 0, n \neq 0)$.

The following two rules act on the model generation method.

- Model extension rule: If there is a clause, $A \rightarrow C$, and a substitution σ such that $A\sigma$ is satisfied in a model M but $C\sigma$ is not satisfied in M, extend the model M by adding $C\sigma$ to the model M.
- Model rejection rule: If there is a negative clause whose antecedent $A\sigma$ is satisfied in a model M, reject the model M.

The task of model generation is to try to construct a model for a given set of clauses, starting with a null set as a model candidate. If the clause set is satisfiable, a model should be found. This method can also be used to prove that the clause set is unsatisfiable, by exploring every possible model candidate to ensure that no model exists for the clause set.

Figure 1 shows the proof tree for a sample problem. We start with an empty model candidate, $M_0 = \phi$. M_0 is first expanded into two cases, $M_1 = \{p(a)\}$ and $M_2 = \{q(b)\}$, by applying the model extension rule to $C6$. Then M_1 is expanded by $C5$ into two cases: $M_3 = \{p(a), q(a)\}$ and $M_4 = \{p(a), r(a)\}$. M_3 is further extended by $C3$ to $M_5 = \{p(a), q(a), s(f(a))\}$. With M_5, however, the model rejection rule is applicable to $C2$; thus M_5 is rejected and marked as closed. On the other hand, M_4 is extended by $C4$ to $M_6 = \{p(a), r(a), s(a)\}$, which is rejected by $C1$. Similarly, the remaining model candidate M_2 is extended by $C3$ to $M_7 = \{q(b), s(f(b))\}$, which is rejected by $C2$. Now that there is no way to construct a model candidate, we can conclude that the clause set S1 is unsatisfiable.

メンテーションを考える。順次、方式Ⅱや方式Ⅲのインプリメンテーションへと拡張していく。

次に、GRAPE の初期知識ベース獲得機能の基本構想を述べる。GRAPE 研究開発の目的は、知識システム開発の上流工程（問題定式化、知識表現選択）支援とグループ知識獲得支援の本質を明らかにすることである。前者については、知識ベースのラピッドプロトタイピング支援機構の構築を通じて、主観的な知識ベース構造を決定することに特徴がある。後者については、グループウェアに対する先行研究であり、KJ法〔Kawakita 87 〕を中心とする協調問題解決支援環境の構築に特徴がある。

GRAPE の精神はKJ法で、その技術としては『Colab+AQUINAS+α』〔Stefik 87,Boose 87 〕を活用することをモットーする。αとしては、諸々のシステム分析技法を活用する。GRAPE は初期知識ベース獲得機能と計画問題支援機能からなる。初期知識ベース獲得機能は、与えられた問題のマクロ決定を支援し、与問題に対する主観的評価のガイドライン（枠組み）を決定する。これに対して、計画問題支援機能は、与えられた問題のミクロ決定を支援し、既存知識を再利用しつつ、前記ガイドラインを満足するプランを生成していく。本論文では、GRAPE の初期知識ベース獲得機能をグループウェア的側面から明らかにしていく。GRAPE の初期知識ベース獲得機能は、図２に示されているように、仮説抽出、仮説構造化、属性抽出、属性構造化、グリッド分析・統合を経て、主観的概念木の生成を支援する。それぞれの機能の詳細は、次章以降で述べる。

3．初期知識ベース獲得機能のシステムフロー

GRAPE 初期知識ベース獲得機能は、いわゆる前処理、基本処理、後処理に相当する問題決定部、基本（グループ）決定部、決定調整（コンサルテーション）部の三つの処理部から成る。標準的には、基本決定部で、その問題に関与する全てのグループメンバが参加し、決定調整部でグループメンバと専門家とのコンサルテーション・プロセスを通じて、グループ決定の結果を調整していく。問題の種類によっては、基本決定部で専門家の知見を整理し、決定調整部でグループメンバとのコンサルテーションを行う。

(1) 問題決定部

GRAPE ユーザは最初、解くべき問題を事前に決定しておく。解くべき問題の決定そのものを支援するグループウェア構築も、極めて魅力的な研究開発課題であるが、KJ法そのものを支援するツールを作ることと同様、現状では極めて困難なのでGRAPE 向きの問題が所与であると仮定する。我々の目標は『コンピュータの得意なことはコンピュータに、人間の得意なことは人間に！』という設計方針のもと、コンピュータと人間との協調型アーキテクチャを明らかにすることにある。また随所で、ユーザのコンピュータ入力がみられるが、誰でも気楽に参加できるように、マウスのクリックによるメニュー選択とキーワード入力を基本とする。

