TR-0763

MGTP上の仮説推論システム

井上　克巳、太田　好彦
長谷川　隆三、中島　誠 (JIPDEC)

April, 1992

© 1992, ICOT

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

MGTP 上の仮説推論システム

Hypothetical Reasoning Systems on the MGTP

井上 克巳　　太田 好彦　　長谷川 隆三

(財) 新世代コンピュータ技術開発機構

108 東京都港区三田 1-4-28 三田国際ビル 21F

inoue@icot.or.jp　ohta@icot.or.jp　hasegawa@icot.or.jp

中島 誠

(財) 日本情報処理開発協会

108 東京都港区三田 1-4-28 三田国際ビル ANNEX

nakasima@icot.or.jp

1992 年 3 月 31 日

改訂: 1992 年 8 月 6 日

ICOT Technical Report　TR-763

概要

仮説推論, あるいはアブダクションを高速化するために並列仮説推論システムを疎結合型並列推論マシン PIM/m 上に開発し評価した. 並列仮説推論システムとして, 最終的に 5 つの方式を実現し評価したが, これらはいずれも, ボトムアップ型の推論エンジン (並列定理証明器 MGTP のモデル生成・拡張機能を利用) に, 仮説を立てたりその無矛盾性を検査する機能を導入したシステムである. まず, MGTP と ATMS を結合する方式では, MGTP と ATMS の結合により同期を取る必要があり待ちが生じるため, プロセッサ台数を増やしたときの並列性には限界がある. 次に, ATMS を用いずに別の MGTP を用いて無矛盾性検査を行わせたところ, この検査が並列に実行できるために並列性を向上させることができた. しかしながら, この方式でもやはり 2 種の MGTP が通信し合うために, 待ちの問題は残されており性能向上には限界があると考えられる. そこで, 1 台の MGTP のみを用いてすべての機能を実現する方式を 3 種試みた. これらの中では, 必要なときにのみモデルの分岐を行う方式 (Skip 方式と呼ぶ) が, 並列化などを加味すると最も有望である, という結論を開発・実験の結果得た.

目次

1 はじめに **3**

2 アブダクション **3**

3 MGTP **4**

4 MGTP による仮説推論の実現 **5**
- 4.1 MGTP を用いた仮説推論システムの構成 7
- 4.2 入力変換規則 8
- 4.3 不要なモデル候補の枝刈り 11
- 4.4 トップダウン情報の組み込み 12

5 評価 **14**
- 5.1 例題および測定結果 14
- 5.2 考察 14
- 5.3 枝刈りルールの効果 17

6 まとめ **19**

A 変換例 **21**
- A.1 *Normal translation* 22
- A.2 *Simple translation* 25
- A.3 *Left-to-Right translation* 29

1 はじめに

今後の情報処理システムでは大規模知識ベースを扱うことが必要とされている. 従来の知識ベースシステムでは, 想定された使用条件と少しでも異なる環境で用いようとすると作動しないという弱点があり, また知識ベースの規模が比較的小さいため, 必要となる知識を前持って列挙しておくことや, 情報が不足しているときに付け加えることが容易に可能であった. これに対して, 大規模知識ベースでは, 様々な用途に応じて予め必要となるすべての情報を完全に記述しておくことは原理的に不可能であり, 知識ベースを修正する場合でもその管理に要するコストは莫大なものになる. したがって, その実現のためには, 知識ベースが不完全であっても何らかの結論が引き出せる機能や, 例外を含むような常識的知識が容易に扱える機能の実現が不可欠である. さらにこれらの機能の実現は, 大規模知識ベースを扱う際の問題にとどまるわけではなく, 知的対話機能を含む知的インタフェース・知的データベースなどの高度な問題解決機能の実現にとっても本質的な課題である.

不完全な知識や常識的知識に基づく問題解決の実現にとって, 一つの鍵となる技術として仮説推論の技術がある. 仮説推論では, 情報が足りないときに仮説を補うこと (アブダクション) や, 一時的に成り立つ知識を取り敢えず正しいと仮定すること (デフォルト推論) により, 不完全な知識を扱うことができる. ところが, 従来の演繹のみに基づく推論システムと比べると, 仮説生成のコストや導入した仮説の無矛盾性の検査のコストがかかり, しかもそれらの計算量が非常に大きい点が実現上克服すべき課題となっている. したがって高速な仮説推論システムの開発が望まれている.

高速な仮説推論を実現するための一つの手段は, 仮説推論を並列化することである. 本稿では仮説推論の並列化のための様々な手法を示し評価する. とくに, 並列モデル生成型定理証明器 MGTP [4] を用いた仮説推論システムの構築手法を提案し, 疎結合型並列計算機 PIM/m 上に実現し評価を行った.

MGTP は, モデル生成法に基づく一階述語論理のための並列定理証明器であり, KL1 [21] により実現されている. MGTP は入力された論理式集合のモデルをボトムアップに生成することを試みる. よって MGTP では, モデルの生成に成功するか否かによって論理式集合の充足不可能性を調べること, および充足可能な論理式集合のすべてのモデルを求めることができる.

以下, 2章で仮説推論の概略, 3章で MGTP の概略を述べ, 4章で開発した 5 種類の仮説推論システムの実現方法を示し, 5章で 2 つの問題を例に行なった計測結果をもとに評価する. また, 付録 Aで例題とその MGTP ルールへの変換例を示す.

2 アブダクション

仮説推論においては, 与えられた知識の集合と矛盾しない限り仮説を導入し, 結論の集合がそれにより拡張される. 仮説推論のうちとくに, ある (観測) 事実を説明することのできる仮説集合を計算することを**アブダクション** (*abduction*) と呼ぶ. この説明という概念は, 診断, 合成, 設計および自然言語理解のような種々の AI 問題の基本的な概念として有用である. これまでに, アブダクションに関する多くの研究が行われてきている ([9] を参照のこと).

以下で考察するアブダクションの定義は [19] と同様である. いま, Σ を論理式の集合, Γ をリテラルの集合, G を閉論理式とする. E を Γ の基礎例の集合 として, もし,

1. $\Sigma \cup E \models G$,
2. $\Sigma \cup E$ は無矛盾,

であるならば, E は (Σ, Γ) からの G の**説明** (*explanation*) であるという. このとき, G は E によって**支持されている** (あるいは G は E に**依存する**) と言うこともある. また, 説明 E のいかなる真部分集合も上記 2 条件を満たさないとき, E を (Σ, Γ) からの G の**極小の** (*minimal*) 説明であるという.

(Σ,Γ) からの G の説明を計算することは，Σ と矛盾しない Γ からの仮説集合を導入することによって G を証明することとみなせる．これはまた，アブダクションが結論発見問題としても形式化できる [8] ことを示す．つまり，あるリテラルに関してそれを証明する代わりに，そのリテラルを証明できたと仮定 (skip) することを許容する．これにより，演繹の最後で空節が導かれるかわりに，それらの skip されたリテラルだけが含まれている新しい定理が導かれる．このようにアブダクションはとくに，トップダウン - 後向き定理証明手続きによる演繹の拡張によって実現できる．例えば，Theorist [19] や SOL- 導出 [8] は，Model Elimination 定理証明手続き [12] の拡張である．

一方アブダクションは，ボトムアップ - 前向き推論手続きによっても実現できる．実際，前向き推論エンジンと ATMS [3] とから構成された APRICOT/0 [15, 16] と呼ばれるアブダクションシステムが開発されている．ここで ATMS は，同じサブゴールを繰り返し証明したり同一の結果を導く異なった仮説集合間の重複した証明とを避けるために，それらの推論結果を導いた根拠を保存する機構として用いられている．

アブダクションのための，これら 2 つの推論方法は互いに相補的な利点と欠点を持っている．トップダウン推論は目標主導型であり，与えられたゴールの証明に無関係なサブゴールの証明は行わないが，先のような重複した計算を行う可能性がある．逆に，ボトムアップ推論では重複計算は行わないが，ゴールの説明と無関係な推論を行ってしまう可能性がある．これらは，ボトムアップ推論によってトップダウン推論を模擬する方法や，逆にトップダウン推論にキャッシュ機構を組み込んで統合するような方法を用いることで，各々の欠点を克服できることを示唆している．例えば，マジックセット法 [1] の一般化である，*"Upside-Down Meta-Interpretation"* 法 [2] は，[20] によりアブダクションに適用され，[17] により無矛盾性検査と統合されるように拡張されている．

以下 4 章で述べる仮説推論システムでは，ボトムアップの手続きとしてアブダクションが実現されている．また，トップダウン情報を反映させるため upside-down meta-interpretation を合わせて適用している．

3 MGTP

ここでは，開発した仮説推論システムのベースとなる MGTP [4, 10] の概略について述べる．

MGTP に対する入力は，一階述語論理式を表す節 (あるいは**ルール**) の集合である．ここでルールは以下の形で与えられる:

$$A_1,\ldots,A_n \rightarrow C_{1,1},\ldots,C_{1,k_1} \mid \ldots \mid C_{m,1},\ldots,C_{m,k_m}.$$

ここに，A_i $(1 \leq i \leq n;\ n \geq 0)$ および C_{k_j} $(k_j \geq 1;\ 1 \leq j \leq m)$ はアトムである．また $\rightarrow$ の左側をルールの**前件**，右側を**後件**という．$m = 0$ のときの $A_1,\ldots,A_n \rightarrow$ を**負節**と呼び，$A_1 \wedge \ldots \wedge A_n$ が成立すれば矛盾することを示す．

以下では基礎アトムの集合を解釈，あるいは**モデル候補** (*model candidate*) と呼ぶ．MGTP には以下の 2 つの操作がある．

- **モデル候補拡張**: あるルール

$$A_1,\ldots,A_n \rightarrow C_{1,1},\ldots,C_{1,k_1} \mid \ldots \mid C_{m,1},\ldots,C_{m,k_m}$$

において，代入 δ のもとに $A_1,\ldots,A_n$ があるモデル候補 M で充足される $(A_1\delta,\ldots,A_n\delta \in M)$ とき，いかなる $j = 1,\ldots,m$ についても，$C_{j,1}\delta,\ldots,C_{j,k_j}\delta \in M$ が成り立たなければ，$C_{j,1}\delta,\ldots,C_{j,k_j}\delta$ を M に加えて拡張する．

- **モデル候補棄却**: ある負節

$$A_1,\ldots,A_n \rightarrow$$

において，代入 δ のもとに $A_1,\ldots,A_n$ があるモデル候補 M で充足されるとき，M を削除する．

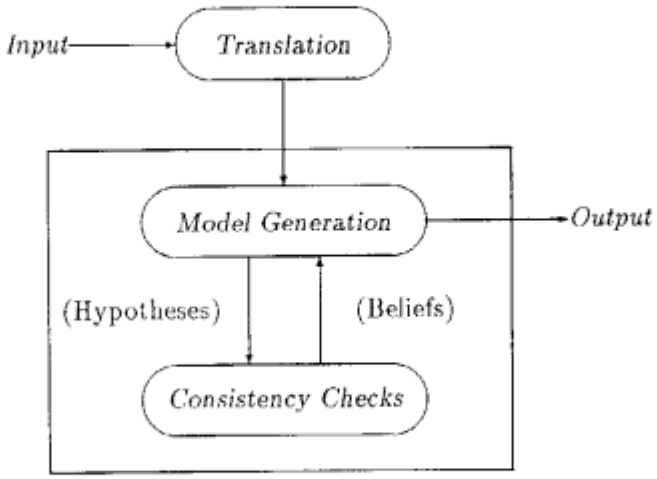

図 1: MGTP を用いた仮説推論システム

ここで $A_1\delta, \ldots, A_n\delta$ を得る過程をモデル候補と前件との**連言照合** (*conjunctive matching*) という. MGTP は, 初期モデル候補集合として $\{\emptyset\}$ から始め, 以上のいずれの操作も適用することができなくなった時点で, 最終的なモデル候補の集合 $\mathcal{M}$ を出力する. このとき, すべての $M \in \mathcal{M}$ について, M はすべてのルールを満足しているため, 与えられた論理式集合の**モデル**となる. もし, $\mathcal{M} = \emptyset$ であれば, すべてのモデル候補が棄却されておりモデルが存在しないため, その論理式集合が矛盾している (あるいは充足不可能) であることを示す. また, $\mathcal{M} \neq \emptyset$ であれば, $\mathcal{M}$ は与えられた節集合のすべての**極小モデル** (*minimal models*) を含んでいる [10].

