知 能 と 複 雑 系　126－2
（２００１．１０．１２）

非決定性アクション言語$\mathcal{NA}$上のプランニング手続き

奥澤 望[1]　鍋島 英知[2]　井上 克巳[1,3]　羽根田 博正[1,3]

[1]神戸大学大学院自然科学研究科
[2]山梨大学工学部コンピュータ・メディア工学科
[3]神戸大学工学部電気電子工学科

[1]〒 657-8501 神戸市灘区六甲台町 1-1
神戸大学大学院 自然科学研究科 電気電子工学専攻
e-mail: okuzawa@s2.eedept.kobe-u.ac.jp

状態の変化を表現する最近の研究において、高級レベルのアクション言語が使用されている．それは，自然言語表現を用いてアクションによる状態の変化を記述する形式的モデルであるといえる．本研究では言語$\mathcal{NA}$上でプランニングの問い合わせに対し、効率良くプランニングを行う処理系を構築することを目的とし，このためのプランニングアルゴリズムを提案する．このため，アルゴリズムにおいて用いる言語$\mathcal{NA}$の領域記述を言語$\mathcal{A}$の領域記述に翻訳できる手続きを提案する．またこの翻訳手続きの正当性も証明する．

非決定性, アクション言語$\mathcal{NA}$, プランニング

Planning in Nondeterministic Action Language $\mathcal{NA}$

Nozomu Okuzawa[*]　Hidetomo Nabeshima[**]
Katsumi Inoue[*,***]　Hiromasa Haneda[*,***]

[*]Graduate School of Science and Technology, Kobe University
[**]Department of Computer Science and
Media Engineering, Yamanashi University
[***]Department of Electrical and Electronics Engineering,
Faculty of Engineering, Kobe University

[*]Division of Electrical and Electoronics Engineering,
Graduate School of Science and Technology, Kobe University
Rokkodai, Nada Kobe 657-8501 Japan
e-mail: okuzawa@s2.eedept.kobe-u.ac.jp

Recent work on representing action has introduced high-level *action languages* to describe effects of actions in a systematic and theoretically sound way. In this paper, we define a new planning algorithm in the nondeterministic action language $\mathcal{NA}$ which is an action language for describing nondeterministic actions. We can find credulous plans in domains of $\mathcal{NA}$ by using this algorithm. To this end, we define an algorithm to transfer $\mathcal{NA}$ to the deterministic action language $\mathcal{A}$. And, we prove this algorithm.

Nondeterministic, Action language $\mathcal{NA}$, planning

1 はじめに

状態の変化とアクションを表現する問題は，人工知能研究の初期の段階から議論されている問題の一つである．その分野で最近活発に研究されているアプローチの 1 つに**アクション言語**がある．アクション言語とは，簡易的な自然言語表現を用いてアクションによる状態の変化を宣言的に記述する形式的モデルである．**アクション言語** $\mathcal{A}$ はアクション言語のなかで最初に提案された言語であり，Gelfond と Lifschitz [1] により提案されている．言語 $\mathcal{A}$ では決定性効果をもつアクションのみ記述することができる．言語 $\mathcal{A}$ を基として，さらなる表現力を求めて様々な非決定性効果を持つアクション言語 [2, 3, 4] やフルーエント間の制約をもつアクション言語 [4] が提案されている．本研究で対象となる**非決定性アクション言語** $\mathcal{NA}$ [6] もその一つで，非決定性効果をもつアクションとフルーエント間の制約を記述することができる．

本研究では言語 $\mathcal{NA}$ 上でプランニングの問い合わせに対し、効率良くプランニングを行う処理系を構築することを目的とする．現在，言語 $\mathcal{A}$ におけるプランニングの問い合わせに対し，高速に応答する**アクション言語処理系** AMP [7] がある．今回提案するプランニングアルゴリズムは AMP を利用することで効率良くプランニングを行える．アルゴリズムは，まず**軽信的 (credulous)** なモデルに対して言語 $\mathcal{NA}$ の領域記述を言語 $\mathcal{A}$ の領域記述に翻訳できる手続き **NAtoA** により言語 $\mathcal{A}$ の領域記述に翻訳し，**NAtoA** により翻訳された領域記述に対し AMP を用いてプランニングを行う．最後に求まったプランを元の領域記述に戻す．これにより言語 $\mathcal{NA}$ において目標状態に到達する可能性のあるプランを求めることができる．また翻訳手続き **NAtoA** の正当性を証明している．