(2) 基本（グループ）決定部

GRAPE の基本決定部は、以下に述べるプロセスから成る。

①最初、その問題のグループデシジョンに参加するグループメンバを決定する。ここでは各メンバの役割は対等であり、その問題の解決のための計画に積極的に参加していることを強調し、彼らのことを参画者と呼ぶ。参画者の人数は、２～10人位を想定しているが、

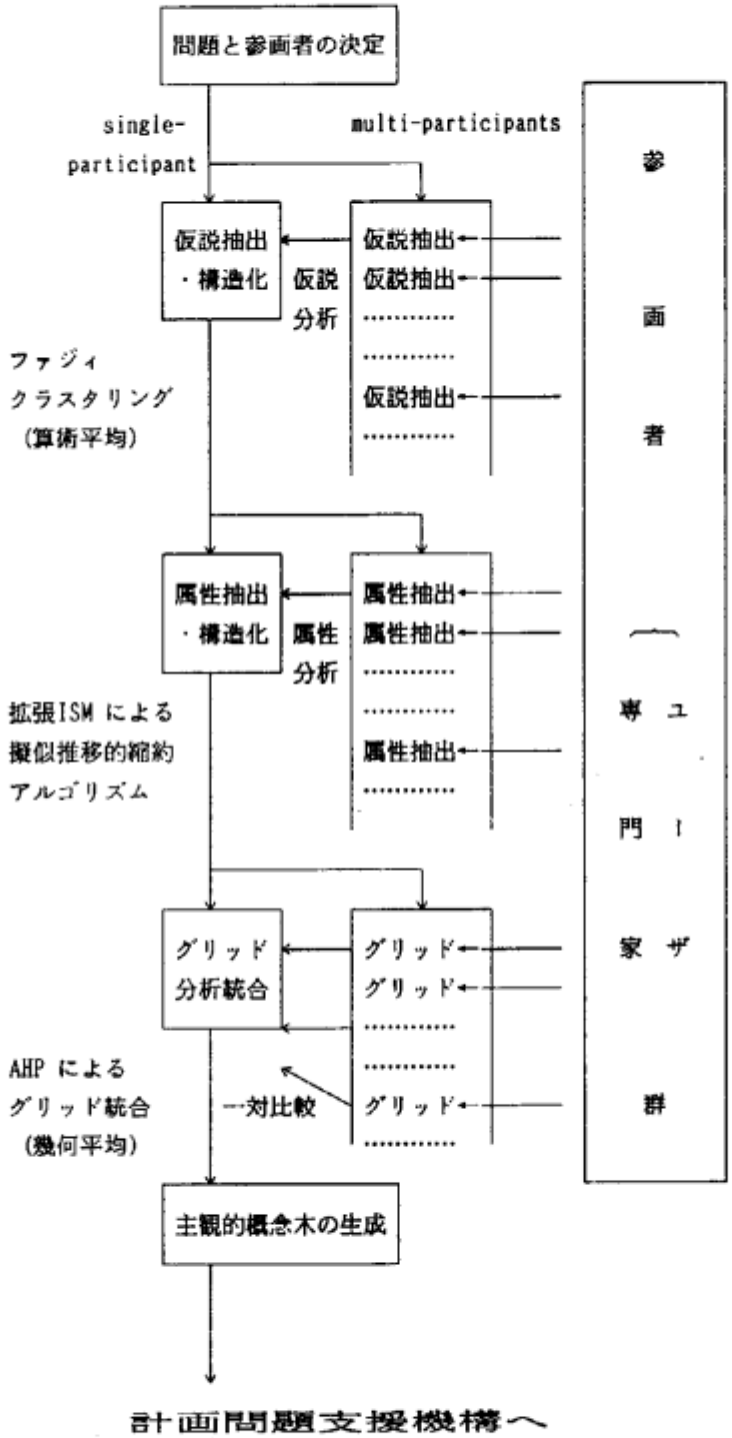

図2　GRAPE 初期知識ベース獲得機能のシステムフロー

4～6人位が理想的であろう。参画者の人数がやたら大きいと、会議で合意をとるのが困難と同じ現象を呈する。このプロセスは電子会議を知的かつ生産的に進めるにはどうするかの設計であり、基本的にはグループ問題解決の要領で決定していけばよい。

②上記参画者の中から、コーディネーターとなる司会役を決定する。当面、司会役の役割は通信の同期をとることであり、ネゴシエーションをガイドすることではない。従って、司会役にコーディネーション特有の能力を要求するものではないが、将来的には生産的なデベートのガイド能力を要求することになるであろう。