MGTP ではモデル候補として基礎アトムのみからなる集合を考えるために, すべての入力ルールが**値域制限** (*range-restrictedness*) の条件を満たしていることが必要である. ここで, あるルールが range-restricted であるとは, 後件に現れる変数はすべて前件に現れていることを意味する. この条件により, 前件に現れる変数に基礎項を代入することによって後件の変数が具体化されるため, いかなるモデル候補拡張においても基礎アトムのみが加えられる. さらに, 値域制限の条件を満たしていれば, 入力ルール中の変数は KL1 変数によって表すことができるたことからユニフィケーションが不要となり, 連言照合に対して高速な KL1 のガードユニフィケーションを用いることができる. また, ルールの後件が選言の場合, そこでモデル候補の拡張に場合分け (**モデル候補分岐**) が必要となる. これら拡張されたモデル候補はすべて変数を含まないから, モデル候補間の共有変数の問題がなくなり, 各々まったく独立にモデル候補を拡張していることができ, 自然な OR 並列実行が可能となる. さらに, 連言照合の履歴を保持する Ramified-Stack Algorithm [4] を採り入れ, 冗長な連言照合を回避することで高速化を計っている.

4 MGTP による仮説推論の実現

本章では, MGTP を用いて仮説推論システムを実現するために, 5 種類の構築方法を提案する. これらの仮説推論システムはいずれも図 1 の概念図によって表される.

ここでは, 仮説推論システムに対する入力を, 仮説を含むホーン節の集合とする. ここで, **ホーン節**は次のいずれかの形式の節である:

$$A_1, \ldots, A_n \rightarrow C \tag{1}$$

$$A_1, \ldots, A_n \rightarrow \tag{2}$$

ここに, A_i $(i = 1, \ldots, n;\ n \geq 0)$ および C はアトムであり, C をここでは結論と呼ぶ. また, (1) を確定節と呼び, (2) は負節である. さて, $A_1, \ldots, A_n$ がすべて成立した場合, "結論 C が成り立つものと仮定できる" ことをメタ述語 **assume** を用いて,

$$A_1, \ldots, A_n \rightarrow \mathbf{assume}\ C \tag{3}$$

のように表す. いま, MGTP 上の仮説推論システムへの入力節の集合を P と書くことにする. このとき, P は次のように細分化することができる:

$$\begin{aligned}
P_1 &: \quad \rightarrow C \\
P_2 &: \quad A_1, \ldots, A_n \rightarrow C \\
P_3 &: \quad A_1, \ldots, A_n \rightarrow \\
P_4 &: \quad \rightarrow \mathbf{assume}\ C \\
P_5 &: \quad A_1, \ldots, A_n \rightarrow \mathbf{assume}\ C
\end{aligned}$$

以後, 上記 P_2, P_3, P_5 で表される節については, $n \geq 1$ であるものとする.

なお, 入力節の集合 P と 2章で表されるアブダクションの枠組との対応は次の通りである. いま, P_4 および P_5 に含まれ (3) 式 $(n \geq 0)$ で表される各節に対して, 結論のメタ述語 **assume** 内に現れる C に対応する新しい述語記号 (**仮定可能述語**; *assumable*, あるいは *abducible*) を導入し, これを C^* とするとき,

$$A_1, \ldots, A_n, C^* \rightarrow C \tag{4}$$

なる確定節を考える. すべての仮定可能述語の集合を Γ_P とし, (4) で表されるすべての節を P'_{45} とし,

$$\Sigma_P = P_1 \cup P_2 \cup P_3 \cup P'_{45}$$

と書くとき, 二つ組 (Σ_P, Γ_P) はアブダクションの枠組となる. もし, ある観測 (**ゴール**) G が仮説推論システムに与えられると, アブダクションによって (Σ_P, Γ_P) からの G の (極小の) 説明をすべて求めることが, 仮説推論システムの目的となる.

また, 仮説推論システムでは, 与えられた入力節集合 P を, 図 1 に示すように, MGTP へのルール形式に変換する. ここで, 5 種類のそれぞれのシステムにおいては, それぞれのシステムに対応するようにルールの変換を行う. さらに, 仮説推論システムではしばしば説明されるべきゴールが指定されるが, このとき, MGTP のボトムアップ・モデル計算において, ゴールに無関係なアトムの説明まで求めてしまうことを避けるために, upside-down meta-interpretation [2, 20] (以下 *UD-meta*) を用いて効率向上を目指すための変換も行うことができる.

変換されたルール集合は, 図 1 において枠で囲まれた部分に渡される. ここでは, (a) *Model Generation* (**モデル生成**) と (b) *Consistency Checks* (**無矛盾性の検査**) の二つの機能が仮説推論システムに必要となる. これらの二つの機能はそれぞれ, アブダクションにおける説明 E の定義の二条件: (a) $\Sigma_P \cup E \models G$, (b) $\Sigma_P \cup E$ は無矛盾, に対応している. ここで, (b) の無矛盾性検査が必要になるのは明らかである. (a) のモデル生成がゴール G の説明 E を求める条件と対応する理由は, Σ_P はホーン節の集合であるため, それに Γ_P のいかなる部分集合 E を加えて拡張したプログラムについても, 極小モデルは各 E に対して必ず一つだけ存在するため, そのような極小モデルの中で G を満たすモデルを発見する問題に帰着できるからである. この性質により, 仮説推論システムではモデル生成に MGTP を用いることができる. MGTP は変換されたルール集合の極小モデルをすべて生成し出力する.

以下では, 5 種類の仮説推論システムの構成および特徴, また各々に合わせた入力の変換規則を示す.

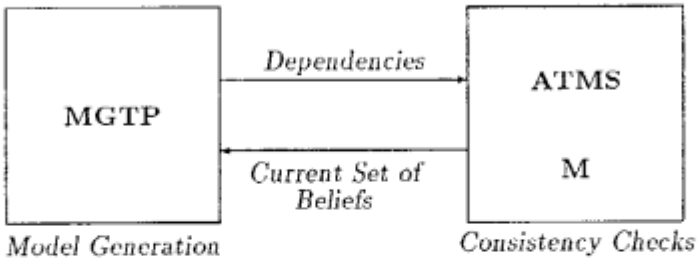

図 2: MGTP+ATMS 方式

4.1 MGTP を用いた仮説推論システムの構成

以下に述べる 5 つの仮説推論システムは, 大別して 2 種類の方式に分類できる. このうち, 最初の分類には, (1) **MGTP+ATMS 方式**, (2) **MGTP 連結方式**, の二つが, また二番目の分類には, (3) **Nogood 仮定方式**, (4) **全モデル生成方式**, (5) **Skip 方式**, の三つが存在する. 以下, これら 5 つの構成と特徴について述べる.

最初の 2 つは, 図 1における *Consistency Checks* (無矛盾性の検査) を *Model Generation* (モデル生成) を行う MGTP とは別のシステム (ATMS あるいはもう 1 種類の MGTP) で行う方式である. モデル生成を行う方の MGTP は推論エンジンとして使われ, 単一のモデル候補 M を参照しながら, それを拡張していく.

残りの 3 つのシステムでは, モデル生成および無矛盾性の検査をともに 1 種類の MGTP によって実現している. 実際, 3章で述べたように, MGTP はモデル生成と無矛盾性検査との両方の機能を備えており, これを単に推論エンジンとして使用するだけではなく, 生成検査機構として使用することができる. すなわち, 無矛盾性の検査を別システムとはせずに, MGTP が有するモデル候補分岐およびモデル候補棄却の機能を用いることで, モデル候補の生成検査問題として扱い, モデル生成とともに単一の MGTP で実現している. このような方式を用いることにより, 大域的な信念の集合 M を保存するかわりに, 複数のモデル候補を分散メモリ中に各々保存すればよいようにできる.

1. MGTP+ATMS 方式

この方式は, APRICOT/0 [15, 16] を並列化したものであり, 仮説集合の無矛盾性のチェックと管理のために ATMS [3] を用いている. MGTP は, 前向き推論エンジンとして用いられ, ATMS は, 現時点で信じられているデータの集合 M を保存するために用いられる. ここで, M はそれぞれのアトムの極小の説明の集合によってラベル付けされた基礎アトムの集合である. この M をモデル候補として, MGTP はモデル拡張を行う. ATMS には KL1 で開発した並列 ATMS [14] を用いている.

2. MGTP 連結方式

MGTP 連結方式は, Stickel [20] によって提案されたボトムアップベースのアブダクションを並列化した二つの版のうちの一つである. Stickel 方式のもう一つの並列化は, 次に示す Nogood 仮定方式である. ただし, [20] では得られた説明の無矛盾性検査を全く考慮していないが, われわれの二方式では考慮しているため, 単純な並列版であるわけではない.

MGTP 連結方式では, モデル生成を行う MGTP (以下 **MGTP-1**) と無矛盾性の検査を行う別の MGTP (以下 **MGTP-2**) とを組み合わせる. **MGTP-1** は保持しているモデル候補 M との連言照合に成功すると, 新しい基礎アトムを導くが, そのときにその後件を支持する仮説集合 (各前件を支持する仮説集合の和集合) の無矛盾性を **MGTP-2** によって検査する. 無矛盾性の検査を複数同時に行えるとき, 必要に応じて **MGTP-2** を複数生成し並列処理の効果を得る.

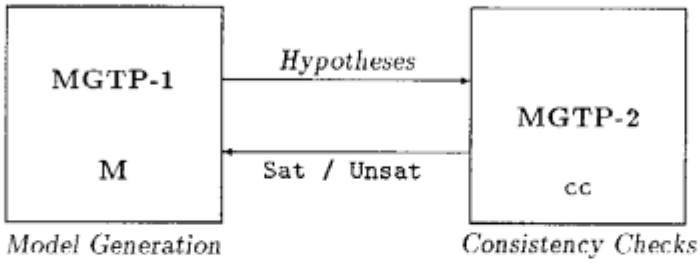

図 3: MGTP 連結方式

3. Nogood 仮定方式

上記2の MGTP 連結方式においては,新しく導かれた基礎アトムを支持する仮説集合(各前件を支持する仮説集合の合成)の無矛盾性検査を,別の MGTP を呼ぶことで実現していた.これに対して本 Nogood 仮定方式では,仮説集合を新しく合成すると,それが矛盾すると仮定するモデル候補と,無矛盾であると仮定するモデル候補とにモデル候補を二分岐する.矛盾すると仮定する方では結論をモデル候補の中に含めない.また,無矛盾であると仮定した方のモデル候補では,その仮説集合に支持される形で結論をモデル候補に含める.この方式も含めて以下の3方式では分岐したモデル候補について,それぞれ独立に推論を行なうことで並列化を実現している.

4. 全モデル生成方式

この全モデル生成方式は,仮説を用いてボトムアップに推論を実現する最も直接的な方法である.本方式では,すべての仮説についてその生成を,仮説を含めるモデル候補とそれを含めないモデル候補とに二分岐することにより表し,それぞれについて独立に推論を行う.また,上記3の Nogood 仮定方式では仮説集合ごとに対する分岐であり,本方式における各仮説に対する分岐とは異なることに注意されたい.

5. Skip 方式

Skip 方式では,全モデル生成方式同様,仮説を扱うときにモデル候補を仮説が成り立つか成り立たないかで分岐するが,このモデル候補分岐をできる限り遅延させる方法であり,モデルの分岐数も少なくて済むという利点を有する.ルール前件中で仮定可能な(*abducible*)アトムが出現すると,後件に移し(すなわち,*skip*し),そのときに初めて skip した仮説についてモデル候補分岐を行う.つまり,前件で abducible でないアトムの連言照合が終わるまで,モデル候補分岐は生じないため,不必要な仮説の生成が抑制され高速化されている.このモデル候補分岐の遅延法は,MGTP による論理プログラムの失敗による否定(negation as failure)の処理[10]と同じ考え方に基づいている.また,後述する Skip 方式に対する変換では,前提部における各仮説はマッチングの対象から除外され右辺に skip されるが,この操作は,トップダウンアプローチである SOL- 導出[8]における **Skip** ルールのボトムアップ処理に相当している.

4.2 入力変換規則

開発した5つの方式による仮説推論システムにおいては,入力節の集合 P をそれぞれ以下のように変換する.これらの変換を *Normal translation* と呼ぶ.