2 アクション言語 $\mathcal{A}$

2.1 構文論

言語 $\mathcal{A}$ においてアクション領域の記述は評価命題と効果命題という 2 種類の命題から構成されている．言語 $\mathcal{A}$ において，**フルーエント** (fluent) とは状態に依存する属性のことであり，状態の変化に伴いフルーエントのとる値も変化する．ここで，フルーエントは真または偽の 2 値をとる．フルーエント名 F の否定 $\neg F$ を**負のフルーエント**といい，F を**正のフルーエント**という．普通，フルーエントを小文字の f で表し，フルーエント名を大文字の F で表す．フルーエントを，論理演算子 $(\vee, \wedge, \neg, \supset, \equiv)$ で結合したものを**式**と呼ぶ．**効果命題**は，次のような形式をとり，アクションとその因果関係を記述する：

$$a \ \textbf{causes} \ \phi \ \textbf{if} \ \psi \tag{2.1}$$

ここで a はアクション名，ϕ はフルーエントの連言，ψ はフルーエントを論理演算子 $(\vee, \wedge, \neg, \supset, \equiv)$ で結合した式である．ϕ をアクションの効果と呼び，ψ をアクションの前提条件と呼ぶ．特に，前提条件が $true$ であるならば効果命題は **if** 以下を省略して次のように記述する：

$$a \ \textbf{causes} \ \phi$$

ここで，$true$ は恒真を意味する特別なフルーエントで，$false = \neg true$ である．**評価命題**は，観測した事実を記述する：

$$\phi \ \textbf{after} \ \ a_1; ...; a_m. \quad (m \geq 0) \tag{2.2}$$

ここで，ϕ はフルーエントの連言，a_i はアクション名である．特に，$m = 0$ のときは次のように記述する：

$$\textbf{initially} \ \phi$$

2.2 意味論

状態 q を，全てのフルーエント名について，正または負のフルーエントのいずれかを含む集合と定義する．

$$q = S \cup \{\neg F \mid F \in \mathcal{F} \setminus \mathcal{S}\}$$

ここで $\mathcal{F}$ は領域記述で利用するすべてのフルーエント名の集合であり，$S \subseteq \mathcal{F}$ である．

任意の状態 q と，フルーエント f のみからなる式 $\phi = f$ に対し，$f \subseteq q$ であるときに限り，**状態 q で式 ϕ が真**であると定義する．状態 q で式 ϕ が真であるならば， $q \models \phi$ と記述する．**遷移関数**はアクション a と状態 σ の対 (σ, a) のすべての集合から状態の集合への写像 Φ である．**解釈**は初期状態 σ_0 と遷移関数 Φ の対 (Φ, σ_0) である．任意の解釈 I についても $I^{a_1;...;a_m}$ により状態を次のように示す．

$$\Phi(a_m, \Phi(a_{m-1}, ..., \Phi(a_1, \sigma_0)...))$$

評価命題に関して，状態 $I^{a_1;...;a_m}$ において ϕ が真であれば評価命題が解釈 I において**真**であるという．

解釈 I (Φ, σ_0) が領域記述の**モデル**であるのは，(i) D に含まれる全ての評価命題が I において真であり，(ii) すべてのアクション名 a とすべてのフルーエント f，すべての状態 σ に対し，以下の条件を満たす場合である：

(a) D が状態 σ で前提条件を満たす式 (2.1) の形の式の効果命題を含めば，$f \in \Phi(A, \sigma)$ である，

(b) D が，そのような効果命題を含まなければ，$f \in \sigma$ と $f \in \Phi(A, \sigma)$ は同値である．

評価命題が領域記述 Dの任意のモデルで真であるならば次のように記述する．

$$D \models (\phi \ \textbf{after} \ \ a_1; ...; a_m)$$

2.3 アクション言語処理系 AMP

AMP [7] は**プランニンググラフ** と **SAT ソルバ**による高速なプランニングアプローチに基づいたアクション言語処理系である．AMP は入力として言語 $\mathcal{A}$ の領域記述を受け取り，様々な問い合わせに高速に応答することができる．AMP は次の 4 つの機能をもつ．

$$Check\ if \qquad D \models \phi \ \textbf{after}\ a_1;\ldots;a_m \tag{2.3}$$

$$Find\ X\ s.t. \qquad D \models X \ \textbf{after}\ a_1;\ldots;a_m \tag{2.4}$$

$$Find\ X\ s.t. \qquad D \models \psi \ \textbf{after}\ X \tag{2.5}$$

$$Find\ X\ s.t. \qquad D \models \textbf{initially}\ X \tag{2.6}$$

式 (2.3) は，アクション列 $a_1;\ldots;a_n$ を実行後の状態において式 ϕ が成立するかどうかいう問い合わせである．この形式の問い合わせを**シミュレーション**と呼ぶ．式 (2.4) は，アクション列 $a_1;\ldots;a_n$ を実行後，どのようなフルーエントが成立するかという問い合わせである．この形式の問い合わせを**実行結果の予測**と呼ぶ．式 (2.5) は，目標条件 ϕ を成立させるアクション列 X の問い合わせ，すなわち**プランニング**である．式 (2.6) は，初期状態で成立するフルーエントの問い合わせである．これは，領域記述 D のモデル $M=(\Phi,\sigma_0)$ の初期状態 σ_0 を求めることに等しい．遷移関数 Φ は効果命題から簡単に求めることができるので，AMP では式 (2.6) の問い合わせを**モデル生成**と呼ぶ．