③司会役からみて参画者が通信し易いマシン接続を決定し、それに伴いマシン間がトークできるように立ち上げる。ここに、マシンの接続方式には、階層結合とネットワーク結合が考えられるが、司会役が通信の同期をとる関係上、ここでは階層結合をとる。PSI のプロセス立上げが遅いため、ネットワーク結合は実用的でない。参考までに、ネットワーク結合の場合、プロセッサの個数の二乗のオーダーに比例したネットワーク設定時間が、階層結合の場合、プロセッサの個数のオーダーに比例したネットワーク設定時間がかかる。ただし、方式Ⅲの場合はネットワーク結合の方が現実的である。

④問題の解の候補集合を仮説（あるいは対象）と呼ぶ。全ての参画者が考えられる仮説を入力していく。入力された情報はWYSIWIS(What You See Is What I See) 原則〔Stefik 87〕で、全ての参画者がウィンドーで見れるようにする。またサブグループ間コミュニケーションの重要性に鑑み、サブウィンドーを共用するコミュニケーションも許容することにする。理想をいえば、Colab のように、大型の共通スクリーンBoardnoterの欲しい所である。

⑤仮説の数がある値(7±2)以上の場合、仮説間のクラスター分析を行い、仮説間構造を抽出していく。クラスター分析としては、（KJ法を含む）マニュアル法、ファジィクラスターリング、林の数量化理論等、種々のものが考えられるが、GRAPE ではデフォールトとして、ファジィクラスターリング〔Mizumoto 88〕を標準装備している。仮説の総数としては、一人当たり10個以内、最大100 個程度の仮説が、快適な対話環境を保証する上で、妥当なリミットと考える。

ただし、ファジィクラスターリングの場合、グループ全体の値は算術平均を採用する。ファジィ行列の整合性をとるためには、ある種の制約条件を満足しなければいけない。そこで、グループ入力の整合性を取るには、付録で述べる幾つかの工夫を要する。

⑥上記の構造化された仮説のトップレベルから順次、それらを差別化する対となる属性を全ての参画者が入力していく。入力形式は((属性，対属性),統合属性）というタップルで与える。対属性入力の理由は、後述の一対比較を円滑に進めるためである。入力された情報はWYSIWIS 原則で、ウィンドー表示されるが、階層構造の各レベルを区分する属性が抽出されていく。

⑦属性の数がある値(7±2)以上の場合、属性間のクラスター分析を行い、属性間構造を抽出していく。クラスター分析としては、（KJ法を含む）マニュアル法、拡張ISM(Interpretive Structural Modelling)〔Kunifuji 79〕、属性クラスタリング等、種々のものが考えられる。GRAPE ではデフォールトとして、拡張ISM を標準装備している。属性の総数としては、階層の各レベル当たり、最大20～30程度の属性が、付与されると考える。

ここでの処理の本質は属性間の従属性・独立性の分析にあるが、拡張ISM をデフォール

トとする理由は、それが基本的に属性間の従属性分析の一手法を提供するからである。お互いに併存到達可能な属性集合同志を等価な属性集合とみなし、縮約した骨格構造を抽出するプロセスを経て、属性間構造が抽出されたと考える。

⑧AHP 〔Saaty 80,Tone 86,Sawai 89 〕利用を想定し、仮説ー属性間の必要な部分構造のグリッド値〔Fransella 77,Hart 86〕を入力していく。仮説ー属性構造間の一対比較に関しては、グループ全体の値は幾何平均を採用する。グリッド値の入力も、全体の整合性を保持するためにある種の制約を満足しなければいけない。グループ全体の整合性を保つために、不完全一対比較行列〔Harker 87,Takeda 89〕を用いた不完全情報の補完法を利用する。

(3) 決定調整（コンサルテーション）部

①グループ決定の結果を修正するコンサルタント役の専門家に参加してもらう。その問題の専門家であるコンサルタントの持ってる情報は侮りがたいものがある。専門家の意見に謙虚に耳を傾け、上記グループ決定の過程では考慮し損ねた視点や指摘された現実的制約を用いて、実行不可能な候補を消去するのに利用できる。

②専門家と上述の参画者（場合によっては、司会役のみ）による基本処理の繰り返しを行い、主観的概念木の修正を行う。それにより、属性の見直し、仮説の追加、評価値の修正が起こり、専門家の意見を尊重しつつ、参画者全員の意見を反映した非単調な変化が起こる。どういうメンバが参加するかについては、プラグマティズムの精神で決定していく。例えば、参画者が少数の場合、専門家と参画者全員が参加するのが望ましいが、参画者が多数の場合、専門家と司会役のみが参加するのが現実的であろう。