1. **MGTP+ATMS 方式**

この方式では P をそのまま用いる.MGTP において,代入 δ のもとで,(3)式の形のルール

$$A_1, \ldots, A_n \rightarrow \mathbf{assume}\ C$$

が適用可能になったとき，ATMS は $assume(C\delta)$ をその時点での仮説集合に追加し，同時に理由付けの集合に

$$A_1\delta, \ldots, A_n\delta, assume(C\delta) \rightarrow C\delta$$

を追加しラベル更新 [3, 14] を行う．ただし，$assume$ は P 中に述語記号として現れないものとする．

2. MGTP 連結方式

MGTP-1 では，入力節集合 P を次のルール集合（これを $P^N_{(1)}$ とする）に変換する:

$$\begin{aligned}
P^N_{1_{(1)}} &: \quad \rightarrow fact(C, \emptyset) \\
P^N_{2_{(1)}} &: \quad fact(A_1, E_1), \ldots, fact(A_n, E_n) \rightarrow fact(C, \mathrm{cc}(\bigcup_{i=1}^{n} E_i)) \\
P^N_{3_{(1)}} &: \quad \text{不要} \\
P^N_{4_{(1)}} &: \quad \rightarrow fact(C, \mathrm{cc}(\{assume(C)\})) \\
P^N_{5_{(1)}} &: \quad fact(A_1, E_1), \ldots, fact(A_n, E_n) \\
&\qquad \rightarrow fact(C, \mathrm{cc}(\bigcup_{i=1}^{n} E_i \cup \{assume(C)\}))
\end{aligned}$$

ここで，述語 $fact$ はその第2引数に第1引数 A が信じられているとする根拠となる仮説の集合 E を持つ．すなわち，$fact(A, E)$ で $\Sigma \cup E \models A$ なるメタ的な意味を表す．こうして，現時点で信じられている基礎アトムの集合 M が，$fact(A, E)$ なる形式で **MGTP-1** に保存される．また，$\mathrm{cc}(E)$ は仮説集合 E の無矛盾性を **MGTP-2** によって検査する関数であり，次で定義される:

$$\mathrm{cc}(E) = \begin{cases} E & \Sigma \cup E \text{ が無矛盾;} \\ \mathrm{nil} & \text{その他.} \end{cases}$$

なお，この無矛盾性の検査は **MGTP-2** で行われるため，負節 (P_3) は **MGTP-1** では不要である．

一方，**MGTP-2** では P を以下のルール集合（これを $P^N_{(2)}$ とする）に変換する:

$$\begin{aligned}
P^N_{1_{(2)}} &: \quad \rightarrow C \\
P^N_{2_{(2)}} &: \quad A_1, \ldots, A_n \rightarrow C \\
P^N_{3_{(2)}} &: \quad A_1, \ldots, A_n \rightarrow \\
P^N_{4_{(2)}} &: \quad assume(C) \rightarrow C \\
P^N_{5_{(2)}} &: \quad A_1, \ldots, A_n, assume(C) \rightarrow C
\end{aligned}$$

MGTP-1 において $P^N_{(1)}$ に含まれるルールが，ある代入 δ のもとで適用可能になると，その前件に含まれる仮説 $E_1, \ldots, E_n$ の和集合，あるいはそれに新たに生成した仮説 $assume(C\delta)$ を加えた仮説集合 E を **MGTP-2** に送る．**MGTP-2** は $P^N_{(2)} \cup E$ が充足不可能であれば（つまり矛盾すれば），関数 cc の値として空 (nil) を返し，そうでなければ cc の値として E 自身を返す．こうして，**MGTP-1** が新しい基礎アトムを導くたびに，合成された仮説集合の無矛盾性が **MGTP-2** によって検査される．

3. Nogood 仮定方式

Nogood 仮定方式では，P を以下のように変換する:

$$
\begin{aligned}
P_1^N &: \quad \rightarrow fact(C,\emptyset)\\
P_2^N &: \quad fact(A_1,E_1),\ldots,fact(A_n,E_n) \rightarrow \\
&\qquad \mathsf{M}good(\bigcup_{i=1}^{n} E_i),\ fact(C,\bigcup_{i=1}^{n} E_i) \mid \mathsf{M}nogood(\bigcup_{i=1}^{n} E_i)\\
P_3^N &: \quad fact(A_1,E_1),\ldots,fact(A_n,E_n) \rightarrow nogood(\bigcup_{i=1}^{n} E_i)\\
P_4^N &: \quad \rightarrow fact(C,\{C\})\\
P_5^N &: \quad fact(A_1,E_1),\ldots,fact(A_n,E_n) \rightarrow \\
&\qquad \mathsf{M}good(\bigcup_{i=1}^{n} E_i \cup \{C\}),\ fact(C,\bigcup_{i=1}^{n} E_i \cup \{C\}) \mid \mathsf{M}nogood(\bigcup_{i=1}^{n} E_i \cup \{C\})
\end{aligned}
$$

ここで $\mathsf{M}good(E)$ ($\mathsf{M}nogood(E)$) は，仮説集合 E が無矛盾である (矛盾している)，と仮定することを示す．P_2^N および P_5^N においては，これらのルールが適用可能になるとモデル候補を 2 つに分岐する．ここで，$\mathsf{M}good(E)$ および $\mathsf{M}nogood(E)$ に関して，これらの式を論理的に取り扱うための条件として次のスキーマを与える:

任意の二つの仮説集合 E, E' に対して，

$$
\begin{aligned}
&\mathsf{M}good(E),\ nogood(E'),\ E' \subseteq E \rightarrow\ ,\\
&\mathsf{M}good(E),\ \mathsf{M}nogood(E'),\ E' \subseteq E \rightarrow\ .
\end{aligned}
$$

すなわち，あるモデル候補で $\mathsf{M}good(E)$ あるいは $\mathsf{M}nogood(E)$ が矛盾することがわかったときにそのモデル候補を棄却する．

4. 全モデル生成方式

全モデル生成方式では，P を以下のように変換する:

$$
\begin{aligned}
P_1^N &: \quad \rightarrow C\\
P_2^N &: \quad A_1,\ldots,A_n \rightarrow C\\
P_3^N &: \quad A_1,\ldots,A_n \rightarrow\\
P_4^N &: \quad \rightarrow C \mid \neg\mathsf{K}C\\
P_5^N &: \quad A_1,\ldots,A_n \rightarrow C \mid \neg\mathsf{K}C
\end{aligned}
$$

ここで，$\neg\mathsf{K}C$ は C が信じられていない，つまり，“その解釈において H を真であると仮定しない”ことを意味する．すなわち各仮説ごとに，それが成り立つモデル候補と成り立たないモデル候補とに分岐される．したがって，$C, \neg\mathsf{K}C$ が同時に成立するようなモデル候補は棄却される．この規則は，以下のようなスキーマで表すことができる:

$$\neg\mathsf{K}C, C \rightarrow\ .$$

なお，このスキーマは失敗による否定を扱うために [10] で導入されたものと同じである．

5. Skip 方式

Skip 方式では，あらかじめ仮定可能な述語 (*abducibles*) がそれ以外のものと区別されていることを前提に変換を行う．以下，入力 P 中の各節の前件中で仮定可能な述語を $H_1,\ldots,H_m$ ($m \geq$

0) とし, $n = l + m;\ l \geq 0$ とする.

$$
\begin{aligned}
P_1 &: \quad \rightarrow C \\
P_2 &: \quad A_1, \ldots, A_l, \underbrace{H_1, \ldots, H_m}_{abducibles} \rightarrow C \\
P_3 &: \quad A_1, \ldots, A_l, \underbrace{H_1, \ldots, H_m}_{abducibles} \rightarrow \\
P_4 &: \quad \rightarrow \mathbf{assume}\ C \\
P_5 &: \quad A_1, \ldots, A_l, \underbrace{H_1, \ldots, H_m}_{abducibles} \rightarrow \mathbf{assume}\ C
\end{aligned}
$$

このとき, 各々のルールは以下のように変換される.

$$
\begin{aligned}
P_1^N &: \quad \rightarrow C \\
P_2^N &: \quad A_1, \ldots, A_n \rightarrow H_1, \ldots, H_m, C \mid \neg \mathsf{K} H_1 \mid \ldots \mid \neg \mathsf{K} H_m \\
P_3^N &: \quad A_1, \ldots, A_n \rightarrow \neg \mathsf{K} H_1 \mid \ldots \mid \neg \mathsf{K} H_m \\
P_4^N &: \quad \text{不要} \\
P_5^N &: \quad A_1, \ldots, A_n \rightarrow H_1, \ldots, H_m, C \mid \neg \mathsf{K} H_1 \mid \ldots \mid \neg \mathsf{K} H_m \mid \neg \mathsf{K} C
\end{aligned}
$$

この変換において, P_4 の各要素に対して, 全モデル生成方式で用いたような C と $\neg \mathsf{K} C$ に分岐するような MGTP ルールを用意しないことに注意されたい. つまり, すべての仮説について二分岐してしまう代わりに, 仮説がある結論を導くために必要となったところで初めて導入する. また, 全モデル生成方式と同様に, H と $\neg \mathsf{K} H$ が同時に成立するようなモデル候補は次のスキーマで棄却される:

$$\neg \mathsf{K} H, H \rightarrow .$$

4.3 不要なモデル候補の枝刈り

前述の通り, MGTP を用いた 5 つの仮説推論システムのうち, 全モデル生成方式と Skip 方式は, 仮説の基礎例ごとにモデル候補分岐を起こす方式である. したがって, MGTP+ATMS 方式や MGTP 連結方式における MGTP のモデル生成では 一個しか保持しないモデル候補を, これらのモデル候補分岐を起こす 2 方式では複数, それも仮説の基礎例の指数オーダーで生成するため, モデル候補数の爆発を起こす可能性がある. しかしながら, これらのモデル候補集合の中で, ゴールの基礎例を含んだモデル候補を考えると, 異なるモデル候補は異なる仮説集合と対応しているため, これらの仮説集合はゴールのすべての異なる説明に対応している.

ところが, アブダクションにおいては一般に, ゴールの極小の説明のみを求めたいことが多い. 例えば, ゴールを満たす二つのモデル候補 $M1$, $M2$ があったとき, $M1$ の中の仮説集合が $\{p(a)\}$, $M2$ の中の仮説集合が $\{p(a), p(b)\}$ であるとすると, $\{p(a)\}$ のみがアブダクションの解として妥当であると考える場合が多い. 実際, 残る 3 方式では, 極小でない説明はまったく (MGTP+ATMS 方式の場合), あるいはほとんど (MGTP 連結方式あるいは Nogood 仮定方式の場合), 生成されない. この理由は, 前者の場合 ATMS は各アトムに対して極小の説明集合 (ラベル) を保持し, またメタ述語 $fact$ を用いる後者の方式では, 基礎アトムが依存する依存する仮説集合を, 前件のすべてのアトムを支持する仮説集合の和集合によってのみ求めるような変換を行っているからである.

そこで, 全モデル生成方式や Skip 方式においても, 極小でない説明を含むモデル候補を削除する方法が望まれる. ここでは, その一つとして, "枝刈りルール" と呼ばれるルールをルール変換時に導入することによって, MGTP における余分なモデル生成を削減する方法 [18] を示す.

いま, 述語と仮定可能述語 (abducibles) との依存関係を次のように定義する:

1. 仮定可能述語は，その述語自身にのみ**依存する**.
2. 仮定可能でない述語は，その述語が結論に現れる確定節の前件の述語が依存する仮定可能述語に**依存する**.
3. ある仮定可能述語に依存する述語が，ある確定節の前件に複数現れるとき，その確定節の後件の述語は，その仮定可能述語に**重複して依存する**.

このとき，入力節集合 P の任意の述語がある仮定可能述語 γ に重複して依存しなければ，負節

$$\gamma(\mathbf{X}),\ \gamma(\mathbf{Y}),\ \mathbf{X} \neq \mathbf{Y} \rightarrow$$

($\mathbf{X}$, $\mathbf{Y}$ は異なる変数の組) を P に加えても，任意のゴール G の極小説明は，もとの P からの極小説明と等しい．この証明，および依存関係の検出アルゴリズムについては [18] を参照されたい．この性質を利用して，重複して依存されないようなすべての仮定可能述語について，上と同様な負節を加えることができる．これらの負節を**枝刈りルール** (*Cut rule*) と呼ぶ．枝刈りルールは負節であるので，モデル候補棄却以外には使われず，したがって生成されるモデル候補数を削減することができる．

枝刈りルールは，全モデル生成方式および Skip 方式において導入することができる．他方式でも導入できるが，前述の通り効果は無い．また Skip 方式の変換においては，仮定可能述語は通常後件に skip されるが，枝刈りルールはその変換の対象とはならないことに注意されたい．

4.4 トップダウン情報の組み込み

ここでは，トップダウン推論をシミュレートすることで，ゴールに関連のないサブゴールの充足を避け推論の高速化を行うために，*UD-meta*を基礎にした入力ルールの変換を示す．

以下に示す 2 つの変換規則は入力節の集合 P に対してまず適用され，ルール集合 P^{UD} (UD は S または LR) が生成され，その後で前節に示した *Normal translation* が P^{UD} に対して施される．ただし，Skip 方式においては，仮定可能述語には P^{UD} を生成する変換規則を適用しない．また，すべての方式において，トップダウン推論をシミュレートするためのメタ述語 (*goal*および *cont*) は *Normal translation* の対象から省かれる．付録 A に例題を用いてこれらの変換例を示しているので参照のこと．

まず，*UD-meta*の単純な変換方法は次の通りである．あるルール

$$A_1, \ldots, A_n \rightarrow C$$

にトップダウン情報を組み込むために，まず，このルールがゴール C があったときにのみ起動されるように変換する．このときには，$A_1, \ldots, A_n$ をそれぞれサブゴールとして起動するためのルールも必要となる．

また，P 中の各負節

$$A_1, \ldots, A_n \rightarrow$$

に対しては，得られる説明の無矛盾性を保証するためには，この節をゴール節のように扱って各 A_i を評価しなければならない．しかしながら，これらの前件のアトムをすべて評価の対象にした場合，無制御のボトムアップ推論と比較して，一般的に評価するアトムの数をそれほど減らすことができない．これは，すべての負節が生成されるべきゴールの説明に関係しているとは限らず，一般には無駄な計算を行ってしまうからである．そこで，*UD-meta*を用いたボトムアップ型の仮説推論システムで，ゴールの説明生成に関係した部分のみにボトムアップ計算を制限させる方法として，[17] の解析手法を用いることで，ゴールと無関係な負節をあらかじめ静的に検出して，それらの負節に関係する部分の計算を行わないような変換方法を用いた．

以下に，入力 P に対する単純な変換規則を示す．ここで，$\overline{C}$ および $\overline{A_i}$ はそれぞれ C および A_i の述語記号とそれらの引数から成るアトムである．これらの変換を，*Simple translation* と呼ぶ．

$$
\begin{aligned}
P_1^S &: \quad \rightarrow C \\
P_2^S &: \\
&\quad goal(\overline{C}) \rightarrow goal(\overline{A_i}) \\
&\quad goal(\overline{C}),\ A_1,\ \ldots,\ A_n \rightarrow C \\
P_3^S &: \\
&\quad \rightarrow goal(\overline{A_i}) \\
&\quad A_1,\ \ldots,\ A_n \rightarrow \\
P_4^S &: \quad \rightarrow \textbf{assume}\ C \\
P_5^S &: \\
&\quad goal(\overline{C}) \rightarrow goal(\overline{A_i}) \\
&\quad goal(\overline{C}),\ A_1,\ \ldots,\ A_n \rightarrow \textbf{assume}\ C
\end{aligned}
$$

ここで，P_3^S に関しては前述の通り，[17] の手法によって検出される，与えられたゴールに関係する負節のみについて変換を行うものとする．

次に，各前件をサブゴールとして起動する場合，$goal(A_{i+1})$ が $goal(A_1)$, ..., $goal(A_i)$ $(1 \leq i \leq n-1)$ がすべて解かれた後にのみ起動されるような，トップダウン推論における left-to-right な解法をシュミレートする変換規則を示す．これらの変換を，*Left-to-Right translation* と呼ぶ．

$$
\begin{aligned}
P_1^{LR} &: \quad \rightarrow C \\
P_2^{LR} &: \\
&\quad goal(C) \rightarrow goal(A_1),\ cont_{k,1}(V) \\
&\quad cont_{k,1}(V),\ A_1 \rightarrow goal(A_2),\ cont_{k,2}(V) \\
&\quad\quad \vdots \\
&\quad cont_{k,m}(V),\ A_m \rightarrow C \\
P_3^{LR} &: \\
&\quad \rightarrow goal(A_i) \\
&\quad A_1,\ \ldots,\ A_n \rightarrow \\
P_4^{LR} &: \quad \rightarrow \textbf{assume}\ C \\
P_5^{LR} &: \\
&\quad goal(C) \rightarrow goal(A_1),\ cont_{k,1}(V) \\
&\quad cont_{k,1}(V),\ A_1 \rightarrow goal(A_2),\ cont_{k,2}(V) \\
&\quad\quad \vdots \\
&\quad cont_{k,m}(V),\ A_m \rightarrow \textbf{assume}\ C
\end{aligned}
$$

ここで，$m \geq 2$ とする．また，k は入力節集合 P 中の 1 つの節ごとに付けられる識別番号である．さらに，$cont_{k,i}(V)$ は A_1, ..., A_{n-1} に対する変数の束縛情報を V に保持していることを意味する．ここでも，P_3^{LR} に関しては，与えられたゴールに関係する負節のみについて変換を行うものとする．

5 評価

開発した5種類の仮説推論システムを2つの例題を用いて評価する.

5.1 例題および測定結果

開発した各方式の仮説推論システムを評価するため以下の2つの問題を取り上げた.

1. **会議開催問題** [14].
   使用したい会議室の空き具合と,出席者の都合とを制約として,会議のスケジューリングを行う問題(*Meeting*で表す)である.付録Aに会議開催問題における *Normal*, *Simple*, および *Left-to-Right*の *translation*を用いた変換例を示す.ただし,以下の計測では *Normal translation*のみについて測定した.これは,問題のサイズが小さく,かつとくに意味のあるような部分問題が存在しないためである.

2. **論理回路設計問題** [13, 15].
   入力された2つの整数の最大公約数を計算する機能を持ち,チップ面積(基本セル数がある値以下)と性能(遅延時間がある値以下)に関する制約を満たす回路を設計する問題である.ここでは,基本セル数が(a) 300以下,(b) 400以下,(c) 500以下,それに対応して遅延時間が(a) 40ns以下,(b) 50ns以下,(c) 60ns以下の場合について評価した.本問題に対する計測は,*Normal*, *Simple*, および *Left-to-Right*のすべての変換を用いて行った.ここで,*Normal*の変換では,回路全体の設計を求める場合(*Circuit*で表す)について計測を行い,*Simple*および *Left-to-Right*の変換では,回路全体のゴールに対してサブゴールの関係にある,減算器のみの設計を求める場合(*Subtracter*で表す),について計測を行った.

64台版のPIM/mを用いて計測した結果を,*Normal*, *Simple*, および *Left-to-Right*の各変換を用いた場合に分けて表1に示す.

さらに,論理回路設計問題(*Circuit*)において,制約を *limit 500-60* とした場合の,(1)実行時間グラフを図4に,また(2)並列台数効果グラフを図5に,それぞれ示す.

5.2 考察

最初に,モデル生成と無矛盾性検査を別のモジュールとして連結した2つの方式について考察する.MGTP+ATMS方式では,並列処理による台数効果がMGTP連結方式に比べて小さい.これは,MGTPとATMSとの間の通信路は1本であるため,MGTPがしばしばATMSの計算結果を待つことになり,並列化の効果には限界があることが理由によると考えられる.この現象はとくに次の二点において顕著に現れる.

1. MGTP+ATMS方式における並列化は,組み合わせた並列ATMSにおけるラベル計算の並列処理によって実現されている.モデル候補を保持する並列ATMSは,ATMSデータであるモデル候補の要素を複数のプロセッサに分散させて管理している.このため,MGTPにおいて連言照合のためのデータの収集のコストが高くなる.これに対して,MGTP連結方式では一度に生成される複数の **MGTP-2** に並列性があり,推論を行う **MGTP-1** のみがモデル候補を保持しているため,連言照合のためのデータ収集のときに問題は生じない.

2. ボトムアップにモデル計算を行う推論エンジンであるMGTPは,ATMS中で信じられているデータの集合をモデル候補としてそれを基に推論を行う.このため,新たにルール中の負節への代入から矛盾する仮説集合が判明するたびに,ATMSがモデル中のすべてのデータのラベルの無矛盾性を検査し直すまでの間,MGTPでの推論は待たされる.これに対してMGTP連結方式では,**MGTP-2** が **MGTP-1** から与えられる仮説集合が矛盾するか否かを検査するのみであ

表 1: 仮説推論システム性能対比表

性能対比表 1: *Normal translation*

problem		System type				
		M+A	M+M	Nogood	All Model	Skip
Meeting		0.4/ 1.0(8)	0.1/ 2.1(64)	14.3/ 19.1(64)	0.5/ 1.9(64)	0.1/1.0(1)
Circuit	(limit 300-40)	64.4/ 1.1(16)	4.0/ 7.5(64)	-	7.0/ 14.1(64)	7.2/ 13.7(64)
	(limit 400-50)	63.9/ 1.1(16)	31.2/ 3.3(64)	-	19.3/ 23.7(64)	23.3/ 19.4(64)
	(limit 500-60)	65.8/ 1.0(1)	54.3/ 2.6(64)	-	22.9/ 22.8(64)	26.1/ 20.6(64)

性能対比表 2: *Simple translation*

problem		System type				
		M+A	M+M	Nogood	All Model	Skip
Subtracter	(limit 500-60)	0.5/ 1.0(8)	1.1/ 1.3(64)	-	0.3/ 1.3(64)	0.6/ 1.2(64)

性能対比表 3: *Left-to-Right translation*

problem		System type				
		M+A	M+M	Nogood	All Model	Skip
Subtracter	(limit 500-60)	18.1/ 1.0(1)	0.8/ 1.0(16)	-	14.6/ 4.2(64)	12.7/ 3.1(64)

M+A :MGTP+ATMS 方式
M+M :MGTP 連結方式
Nogood :Nogood 仮定方式
All Model :全モデル生成方式
Skip :Skip 方式

表中のデータ表記: T / $R(PEs)$
T : 結論を得るまでの最小時間 (sec)
(注) " - " は 30 分以上.
R : 1 台のプロセッサを用いたときに対する T の Speed up 率 (倍)
PEs : T が求まったときの使用プロセッサ数 (台)

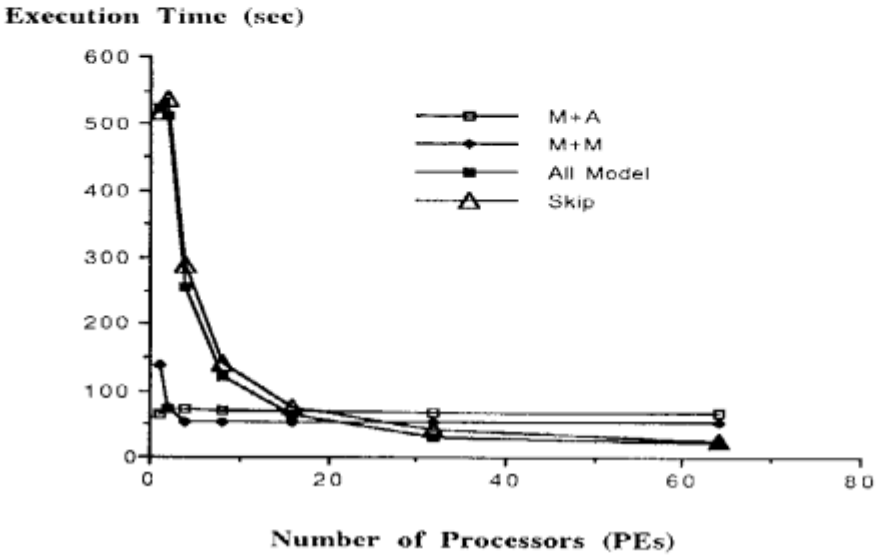

図 4: MGTP を用いた仮説推論システム: 実行時間 (*Circuit: limit 500-60*)

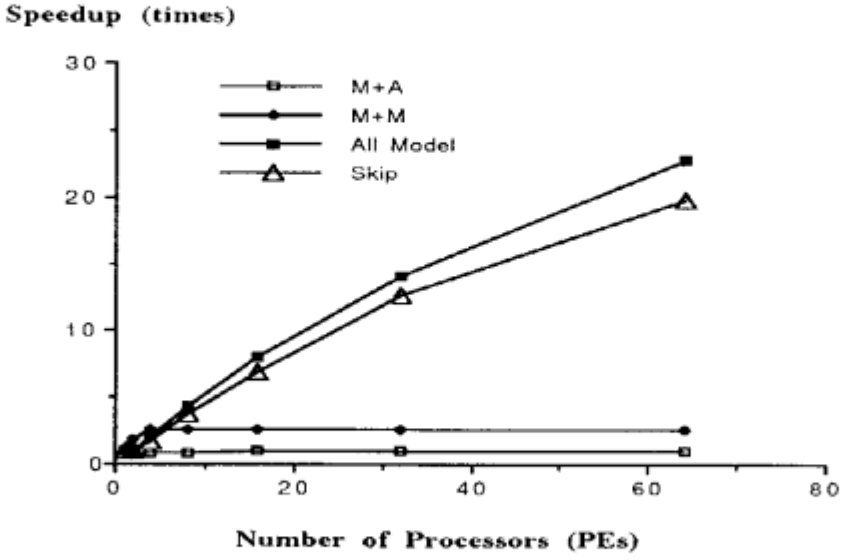

図 5: MGTP を用いた仮説推論システム: 台数効果 (*Circuit: limit 500-60*)

り，**MGTP-1** が保持するモデル候補全体に対する無矛盾性の検査を行わない．したがって，複数の **MGTP-2** を並列に処理することで MGTP+ATMS 方式で生じるような推論エンジンの待ちを少なくすることが可能となる．

このような理由により MGTP 連結方式は MGTP+ATMS 方式に対して，実行時間および並列化による台数効果において優れた結果を示した．

しかしながら，MGTP 連結方式では **MGTP-1** が並列化されていないので，その並列化の効果は，どれだけ多くの無矛盾性検査が並列になされるかに大きく依存している．つまり，2 つの異なったコンポーネントから構成されているので，並列化の可能性には限界がある．