上記フローに関して配慮された幾つかの検討事項を指摘しておく。属性の尺度が、名義尺度・順序尺度・距離尺度・比尺度のどれであるかによって、グリッド統合の方法が異なってくる。仮説間の親近度は距離尺度、属性間の一対比較値は比尺度と見なすのが自然なので、前者は算術平均、後者は幾何平均を採用した。これら尺度間の変換を行い、尺度統一をするのも一つのアプローチである。実際、順序尺度を距離尺度に変換する展開法が知られているが、一般にこの種の変換は極めて難しい。万一、尺度統一できたとしても、既存の多目的決定理論との違いが無くなる危険性があるので、当面、尺度統一は考えない。属性によっては、主観値を与えるのが適切なものと、客観値を与えるのが適切なものと、２種類ある。これらの属性の区分をし、初期知識ベース獲得機能では前者を、計画問題支援機能では後者を利用するのが素直である。分類や決定に利用できる客観的データベースが存在する場合、たとえばQuinlan のID3 アルゴリズムを利用するのが自然である。主観的評価と客観的評価との統合は極めて大切な課題であるが、当面は別個に出来た概念木同志の整合性をチェックするのに用いる。

4．構造化のための諸手法

本節では、GRAPE でグループの初期知識を獲得しながら、それを知識ベース向きに構造化する過程で使用される諸々のシステム・モデリング技法のうち、特にグループ知識獲得特有の全体の整合性を保持するための技法を提案する。

(1) 仮説構造化とファジィクラスタリング

仮説間の構造解析をする際、類似度という距離尺度を手掛かりに、ファジィクラスタリ

Table 2: Performace of Tableaux for PTL(msec)

No.	Theorem	I	II	III	ALS
1	$\Diamond\Box a \supset \Box\Diamond\Box a$	900	2790	2770	6383
2	$\neg\Diamond a \equiv \Box\neg a$	20	19	19	6617
3	$\Diamond a \vee \Diamond\neg a$	10	<1	10	683
4	$a \supset \Diamond a$	<1	<1	10	500
5	$\bigcirc a \supset \Diamond a$	19	9	10	1783
6	$\Box a \equiv \Box\Box a$	19	30	39	3050
7	$\Diamond a \equiv \Diamond\Diamond a$	49	30	49	3367
8	$\Diamond\neg a = \neg\Box a$	10	10	20	1583
9	$\Box(a \supset b) \supset (\Diamond a \supset \Diamond b)$	39	30	30	3067
10	$\Box(a \wedge b) \equiv (\Box a) \wedge (\Box b)$	50	50	60	10250
11	$\Diamond(a \vee b) \equiv (\Diamond a) \vee (\Diamond b)$	99	70	70	9717
12	$(\Box a \vee \Box b) \supset \Box(a \vee b)$	29	29	30	3250
13	$\Diamond(a \wedge b) \supset (\Diamond a \wedge \Diamond b)$	39	39	59	3317
14	$(\Box a \wedge \Diamond b) \supset \Diamond(a \wedge b)$	30	30	30	2967
15	$\bigcirc(a \wedge b) \equiv (\bigcirc a \wedge \bigcirc b)$	30	30	30	8283
16	$\bigcirc(a \vee b) = (\bigcirc a \vee \bigcirc b)$	40	30	30	8317
17	$\bigcirc(a \supset b) \equiv (\bigcirc a \supset \bigcirc b)$	40	30	30	8367
18	$\bigcirc(a \equiv b) \equiv (\bigcirc a \equiv \bigcirc b)$	49	50	60	19033
19	$\bigcirc\Box a \equiv \Box\bigcirc a$	329	210	89	8800
20	$\bigcirc\Diamond a \equiv \Diamond\bigcirc a$	680	259	130	9000
21	$\Box\Diamond\Box a \equiv \Diamond\Box a$	899	2849	2769	46983
22	$\Diamond\Box\Diamond a \equiv \Box\Diamond a$	259	5179	2060	44467
23	$\Box a = (a \wedge \bigcirc\Box a)$	59	129	140	30867
24	$\Diamond a \equiv (a \vee \bigcirc\Diamond a)$	170	139	160	27167
25	$(a \wedge \Diamond\neg a) \supset \Diamond(a \wedge \bigcirc\neg a)$	239	180	199	117167
26	$(\neg a)\mathbf{U}a \equiv \Diamond a$	30	30	39	6700
27	$\Box a \wedge \Diamond b \supset a\mathbf{U}b$	30	40	60	6350
28	$(a\mathbf{U}b)\mathbf{U}b \supset a\mathbf{U}b$	50	30	30	14833
29	$a\mathbf{U}(a\mathbf{U}b) \supset a\mathbf{U}b$	319	199	250	25133
30	$\Box a \wedge (b\mathbf{U}c) \supset (a \wedge b)\mathbf{U}(a \wedge c)$	409	150	229	34983
31	$(a \wedge b)\mathbf{U}c \equiv (a\mathbf{U}c) \wedge (b\mathbf{U}c)$	189	169	189	27033
32	$a\mathbf{U}(b \vee c) \equiv (a\mathbf{U}b) \vee (a\mathbf{U}c)$	3100	1390	1150	236183
33	$(\Diamond a \vee \Diamond b) \supset (((\neg a)\mathbf{U}b) \vee ((\neg b)\mathbf{U}a))$	189	99	109	119367
34	$a\mathbf{U}(b \wedge c) \supset (a\mathbf{U}b) \wedge (a\mathbf{U}c)$	329	240	400	32417
35	$(a\mathbf{U}c) \vee (b\mathbf{U}c) \supset (a \vee b)\mathbf{U}c$	69	89	99	11817
36	$(a \supset b)\mathbf{U}c \supset (a\mathbf{U}c \supset b\mathbf{U}c)$	90	60	70	10000
37	$(a\mathbf{U}b) \wedge ((\neg b)\mathbf{U}c) \supset a\mathbf{U}c$	350	120	219	33583
38	$(a\mathbf{U}b)\mathbf{U}c \supset (a \vee b)\mathbf{U}c$	229	109	460	20267
39	$a\mathbf{U}(b\mathbf{U}c) \supset (a \vee b)\mathbf{U}c$	6809	4561	4400	33367
	Total	16300	19509	16607	997018