一方，1 台の MGTP によって構築された 3 つのシステムは，MGTP+ATMS 方式や MGTP 連結方式では別システムとして行っているような無矛盾性の検査を，1 台の MGTP の中に組み込んだ方式である．すなわち，これらのシステムでは，大域的な信念の集合を保存する代わりに，複数のモデル候補を分散メモリ中に各々保存すればよいために，上記 2 つの方式よりも多くの並列性を引き出すことができる．したがって，MGTP+ATMS 方式および MGTP 連結方式とは異なり，組み合わせたシステム間の通信コストや，無矛盾性の検査の結果を待って推論が行なわれるといった問題はない．

ところが，1 台の MGTP によって構築されたシステムでは，仮説生成の段階でモデル候補を分岐させるためにその数の増加が問題となる．以下，N をすべての仮説の基礎例の数とするときに，3 つのシステムのモデル候補生成に関する計算量について考察する．

1. Nogood 仮定方式では，生成した仮説ごとに矛盾と無矛盾を仮定することから，生成するモデル候補数は最悪で 2^{2^N} となり，以下で述べる全モデル生成および Skip 方式のそれをはるかに上回る．また，スキーマ前件にある仮説集合の包含関係のチェックのコストも高く，効率において Nogood 仮定方式は他の 2 システムに大きく劣る．
2. 全モデル生成方式は，仮説の基礎例各々に対して，それが成り立つモデル候補と成り立たないモデル候補とに分岐するため，全部で 2^N のモデル候補を生成する．したがって，とくに無条件で仮説を生成するルール (入力 P_4 で表されるルール) の多い問題では，非現実的な数のモデル候補を生成することになり，実用的な応用ではモデル候補の数が爆発してしまう可能性がある．
3. これに対して Skip 方式は，生成されるモデル候補の数を可能な限り減少させるために，仮説におけるモデル候補分岐を遅延させる方法である．つまり Skip 方式では，入力 P 中で仮定可能なものを含むルールの数を k とし，1 つのルールに現れる仮定可能なものの最大数を m，仮説はすべて他の仮説から独立したものであるとすると，最大の場合で $(m+1)^k$ 個のモデル候補を生成する．このとき全モデル生成方式では 2^{mk} 個のモデル候補を生成してしまう．したがって，$m \geq 1$ であれば Skip 方式でのモデル候補の生成数は，全モデル生成方式のそれを下回ることになる．会議開催問題ではこのため，全モデル方式と比べて優れた結果が得られている．しかし，論理回路設計問題のように，無条件に仮説を生成するルールが存在しないような問題では，全モデル生成方式と比較してその効果を十分には発揮することができない．

ところで，論理設計問題で制約を *limit 400-50* 以下とすると，全モデル生成方式で 2^{44} 以上のモデル候補が存在することがわかっており，極小でない説明を含むモデル候補を削除する枝刈りルールを導入しない場合，MGTP を 1 台のみ用いる 3 つの方式ではどれも現実的に解けなかった．そこで，表 1，図 4，図 5 においては，全モデル生成方式と Skip 方式に関しては，枝刈りルールを導入した場合の結果を示している．

5.3 枝刈りルールの効果

ここでは，枝刈りルール導入による計算量削減の効果について考察する．ここでは，枝刈りルールを Skip 方式に適用し，MGTP 連結方式と比較してその有効性を示す．ただし，一般的な議論では，最

悪の場合どちらも仮説の基礎例の数の指数オーダーとなってしまうので，枝刈りルールが有効に働く典型的な次の入力節集合 P について考察する:

$$\begin{aligned} &\rightarrow && q(c_1)\\ &\rightarrow && q(c_2)\\ &&& \vdots\\ &\rightarrow && q(c_n)\\ p(X),\ q(X) &\rightarrow && r(X)\\ r(X) &\rightarrow && g(X)\\ q(X) &\rightarrow && \textbf{assume}\ p(X) \end{aligned}$$

ここで，p は仮定可能述語であることに注意されたい．

まず，MGTP 連結方式では，次のような MGTP ルールに変換される:

$$\begin{aligned} &\rightarrow && fact(q(c_1),\emptyset)\\ &\rightarrow && fact(q(c_2),\emptyset)\\ &&& \vdots\\ &\rightarrow && fact(q(c_n),\emptyset)\\ fact(p(X),E_1),\ fact(q(X),E_2) &\rightarrow && fact(r(X),\mathrm{cc}(E_1\cup E_2))\\ fact(r(X),E) &\rightarrow && fact(g(X),\mathrm{cc}(E))\\ fact(q(X),E) &\rightarrow && fact(p(X),\mathrm{cc}(E\cup\{p(X)\})) \end{aligned}$$

この場合，MGTP での照合回数は，$O(n^2)$ である．また，最大 n 個の関数 cc の評価が並列に実行されるが，これは照合に要する時間に無関係であるため，実際に並列実行したときの実時間（以下，計算時間という）も $O(n^2)$ である．

次に，Skip 方式では，次のような MGTP ルールに変換される:

$$\begin{aligned} &\rightarrow && q(c_1)\\ &\rightarrow && q(c_2)\\ &&& \vdots\\ &\rightarrow && q(c_n)\\ q(X) &\rightarrow && p(X),\ r(X) \mid \neg \mathsf{K}\, p(X)\\ r(X) &\rightarrow && g(X) \end{aligned}$$

この場合，非ホーン節の前件で n 個の要素が照合され，二分岐によって生成されるプロセス数は 2^n 個であり，この分岐した各プロセス内で独立に照合が行われる．ここで最悪の場合の照合回数は n であり，その場合 MGTP での照合回数は $O(n\cdot 2^n)$ である．ここで，並列化する要素プロセッサの数を n と同じだけ取っても，$O(2^n)$ の計算時間がかかる．

上の例題のような入力節集合において，仮定可能述語の基礎例の数（この場合 n）が比較的大きく，その仮定可能述語（この場合 p）が任意の述語（この場合，p, q, r, g）に重複して依存しない場合に，枝刈りルールが有効に作用することを示す．この場合の枝刈りルールは，

$$p(X),\ p(Y),\ X\neq Y \rightarrow$$

である． いま，非ホーン節の前件で $q(c_1)$ が照合されると二分岐する．次にその非ホーン節の前件で $q(c_2)$ が照合され，組み合わされて 4 分岐となる． ここで，上の枝刈りルールによって，$p(c_1)$ と $p(c_2)$ を含むモデル候補は即座に棄却される．残りのモデル候補は，仮定可能述語の基礎例が 1 個以下入ったもののみである．同様にして，以降 n 段まで実行されるが，モデル候補数は $O(n)$ でおさえられ，さら

に各モデル候補においては，仮定可能述語の基礎例の数は2個を越えないので，照合回数も $O(n)$ でおさえられる．よって，MGTP での照合回数は $O(n^2)$ である．ここで，要素プロセッサの数を n と同じだけ取れるとすると，$O(n)$ の計算時間で済むことになる．したがって，枝刈りルールを先の変換プログラムに加えて MGTP で推論を行うと，計算量を大幅に減少させることができる．

6 まとめ

5種類の仮説推論システムを提案しその評価を行った．とくに MGTP 連結方式と Skip 方式に対して，その効率において優れた可能性が認められた．しかし，MGTP 連結方式では，根本的に組み合わせたシステム間の通信コストや，無矛盾性の検査の結果を推論が待たされるといった問題は残っており，より大きな問題の解法を行う場合，並列性の向上に限界が生じることが予想される．一方，Skip 方式では，基本的に待ちの問題は起きないが，MGTP 連結方式では1個しかないモデル候補を指数オーダーで作ってしまう問題がある．そこで，ルールを変換する際に，ゴールを含むモデル候補のうち，極小でない説明を含むものを削除するための枝刈りルールを導入することによって，MGTP での余分なモデル生成の削減に成功した．

ところで，これらの“待ち”と“モデル爆発”とは，ある種のトレードオフの関係にあると考えられる．すなわち，無矛盾性検査のコストが小さく，検査の回数が多い場合は，比較的待ちが少く済むため MGTP 連結方式による性能を引き出すことができる．実験した論理回路設計問題はこのような場合にあてはまるため，並列化による効果は認められないもののある程度速い時間で解くことができた．逆に無矛盾性検査のコストが大きく，検査の回数が比較的少ない場合は，Skip 方式が適していると考えられる．

最後に，[10] において，論理プログラムにおける**失敗による否定**(*negation as failure*) を MGTP ルールに変換して計算する方法が与えられている．これは，**安定モデル意味論**(*stable model semantics*) [5, 6] に対して，健全かつ完全な計算手続きとなっている．また，失敗による否定がプログラムに含まれている場合のアブダクションの枠組が，[11, 7] で提案されている．これらの枠組は，知識ベース内に，仮定可能な述語，および失敗による否定のような非単調な効果を持つオペレータ，の両方が含まれているような場合に対する論理的意味を与えている．Skip 方式は容易に [10] の技術と MGTP 上で統合できるという利点を有しており，これらの拡張されたアブダクションの枠組を計算するためにも用いることができる．

謝辞

本研究に関して議論頂いた Mark Stickel 氏，François Bry 氏および古川康一氏，ならびに本研究の機会を与えて下さった ICOT の淵一博所長に感謝します．

参考文献

[1] F. Bancilhon, D. Maier, Y. Sagiv and J.D. Ullman. Magic sets and other strange ways to implement logic programs. In: *Proc. 5th ACM SIGMOD-SIGACT Symp. Principles of Database Systems*, pages 1–15, 1986.

[2] F. Bry. Query evaluation in recursive databases: bottom-up and top-down reconciled. *Data & Knowledge Engineering*, 5:289–312, 1990.

[3] J. de Kleer. An assumption-based TMS. *Artif. Intell.*, 28:127–162, 1986.

[4] H. Fujita and R. Hasegawa. A model generation theorem prover in KL1 using a ramified-stack algorithm. In: *Proc. 8th Int. Conf. Logic Programming*, pages 494–500, 1991.

[5] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In: *Proc. 5th Int. Conf. and Symp. Logic Programming*, pages 1070–1080, 1988.

[6] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[7] K. Inoue. Extended logic programs with default assumptions. In: *Proc. 8th Int. Conf. Logic Programming*, pages 490–504, 1991.

[8] K. Inoue. Consequence-finding based on ordered linear resolution. In: *Proc. IJCAI-91*, pages 158–164, 1991. An extended version: Linear resolution for consequence-finding. To appear in: *Artif. Intell.*, 56, 1992.

[9] 井上 克巳. アブダクションの原理. 人工知能学会誌, 7(1):48 59, 1992年1月.

[10] K. Inoue, M. Koshimura and R. Hasegawa. Embedding negation as failure into a model generation theorem prover. In: *Proc. 11th Int. Conf. Automated Deduction*, Lecture Notes in Artificial Intelligence, 607, pages 400–415, Springer-Verlag, 1992.

[11] A.C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In: *Proc. ECAI-90*, pages 385–391, 1990.

[12] D.W. Loveland. *Automated Theorem Proving: A Logical Basis*. North-Holland, 1978.

[13] F. Maruyama, T. Kakuda, Y. Masunaga, Y. Minoda, S. Sawada and N. Kawato. CoLODEX: a cooperative expert system for logic design. In: *Proc. Int. Conf. Fifth Generation Computer Systems 1988*, pages 1299–1169, 1988.

[14] 中島 誠, 太田 好彦, 井上 克巳. KL1による並列ATMS. ICOT Technical Report TR-667, ICOT, 1991年6月.

[15] Y. Ohta and K. Inoue. A forward-chaining multiple context reasoner and its application to logic design. In: *Proc. 2nd IEEE Int. Conf. Tools for Artificial Intelligence*, pages 386–392, 1990.

[16] 太田 好彦, 井上 克巳. ATMSを用いた前向き仮説推論システムにおける効率的な推論方式. 人工知能学会誌, 6(2):247 259, 1991年3月.

[17] Y. Ohta and K. Inoue. A forward-chaining hypothetical reasoner based on upside-down meta-interpretation. In: *Proc. Int. Conf. Fifth Generation Computer Systems 1992*, pages 522–529, 1992.

[18] 太田 好彦, 井上 克巳, 長谷川 隆三, 中島 誠. モデル生成型定理証明器を用いたアブダクションの計算における効率化手法. ICOT Technical Report (to appear), ICOT, 1992年8月.

[19] D. Poole, R. Goebel and R. Aleliunas. Theorist: a logical reasoning system for defaults and diagnosis. In: N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–352, Springer-Verlag, 1987.

[20] M.E. Stickel. Upside-down meta-interpretation of the model elimination theorem-proving procedure for deduction and abduction. ICOT Technical Report TR-664, ICOT, 1991.

[21] K. Ueda and T. Chikayama. Design of the kernel lnaguage for the prallel inference machine. *Computer J.*, 33:494–500, 1990.