7 Conclusion

A modal propositional tableaux is implemented by writing the tableaux rules as a set of input clauses for MGTP. This tableaux system is concise in the sense that only clauses are related to the tableaux rules. We can obtain a modal propositional tableaux for K by writing nothing more than a 12-line program if the input clauses for MGTP are regarded as being a programming language.

This is a kind of meta-programming technique. In general, the speed of the program, made using a meta-programming technique, is expected to be slow because of its interpretation overhead. However, the performance is actually relatively good. This is due to the interpretation overhead being quite small in MGTP.

To make a tableaux prover, examination of the following points is important.

(1) How to represent a target mathematical domain.

(2) How to increase the inference speed.

In our method, (1) results in how to express the properties of the mathematical domain in an MGTP clause, while (2) results in how to attain MGTP efficiency. Isolating (1) from (2) makes the implementation of the tableaux prover brief and compact.

There are several areas to which modal logic can be applied, such as specification description, verification, knowledge representation. This paper shows that MGTP could be a useful tool for implementing a variety of modal propositional tableaux systems to cover these areas, although the proving mechanism for MGTP is quite simple.

In the near future, we intend to build several modal tableaux systems based on the method given in this paper to show its effectiveness.

References

[1] M. Fitting: Proof Methods for Modal and Intuitionistic Logics, D.Reidel Publishing Co., Dordrecht 1983.

[2] M. Fitting: First-Order Modal Tableaux. *J. Automated Reasoning*, 4(2), 1988.

[3] H. Fujita and R. Hasegawa: A Model Generation Theorem Prover in KL1 Using Ramified-Stack Algorithm. In *Proc. 8th ICLP*, pages 535–548, 1991. also in ICOT TR-606 1990.

[4] S.A. Kripke: Semantical Analysis of Modal Logic I, Normal Propositional Calculi. *Zeitschrift für mathemtische Logik und Grundlagen der Mathematik* 9, 67–96, 1963.

[5] R.E. Ladner: The Computational Complexity of Provability in Systems of Modal Propositional Logic. *SIAM J. Comput.*, 6(3):467–480, 1977.

[6] D.A. Plaisted and S-J. Lee: Inference by Clause Linking. 1990.

[7] R.M. Smullyan: *First-Order Logic*, volume 43 of *Ergebnisse der Mathematik*. Springer-Verlag, Berlin, 1968.

[8] N. Yonezaki. Modal Logic. *J. IPS Japan*, 30(6), 1989. (in Japanese)