A 変換例

以下に，例題として用いた会議開催問題をそれぞれのシステム向けに，3種類の変換規則で変換したものをMGTPルールの形で示す．まず，入力 P として，会議開催問題を表す論理式集合を次のように与える．

$$\rightarrow cand(a, self)$$
$$\rightarrow cand(a, agent)$$
$$\rightarrow cand(b, self)$$
$$\rightarrow cand(b, agent)$$
$$\rightarrow room(212)$$
$$\rightarrow room(213)$$
$$\rightarrow day(mon)$$
$$\rightarrow day(tue)$$
$$\rightarrow reserved(212, mon)$$
$$\rightarrow reserved(213, mon)$$
$$\rightarrow reserved(213, tue)$$
$$room(Room),\ day(Day) \rightarrow \textbf{assume}\ opR(Room, Day)$$
$$day(Day),\ cand(Name, Id) \rightarrow \textbf{assume}\ able(Name, Id, Day)$$
$$opR(Room, Day),\ reserved(Room, Day) \rightarrow$$
$$able(a, agent, mon) \rightarrow$$
$$able(b, self, tue) \rightarrow$$
$$able(a, Ida, Day),\ able(b, Idb, Day), opR(Room, Day) \rightarrow open(Day, Room, Ida, Idb)$$

以下のMGTPルール中に現れるユーザ定義関数の仕様は次の通りである．

- `cc(+List,-UnionList,-CFlag)`
  MGTP連結方式で用いる．`List` はMGTP-1で生成された仮説集合を要素とするリストであり，それらの論理和集合 E がMGTP-2において矛盾しないとき，`UnionList` に E を返し，`CFlag` に `in` を返す．もし，矛盾すれば `UnionList` に空リストを返し，`CFlag` に `out` を返す．
- `new_assumption(+Assumption,-Bitvector)`
  MGTP連結とNogood仮定方式において用いる．仮説集合の和集合計算の高速化のため，仮説 `Assumption` のビットベクタ表現[3]を求め，`BitVector` に返す．
- `superE(+E1,+E2,-Yn)`
  Nogood仮定方式で用いる．仮説集合 `E1` が `E2` を包含していれば `Yn` に `yes`，そうでなければ `no` を返す．
- `union(+List,-UnionList)`
  Nogood仮定方式で用いる．`List` は仮説集合のリスト．それらの論理和を `UnionList` に返す．

A.1 *Normal translation*

```
%%%%%-------------------------------------------------------%%%%%
            MGTP+ATMS 方式
%%%%%-------------------------------------------------------%%%%%
:- problem("meeting").
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
true --> reserved(213,tue).
room(Room),day(Day) --> assume(opR(Room,Day)).
day(Day),cand(Name,Id) --> assume(able(Name,Id,Day)).
opR(Room,Day),reserved(Room,Day) --> false.
able(a,agent,mon) --> false.
able(b,self,tue) --> false.
able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day) --> open(Day,Room,Ida,Idb).
%%%%%-------------------------------------------------------%%%%%
            MGTP 連結方式
%%%%%-------------------------------------------------------%%%%%
%
% MGTP-1
%
:- problem("meeting").
true --> fact(cand(a,self),{[],32#{0}},in).
true --> fact(cand(a,agent),{[],32#{0}},in).
true --> fact(cand(b,self),{[],32#{0}},in).
true --> fact(cand(b,agent),{[],32#{0}},in).
true --> fact(room(212),{[],32#{0}},in).
true --> fact(room(213),{[],32#{0}},in).
true --> fact(day(mon),{[],32#{0}},in).
true --> fact(day(tue),{[],32#{0}},in).
true --> fact(reserved(212,mon),{[],32#{0}},in).
true --> fact(reserved(213,mon),{[],32#{0}},in).
true --> fact(reserved(213,tue),{[],32#{0}},in).
fact(room(Room),_H1,in),fact(day(Day),_H2,in) --> fact(opR(Room,Day),_H,_Io),
        {{& table<<=[new_assumption(opR(Room,Day),_Na)],
          & mgtp2<<=[cc([_H1,_H2,{[assume_opR(Room,Day)],_Na}],_H,_Io)]}}.
fact(day(Day),_H1,in),fact(cand(Name,Id),_H2,in) --> fact(able(Name,Id,Day),_H,_Io),
        {{& table<<=[new_assumption(able(Name,Id,Day),_Na)],
          & mgtp2<<=[cc([_H1,_H2,{[assume_able(Name,Id,Day)],_Na}],_H,_Io)]}}.
fact(able(a,Ida,Day),_H1,in),fact(able(b,Idb,Day),_H2,in),fact(opR(Room,Day),_H3,in) -->
        fact(open(Day,Room,Ida,Idb),_H,_Io),{{& mgtp2<<=[cc([_H1,_H2,_H3],_H,_Io)]}}.
%
% MGTP-2
%
:- problem("meetingORG").
```

```
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
true --> reserved(213,tue).
room(Room),day(Day),assume_opR(Room,Day) --> opR(Room,Day).
day(Day),cand(Name,Id),assume_able(Name,Id,Day) --> able(Name,Id,Day).
opR(Room,Day),reserved(Room,Day) --> false.
able(a,agent,mon) --> false.
able(b,self,tue) --> false.
able(a,Ida,Day),able(b,idb,Day),opR(Room,Day) --> open(Day,Room,Ida,Idb).
%%%%%------------------------------------------------------%%%%%
       Nogood 仮定方式
%%%%%------------------------------------------------------%%%%%
:- problem("meetingORG").
true --> fact(cand(a,self),{[],32#{0}}).
true --> fact(cand(a,agent),{[],32#{0}}).
true --> fact(cand(b,self),{[],32#{0}}).
true --> fact(cand(b,agent),{[],32#{0}}).
true --> fact(room(212),{[],32#{0}}).
true --> fact(room(213),{[],32#{0}}).
true --> fact(day(mon),{[],32#{0}}).
true --> fact(day(tue),{[],32#{0}}).
true --> fact(reserved(212,mon),{[],32#{0}}).
true --> fact(reserved(213,mon),{[],32#{0}}).
true --> fact(reserved(213,tue),{[],32#{0}}).
fact(room(Room),_H1),fact(day(Day),_H2) --> fact(opR(Room,Day),_H3),
        {{& table<<=[new_assumption(opR(Room,Day),_Na)],
          nogood_lib:union(_H1,_H2,{[opR(Room,Day)],_Na},_H3)}},
        m(good(_H3)) ; m(nogood(_H3)).
fact(day(Day),_H1),fact(cand(Name,Id),_H2) --> fact(able(Name,Id,Day),_H3),
        {{& table<<=[new_assumption(able(Name,Id,Day),_Na)],
          nogood_lib:union(_H1,_H2,{[able(Name,Id,Day)],_Na},_H3)}},
        m(good(_H3)) ; m(nogood(_H3)).
fact(opR(Room,Day),_H1),fact(reserved(Room,Day),_H2) -->
        nogood(_H3),{{nogood_lib:union(_H1,_H2,_H3)}}.
fact(able(a,agent,mon),_H) --> nogood(_H).
fact(able(b,self,tue),_H) --> nogood(_H).
fact(able(a,Ida,Day),_H1),fact(able(b,Idb,Day),_H2),fact(opR(Room,Day),_H3) -->
        fact(open(Day,Room,Ida,Idb),_H4),
        {{nogood_lib:union(_H1,_H2,_H3,_H4)}}.
m(good(H1)),nogood(H2) --> false(YN,H1,H2),{{nogood_lib:superE(H2,H1,YN)}}.
m(good,H1)),m(nogood(H2)) --> false(YN,H1,H2),{{nogood_lib:superE(H2,H1,YN)}}.
false(yes,H1,H2) --> false.
%%%%%------------------------------------------------------%%%%%
            全モデル生成方式
```

```
%%%%%------------------------------------------------------%%%%%
:- problem("meeting").
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
true --> reserved(213,tue).
room(Room),day(Day) --> opR(Room,Day) ; -k(opR(Room,Day)).
day(Day),cand(Name,Id) --> able(Name,Id,Day) ; -k(able(Name,Id,Day)).
opR(Room,Day),reserved(Room,Day) --> false.
able(a,agent,mon) --> false.
able(b,self,tue) --> false.
able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day) --> open(Day,Room,Ida,Idb).
%%% Cut rule
opR(A,B),opR(C,D),{{noteq(A,C)}} --> false.
opR(A,B),opR(C,D),{{noteq(B,D)}} --> false.
%%%%%------------------------------------------------------%%%%%
            Skip 方式
%%%%%------------------------------------------------------%%%%%
:- problem("meeting").
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
true --> reserved(213,tue).
reserved(Room,Day) --> -k(opR(Room,Day)).
true --> -k(able(a,agent,mon)).
true --> -k(able(b,self,tue)).
cand(a,Ida),cand(b,Idb),room(Room),day(Day) -->
        able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day),open(Day,Room,Ida,Idb) ;
        -k(able(a,Ida,Day)) ;
        -k(able(b,Idb,Day)) ;
        -k(opR(Room,Day)).
%%% Cut rule
opR(A,B),opR(C,D),{{noteq(A,C)}} --> false.
opR(A,B),opR(C,D),{{noteq(B,D)}} --> false.
```

A.2 *Simple translation*

```
%%%%%----------------------------------------------------------%%%%%
            MGTP+ATMS 方式
%%%%%----------------------------------------------------------%%%%%
:- problem("meeting").
true --> goal(open).
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
goal(opR),room(Room),day(Day) --> assume(opR(Room,Day)).
goal(opR) --> goal(room).
goal(opR) --> goal(day).
goal(able),day(Day),cand(Name,Id) --> assume(able(Name,Id,Day)).
goal(able) --> goal(day).
goal(able) --> goal(cand).
opR(Room,Day),reserved(Room,Day) --> false.
true --> goal(opR).
able(a,agent,mon) --> false.
true --> goal(able).
able(b,self,tue) --> false.
true --> goal(able).
goal(open),able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day) --> open(Day,Room,Ida,Idb).
goal(open) --> goal(able).
goal(open) --> goal(able).
goal(open) --> goal(opR).
%%%%%----------------------------------------------------------%%%%%
            MGTP 連結方式
%%%%%----------------------------------------------------------%%%%%
%%%%%
%%%%% MGTP-1
%%%%%
:- problem("meeting).
true --> goal(open(4)).
true --> fact(cand(a,self),{[],32#{0}},in).
true --> fact(cand(a,agent),{[],32#{0}},in).
true --> fact(cand(b,self),{[],32#{0}},in).
true --> fact(cand(b,agent),{[],32#{0}},in).
true --> fact(room(212),{[],32#{0}},in).
true --> fact(room(213),{[],32#{0}},in).
true --> fact(day(mon),{[],32#{0}},in).
true --> fact(day(tue),{[],32#{0}},in).
true --> fact(reserved(212,mon),{[],32#{0}},in).
true --> fact(reserved(213,mon),{[],32#{0}},in).
true --> fact(reserved(213,tue),{[],32#{0}},in).
```

```
goal(opR(2)),fact(room(Room),A,in),fact(day(Day),B,in) --> fact(opR(Room,Day),C,D),
        {{& table<<=[new_assumption(opR(Room,Day),E)],
          & mgtp2<<=[cc([{[assume_opR(Room,Day)],E},A,B],C,D)]}}.
goal(opR(2)) --> goal(room(1)).
goal(opR(2)) --> goal(day(1)).
goal(able(3)),fact(day(Day),A,in),fact(cand(Name,Id),B,in) -->
        fact(able(Name,Id,Day),C,D),{{& table<<=[new_assumption(cand(Name,Id),E)],
          & mgtp2<<=[cc([{[assume_able(Name,Id,Day)],E},A,B],C,D)]}}.
goal(able(3)) --> goal(day(1)).
goal(able(3)) --> goal(cand(2)).
goal(open(4)),fact(able(a,Ida,Day),A,in),fact(able(b,Idb,Day),B,in),
        fact(opR(Room,Day),C,in) --> fact(open(Day,Room,Ida,Idb),D,E),
        {{& mgtp2<<=[cc([A,B,C],D,E)]}}.
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(opR(2)).
%%%%%
%%%%% MGTP-2
%%%%%
:- problem("meeting").
true --> goal(open(4)).
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
goal(opR(2)),room(Room),day(Day),assume_opR(Room,Day)) --> opR(Room,Day).
goal(opR(2)) --> goal(room(1)).
goal(opR(2)) --> goal(day(1)).
goal(able(3)),day(Day),cand(Name,Id),assume_able(Name,Id,Day) --> able(Name,Id,Day).
goal(able(3)) --> goal(day(1)).
goal(able(3)) --> goal(cand(2)).
true --> goal(opR(2)).
opR(Room,Day),reserved(Room,Day) --> false.
true --> goal(able(3)).
able(a,agent,mon) --> false.
true --> goal(able(3)).
able(b,self,tue) --> false.
goal(open(4)),able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day) --> open(Day,Room,Ida,Idb).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(opR(2)).

%%%%%-----------------------------------------------------------%%%%%
        Nogood 仮定方式
%%%%%-----------------------------------------------------------%%%%%
```

```
:- problem("meeting").
true --> goal(open(4)).
m(good(H1)),nogood(H2) --> false(YN,H1,H2),{{nogood_lib:superE(H2,H1,YN)}}.
m(good(H1)),m(nogood(H2)) --> false(YN,H1,H2),{{nogood_lib:superE(H2,H1,YN)}}.
false(yes,H1,H2) --> false.
true --> fact(cand(a,self),{[],32#{0}}).
true --> fact(cand(a,agent),{[],32#{0}}).
true --> fact(cand(b,self),{[],32#{0}}).
true --> fact(cand(b,agent),{[],32#{0}}).
true --> fact(room(212),{[],32#{0}}).
true --> fact(room(213),{[],32#{0}}).
true --> fact(day(mon),{[],32#{0}}).
true --> fact(day(tue),{[],32#{0}}).
true --> fact(reserved(213,tue),{[],32#{0}}).
true --> fact(reserved(212,mon),{[],32#{0}}).
true --> fact(reserved(213,mon),{[],32#{0}}).
goal(opR(2)),fact(room(Room),A),fact(day(Day),B) --> fact(opR(Room,Day),C),
        {{& table<<=[new_assumption(opR(Room,Day)D)],
          nogood_lib:union([{[opR(Room,Day)],D},A,B],C)}},
        m(good(C)) ; m(nogood(C)).
goal(opR(2)) --> goal(room(1)).
goal(opR(2)) --> goal(day(1)).
goal(able(3)),fact(day(Day),A),fact(cand(Name,Id),B) --> fact(able(Name,Id,Day),C),
        {{& table<<=[new_assumption(able(Name,Id,Day),D)],
          nogood_lib:union([{[able(Name,Id,Day)],D},A,B],C)}},
        m(good(C)) ; m(nogood(C)).
goal(able(3)) --> goal(day(1)).
goal(able(3)) --> goal(cand(2)).
true --> goal(opR(2)).
fact(opR(Room,Day),A),fact(reserved(Room,Day),B) --> nogood(C),{{nogood_lib:union([A,B],C)}}.
true --> goal(able(3)).
fact(able(a,agent,mon),A) --> nogood(B),{{nogood_lib:union([A],B)}}.
true --> goal(able(3)).
fact(able(b,self,tue),A) --> nogood(B),{{nogood_lib:union([A],B)}}.
goal(open(4)),fact(able(a,Ida,Day),A),fact(able(b,Idb,Day),B),fact(opR(Room,Day),C) -->
        fact(open(Day,Room,Ida,Idb),D),{{nogood_lib:union([A,B,C],D)}},
        m(good(D)) ; m(nogood(D)).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(opR(2)).

%%%%%------------------------------------------------------------%%%%%
            全モデル生成方式
%%%%%------------------------------------------------------------%%%%%
:- problem("meeting").
true --> goal(open(4)).
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
```

```
true --> day(mon).
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
goal(opR(2)),room(Room),day(Day) --> opR(Room,Day) ; -k(opR(Room,Day)).
goal(opR(2)) --> goal(room(1)).
goal(opR(2)) --> goal(day(1)).
goal(able(3)),day(Day),cand(Name,Id) --> able(Name,Id,Day) ; -k(able(Name,Id,Day)).
goal(able(3)) --> goal(day(1)).
goal(able(3)) --> goal(cand(2)).
true --> goal(opR(2)).
opR(Room,Day),reserved(Room,Day) --> false.
true --> goal(able(3)).
able(a,agent,mon) --> false.
true --> goal(able(3)).
able(b,self,tue) --> false.
goal(open(4)),able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day) --> open(Day,Room,Ida,Idb).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(able(3)).
goal(open(4)) --> goal(opR(2)).
%%% Cut rule
opR(A,B),opR(C,D),{{noteq(A,C)}} --> false.
opR(A,B),opR(C,D),{{noteq(B,D)}} --> false.

%%%%%----------------------------------------------------------%%%%%
            Skip 方式
%%%%%----------------------------------------------------------%%%%%
:- problem("meetingSKIP").

true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,tue).
true --> reserved(213,mon).
reserved(Room,Day) --> -k(opR(Room,Day)).
true --> -k(able(a,agent,mon)).
true --> -k(able(b,self,tue)).
goal(open(4)),cand(a,Ida),cand(b,Idb),room(Room),day(Day) -->
        able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day),open(Day,Room,Ida,Idb) ;
        -k(able(a,Ida,Day)) ;
        -k(able(b,Idb,Day)) ;
        -k(opR(Room,Day)).
goal(open(4)) --> goal(cand(2)).
goal(open(4)) --> goal(cand(2)).
goal(open(4)) --> goal(room(1)).
```

```
goal(open(4)) --> goal(day(1)).
%%% Cut rule
opR(A,B),opR(C,D),{{noteq(A,C)}} --> false.
opR(A,B),opR(C,D),{{noteq(B,D)}} --> false.
true --> goal(open(4)).
```

A.3 *Left-to-Right translation*

```
%%%%%-----------------------------------------------------------%%%%%
            MGTP+ATMS 方式
%%%%%-----------------------------------------------------------%%%%%
:- problem("meeting").
true --> goal(open($(3),$(2),$(1),$(0))).
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
goal(open(A,B,C,D)) --> goal(able(a,$(1),$(3))),rule90_0([$(0),$(1),$(2),$(3)]).
rule90_0([A,B,C,D]),able(a,E,F) --> goal(able(b,$(0),F)),rule91_1([$(0),E,$(2),F]).
rule91_1([A,B,C,D]),able(b,E,D) --> goal(opR($(2),D)),rule92_2([E,B,$(2),D]).
rule92_2([A,B,C,D]),opR(E,D) --> open(D,E,B,A).
goal(opR(A,D)) --> goal(room($(1))),rule10_3([$(0),$(1)]).
rule10_3([A,B]),room(C) --> goal(day($(0))),rule11_4([$(0),C]).
rule11_4([A,B]),day(C) --> assume(opR(B,C)).
goal(able(b,A,B)) --> goal(day($(0))),rule20_5([$(0),$(1),b]).
rule20_5([A,B,b]),day(C) --> goal(cand(b,$(1))),rule21_6([C,$(1),b]).
rule21_6([A,B,b]),cand(b,C) --> assume(able(b,C,A)).
goal(able(a,A,B)) --> goal(day($(0))),rule20_7([$(0),$(1),a]).
rule20_7([A,B,a]),day(C) --> goal(cand(a,$(1))),rule21_8([C,$(1),a]).
rule21_8([A,B,a]),cand(a,C) --> assume(able(a,C,A)).
true --> goal(opR($(1),$(0))),rule40_9([$(0),$(1)]).
rule40_9([A,B]),opR(C,D) --> goal(reserved(C,D)),rule41_10([D,C]).
rule41_10([A,B]),reserved(B,A) --> false.
true --> goal(able(a,agent,mon)),rule70_11([]).
rule70_11([]),able(a,agent,mon) --> false.
true --> goal(able(b,self,tue)),rule80_12([]).
rule80_12([]),able(b,self,tue) --> false.
goal(able(b,self,tue)) --> goal(day(tue)),rule20_13([tue,self,b]).
rule20_13([tue,self,b]),day(tue) --> goal(cand(b,self)),rule21_14([tue,self,b]).
rule21_14([tue,self,b]),cand(b,self) --> assume(able(b,self,tue)).
goal(able(a,agent,mon)) --> goal(day(mon)),rule20_15([mon,agent,a]).
rule20_15([mon,agent,a]),day(mon) --> goal(cand(a,agent)),rule21_16([mon,agent,a]).
rule21_16([mon,agent,a]),cand(a,agent) --> assume(able(a,agent,mon)).
goal(opR(A,B)) --> goal(room($(1))),rule10_17([$(0),$(1)]).
```

```
rule10_17([A,B]),room(C) --> goal(day($(0))),rule11_18([$(0),C]).
rule11_18([A,B]),day(C) --> assume(opR(B,C)).
%%%%%------------------------------------------------------%%%%%
            MGTP 連結方式
%%%%%------------------------------------------------------%%%%%
%%%%%
%%%%% MGTP-1
%%%%%
:- problem("meeting").
true --> fact(cand(a,self),{[],32#{0}},in).
true --> fact(cand(a,agent),{[],32#{0}},in).
true --> fact(cand(b,self),{[],32#{0}},in).
true --> fact(cand(b,agent),{[],32#{0}},in).
true --> fact(room(212),{[],32#{0}},in).
true --> fact(room(213),{[],32#{0}},in).
true --> fact(day(mon),{[],32#{0}},in).
true --> fact(day(tue),{[],32#{0}},in).
true --> fact(reserved(212,mon),{[],32#{0}},in).
true --> fact(reserved(213,mon),{[],32#{0}},in).
true --> fact(reserved(213,tue),{[],32#{0}},in).
goal(open(_,_,_,_)) --> goal(able(a,"Ida","Day")),
        cont(1,{[],32#{0}},{"Day","Room","Ida","Idb"},in).
cont(1,_H1,{_,_,_,_},in),fact(able(a,Ida,Day),_H2,in) --> goal(able(b,"Idb",Day)),
cont(2,_H3,{Day,"Room",Ida,"Idb"},_Io),{{& mgtp2<<=[cc(_H1,_H2,_H3,_Io)]}}.
cont(2,_H1,{Day,_,Ida,_},in),fact(able(b,Idb,Day),_H2,in) --> goal(opR("Room",Day)),
cont(3,_H3,{Day,"Room",Ida,Idb},_Io),{{& mgtp2<<=[cc(_H1,_H2,_H3,_Io)]}}.
cont(3,_H1,{Day,_,Ida,Idb},in),fact(opR(Room,Day),_H2,in) -->
        fact(open(Day,Room,Ida,Idb),_H3,_Io),{{& mgtp2<<=[cc(_H1,_H2,_H3,_Io)]}}.
goal(able(Name,_,_)) --> goal(day("Day")),cont(4,{[],32#{0}},{Name,"Id","Day"},in).
cont(4,_H1,{Name,_,_},in),fact(day(Day),_H2,in) --> goal(cand(Name,"Id")),
        cont(5,_H3,{Name,"Id",Day},_Io),{{& mgtp2<<=[cc(_H1,_H2,_H3,_Io)]}}.
cont(5,_H1,{Name,_,Day},in),fact(cand(Name,Id),_H2,in) --> fact(able(Name,Id,Day),_H3,_Io),
        {{& table<<=[new_assumption(_Na)],
          & mgtp2<<=[cc([{[assume_able(Name,Id,Day)],_Na}_H1,_H2],_H3,_Io)]}}.
goal(opR(_,Day)) --> goal(room("Room")),cont(7,{[],32#{0}},{"Room",Day},in).
cont(7,_H1,{_,Day},in),fact(room(Room),_H2,in) --> goal(day(Day)),cont(8,_H3,{Room,Day},_Io),
        {{& mgtp2<<=[cc(_H1,_H2,_H3,_Io)]}}.
cont(8,_H1,{Room,Day},in),fact(day(Day),_H2,in) --> fact(opR(Room,Day),_H3,_Io),
        {{& table<<=[new_assumption(_Na)],
          & mgtp2<<=[cc([{[assume_opR(Room,Day)],_Na},_H1,_H2],_H3,_Io)]}}}}.
%%%%%
%%%%% MGTP-2
%%%%%
:- problem("meeting").
true --> goal(open($(3),$(2),$(1),$(0))).
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
```

```
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
goal(open(A,B,C,D)) --> goal(able(a,$(1),$(3))),rule90_0([$(0),$(1),$(2),$(3)]).
rule90_0([A,B,C,D]),able(a,E,F) --> goal(able(b,$(0),F)),rule91_1([$(0),E,$(2),F]).
rule91_1([A,B,C,D]),able(b,E,D) --> goal(opR($(2),D)),rule92_2([E,B,$(2),D]).
rule92_2([A,B,C,D]),opR(E,D) --> open(D,E,B,A).
goal(opR(A,B)) --> goal(room($(1))),rule10_3([$(0),$(1)]).
rule10_3([A,B]),room(C) --> goal(day($(0))),rule11_4([$(0),C]).
rule11_4([A,B]),day(C),assume_opR(B,C) --> opR(B,C).
goal(able(b,A,B)) --> goal(day($(0))),rule20_5([$(0),$(1),b]).
rule20_5([A,B,b]),day(C) --> goal(cand(b,$(1))),rule21_6([C,$(1),b]).
rule21_6([A,B,b]),cand(b,C),assume_able(b,C,A) --> able(b,C,A).
goal(able(a,A,B)) --> goal(day($(0))),rule20_7([$(0),$(1),a]).
rule20_7([A,B,a]),day(C) --> goal(cand(a,$(1))),rule21_8([C,$(1),a]).
rule21_8([A,B,a]),cand(a,C),assume_able(a,C,A) --> able(a,C,A).
true --> goal(opR($(1),$(0))),rule40_9([$(0),$(1)]).
rule40_9([A,B]),opR(C,D) --> goal(reserved(C,D)),rule41_10([D,C]).
rule41_10([A,B]),reserved(B,A) --> false.
true --> goal(able(a,agent,mon)),rule70_11([]).
rule70_11([]),able(a,agent,mon) --> false.
true --> goal(able(b,self,tue)),rule80_12([]).
rule80_12([]),able(b,self,tue) --> false.
goal(able(b,self,tue)) --> goal(day(tue)),rule20_13([tue,self,b]).
rule20_13([tue,self,b]),day(tue) --> goal(cand(b,self)),rule21_14([tue,self,b]).
rule21_14([tue,self,b]),cand(b,self),assume_able(b,self,tue) --> able(b,self,tue).
goal(able(a,agent,mon)) --> goal(day(mon)),rule20_15([mon,agent,a]).
rule20_15([mon,agent,a]),day(mon) --> goal(cand(a,agent)),rule21_16([mon,agent,a]).
rule21_16([mon,agent,a]),cand(a,agent),assume_able(a,agent,mon) --> able(a,agent,mon).
goal(opR(A,B)) --> goal(room($(1))),rule10_17([$(0),$(1)]).
rule10_17([A,B]),room(C) --> goal(day($(0))),rule11_18([$(0),C]).
rule11_18([A,B]),day(C),assume_opR(B,C) --> opR(B,C).
%%%%%------------------------------------------------------------%%%%%
                Nogood 仮定方式
%%%%%------------------------------------------------------------%%%%%
:- problem("meeting").
true --> goal(open($(3),$(2),$(1),$(0))).
m(good(H1)),nogood(H2) --> false(YN,H1,H2),{{nogood_lib:superE(H2,H1,YN)}}.
m(good(H1)),m(nogood(H2)) --> false(YN,H1,H2),{{nogood_lib:superE(H2,H1,YN)}}.
false(yes,H1,H2) --> false.
true --> fact(cand(a,self),{[],32#{0}}).
true --> fact(cand(a,agent),{[],32#{0}}).
true --> fact(cand(b,self),{[],32#{0}}).
true --> fact(cand(b,agent),{[],32#{0}}).
true --> fact(room(212),{[],32#{0}}).
true --> fact(room(213),{[],32#{0}}).
true --> fact(day(mon),{[],32#{0}}).
true --> fact(day(tue),{[],32#{0}}).
true --> fact(reserved(213,tue),{[],32#{0}}).
true --> fact(reserved(212,mon),{[],32#{0}}).
true --> fact(reserved(213,mon),{[],32#{0}}).
```

```
goal(open(A,B,C,D)) --> goal(able(a,$(1),$(3))),rule90_0([$(0),$(1),$(2),$(3)],{[],32#{0}}).
rule90_0([A,B,C,D],E),fact(able(a,F,G),H) --> goal(able(b,$(0),G)),rule91_1([$(0),F,$(2),G],I),
        {{nogood_lib:union([E,H],I)}},m(good(i)) ; m(nogood(1)).
rule91_1([A,B,C,D],E),fact(able(b,F,D),G) --> goal(opR($(2),D)),rule92_2([F,B,$(2),D],H),
        {{nogood_lib:union([E,G],H)}},m(good(H)) ; m(nogood(H)).
rule92_2([A,B,C,D],E),fact(opR(F,D),G) --> fact(open(D,F,B,A),H),{{nogood_lib:union([E,G],H)}},
        m(good(H)) ; m(nogood(H)).
goal(opR(A,B)) --> goal(room($(1))),rule10_3([$(0),$(1)],{[],32#{0}}).
rule10_3([A,B],C),fact(room(D),E) --> goal(day($(0))),rule11_4([$(0),D],F),
        {{nogood_lib:union([C,E],F)}},m(good(F)) ; m(nogood(F)).
rule11_4([A,B],C),fact(day(D),E) --> fact(opR(B,D),F),
        {{& table<<=[new_assumption(opR(B,D),F),G)],
          nogood_lib:union([{[opR(B,D)],G},C,E],F)}},m(good(F)) ; m(nogood(F)).
goal(able(b,A,B)) --> goal(day($(0))),rule20_5([$(0),$(1),b],{[],32#{0}}).
rule20_5([A,B,b],C),fact(day(D),E) --> goal(cand(b,$(1))),rule21_6([D,$(1),b],F),
        {{nogood_lib:union([C,E],F)}},m(good(F)) ; m(nogood(F)).
rule21_6([A,B,b],C),fact(cand(b,D),E) --> fact(able(b,D,A),F),
        {{& table<<=[new_assumption(able(b,D,A),G)],
          nogood_lib:union([{[able(b,D,A)],G},C,E],F)}},m(good(F)) ; m(nogood(F)).
goal(able(a,A,B)) --> goal(day($(0))),rule20_7([$(0),$(1),a],{[],32#{0}}).
rule20_7([A,B,a],C),fact(day(D),E) --> goal(cand(a,$(1))),rule21_8([D,$(1),a],F),
        {{nogood_lib:union([C,E],F)}},m(good(F)) ; m(nogood(F)).
rule21_8([A,B,a],C),fact(cand(a,D),E) --> fact(able(a,D,A),F),
        {{& table<<=[new_assumption(able(a,D,A),G)],
          nogood_lib:union([{[able(a,D,A)],G},C,E],F)}},m(good(F)) ; m(nogood(F)).
true --> goal(opR($(1),$(0))),rule40_9([$(0),$(1)],{[],32#{0}}).
rule40_9([A,B],C),fact(opR(D,E),F) --> goal(reserved(D,E)),rule41_10([E,D],G),
        {{nogood_lib:union([C,F],G)}},m(good(G)) ; m(nogood(G)).
rule41_10([A,B],C),fact(reserved(B,A),D) --> nogood(E),{{nogood_lib:union([C,D],E)}}.
true --> goal(able(a,agent,mon)),rule70_11([],{[],32#{0}}).
rule70_11([],A),fact(able(a,agent,mon),B) --> nogood(C),{{nogood_lib:union([A,B],C)}}.
true --> goal(able(b,self,tue)),rule80_12([],{[],32#{0}}).
rule80_12([],A),fact(able(b,self,tue),B) --> nogood(C),{{nogood_lib:union([A,B],C)}}.
goal(able(b,self,tue)) --> goal(day(tue)),rule20_13([tue,self,b],{[],32#{0}}).
rule20_13([tue,self,b],A),fact(day(tue),B) --> goal(cand(b,self)),rule21_14([tue,self,b],C),
        {{nogood_lib:union([A,B],C)}},m(good(C)) ; m(nogood(C)).
rule21_14([tue,self,b],A),fact(cand(b,self),B) --> fact(able(b,self,tue),C),
        {{& table<<=[new_assumption(able(b,self,tue),D)],
          nogood_lib:union([{[able(b,self,tue)],D},A,B],C)}},m(good(C)) ; m(nogood(C)).
goal(able(a,agent,mon)) --> goal(day(mon)),rule20_15([mon,agent,a],{[],32#{0}}).
rule20_15([mon,agent,a],A),fact(day(mon),B) --> goal(cand(a,agent)),rule21_16([mon,agent,a],C),
        {{nogood_lib:union([A,B],C)}},m(good(C)) ; m(nogood(C)).
rule21_16([mon,agent,a],A),fact(cand(a,agent),B) --> fact(able(a,agent,mon),C),
        {{& table<<=[new_assumption(able(a,agent,mon),D)],
          nogood_lib:union([{[able(a,agent,mon)],D},A,B],C)}},m(good(C)) ; m(nogood(C)).
goal(opR(A,B)) --> goal(room($(1))),rule10_17([$(0),$(1)],{[],32#{0}}).
rule10_17([A,B],C),fact(room(D),E) --> goal(day($(0))),rule11_18([$(0),D],F),
        {{nogood_lib:union([C,E],F)}},m(good(F)) ; m(nogood(F)).
rule11_18([A,B],C),fact(day(D),E) --> fact(opR(B,D),F),
        {{& table<<=[new_assumption(opR(B,D),G)],
          nogood_lib:union([{[opR(B,D)],G},C,E],F)}},m(good(F)) ; m(nogood(F))
%%%%%----------------------------------------------------%%%%%
```

全モデル生成方式

```
%%%%%-----------------------------------------------------%%%%%
:- problem("meeting").
true --> goal(open($(3),$(2),$(1),$(0))).
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(213,tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
goal(open(A,B,C,D)) --> goal(able(a,$(1),$(3))),rule90_0([$(0),$(1),$(2),$(3)]).
rule90_0([A,B,C,D]),able(a,E,F) --> goal(able(b,$(0),F)),rule91_1([$(0),E,$(2),F]).
rule91_1([A,B,C,D]),able(b,E,D) --> goal(opR($(2),D)),rule92_2([E,B,$(2),D]).
rule92_2([A,B,C,D]),opR(E,D) --> open(D,E,B,A).
goal(opR(A,B)) --> goal(room($(1))),rule10_3([$(0),$(1)]).
rule10_3([A,B]),room(C) --> goal(day($(0))),rule11_4([$(0),C]).
rule11_4([A,B]),day(C) --> opR(B,C) ; -k(opR(B,C)).
goal(able(b,A,B)) --> goal(day($(0))),rule20_5([$(0),$(1),b]).
rule20_5([A,B,b]),day(C) --> goal(cand(b,$(1))),rule21_6([C,$(1),b]).
rule21_6([A,B,b]),cand(b,C) --> able(b,C,A) ; -k(able(b,C,A)).
goal(able(a,A,B)) --> goal(day($(0))),rule20_7([$(0),$(1),a]).
rule20_7([A,B,a]),day(C) --> goal(cand(a,$(1))),rule21_8([C,$(1),a]).
rule21_8([A,B,a]),cand(a,C) --> able(a,C,A) ; -k(able(a,C,A)).
ture --> goal(opR($(1),$(0))),rule40_9([$(0),$(1)]).
rule40_9([A,B]),opR(C,D) --> goal(reserved(C,D)),rule41_10([D,C]).
rule41_10([A,B]),reserved(B,A) --> false.
ture --> goal(able(a,agent,mon)),rule70_11([]).
rule70_11([]),able(a,agent,mon) --> false.
true --> goal(able(b,self,tue)),rule80_12([]).
rule80_12([]),able(b,self,tue) --> false.
goal(able(b,self,tue)) --> goal(day(tue)),rule20_13([tue,self,b]).
rule20_13([tue,self,b]),day(tue) --> goal(cand(b,self)),rule21_14([tue,self,b]).
rule21_14([tue,self,b]),cand(b,self) --> able(b,self,tue) ; -k(able(b,self,tue)).
goal(able(a,agent,mon)) --> goal(day(mon)),rule20_15([mon,agent,a]).
rule20_15([mon,agent,a]),day(mon) --> goal(cand(a,agent)),rule21_16([mon,agent,a]).
rule21_16([mon,agent,a]),cand(a,agent) --> able(a,agent,mon) ; -k(able(a,agent,mon)).
goal(opR(A,B)) --> goal(room($(1))),rule10_17([$(0),$(1)]).
rule10_17([A,B]),room(C) --> goal(day($(0))),rule11_18([$(0),C]).
rule11_18([A,B]),day(C) --> opR(B,C) ; -k(opR(B,C)).
%%%%%-----------------------------------------------------%%%%%
```

Skip 方式

```
%%%%%-----------------------------------------------------%%%%%
:- problem("meeting").
true --> cand(a,self).
true --> cand(a,agent).
true --> cand(b,self).
true --> cand(b,agent).
```

```
true --> room(212).
true --> room(213).
true --> day(mon).
true --> day(tue).
true --> reserved(212,mon).
true --> reserved(213,mon).
true --> reserved(213,tue).
true --> -k(able(a,agent,mon)).
true --> -k(able(b,self,tue)).
true --> goal(reserved("R","D")).
reserved(R,D) --> -k(opR(R,D)).
goal(open(_,_,_,_)) --> goal(cand(a,"Ida")),cont(1,{a,"Ida"}).
cont(1,{A,_}),cand(A,Ida) --> goal(cand(b,"Idb")),cont(2,{A,Ida,b,"Idb"}).
cont(2,{A,Ida,B,_}),cand(B,Idb) --> goal(room("Room")),cont(3,{A,Ida,B,Idb,"Room"}).
cont(3,{A,Ida,B,Idb,_}),room(Room) --> goal("Day"),cont(4,{A,Ida,B,Idb,Room,"Day"}).
cont(4,{A,Ida,B,Idb,Room,_}),day(Day) -->
        able(a,Ida,Day),able(b,Idb,Day),opR(Room,Day),open(Day,Room,Ida,Idb) ;
        -k(able(a,Ida,Day)) ;
        -k(able(b,Idb,Day)) ;
        -k(opR(Room,Day)).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```