3 アクション言語 $\mathcal{NA}$

3.1 構文論

アクション言語 $\mathcal{NA}$ では 3 種類の命題 (効果命題，制約，評価命題) を記述できる．フルーエントは言語 $\mathcal{A}$ と同様に定義する．まず，アクションとその効果を記述する**効果命題**がある：

$$a\ \textbf{causes}\ \phi_1|\cdots|\phi_n\ \textbf{if}\ \psi$$

ここで a はアクション名，各 ϕ_i はフルーエントの連言，ψ は式である．各 ϕ_i をアクションの**効果**と呼び，ψ をアクションの**前提条件**と呼ぶ．$n=1$ のときアクションの効果は**決定的**であり，$n\geq 2$ のとき**非決定的**である．次に，状態に対する**制約**が記述できる：

$$\textbf{always}\ \phi$$

ここでは ϕ は式である．この命題は，あらゆる状態において，式 ϕ が常に成立することを言明する．最後に，観測した事実を記述する**評価命題**がある：

$$\phi\ \textbf{after}\ \vec{a}$$

ここで ϕ は式，$\vec{a}$ はアクション列である．この命題は，アクション列 $\vec{a}$ を実行した後，ϕ が成立することを言明している．また評価命題は正則表現 r を用いて次のように記述することも出来る：

$$\phi\ \textbf{after}\ r$$

この命題は，$L(r)$ に含まれる任意のアクション列 $\vec{a}$ を実行した後，ϕ が成立することを言明している．

これらは言語 $\mathcal{A}$ と同じように省略形を持つ．また a **causes** $false$ **if** ψ の形をした効果命題に関しては次のように省略できる：

$$\textbf{impossible}\ a\ \textbf{if}\ \psi$$

3.2 意味論

言語 $\mathcal{NA}$ において**領域記述**は，効果命題，制約，評価命題から構成される．その**解釈**を**非決定性状態遷移システム** $\langle Q,A,\Phi,q_0\rangle$ で与える．ここで Q は状態の集合，A はアクション名の集合，Φ は $Q\times A$ から 2^Q への**遷移関数**であり，q_0 は初期状態を表す．

任意の状態 q と制約 **always** ϕ に対し，$q\models\phi$ であるとき，かつそのときにかぎり，**状態 q で制約 always ϕ が真**であると定義し，$q\models$ **always** ϕ と記述する．状態 q が，領域記述に含まれる全ての制約を充足するならば，それを**正当な状態**と呼ぶ．

正当な遷移関数は次式で定義される．ここにおいて σ は任意の状態，a はアクションである．

$$\Phi(\sigma,a)=\{\sigma'\subseteq Q \mid \sigma'=(\sigma\backslash(|e|^+\cup|e|^-))\cup e,\ e\in effect(\mathcal{P})\}$$

ここにおいて $e\in effect(\mathcal{P})$ はアクションの結果であり，$\mathcal{P}=\{P\in D \mid P=(a\ \textbf{causes}\ \phi_1|\cdots|\phi_n\ \textbf{if}\ \psi)$ かつ $\sigma\models\psi\}$ である効果命題における $\phi_1,\ldots,\phi_n$ の各連言肢の和集合である．さらに関数 $effect$ は効果命題の集合 $\{P_1,\ldots,P_m\}$ に対して次のように拡張される：

$$effect(\{P_1,\ldots,P_m\})=$$
$$\{e_1\cup\cdots\cup e_m \mid e_1\in effect(P_1),\ldots,e_m\in effect(P_m)\}$$

正当な遷移関数において，アクションの結果 $e\in effect(\mathcal{P})$ に含まれないフルーエントは，遷移前の値を遷移後も持続する定義となっている．これは**慣性の法則**を表している．

モデルを定義するために補助的な定義を与える．任意の解釈 $M=\langle Q,A,\Phi,q_0\rangle$ と任意の式 ϕ に対し，$M(\phi)=\langle Q,A,\Phi,q_0,Q(\phi)\rangle$ と定義する．ここで $Q(\phi)=\{q\in Q \mid q\models\phi\}$ である．もし状態数が有限であるならば，$M(\phi)$ は，まさに**有限オートマトン**である．すなわち $Q(\phi)$ は，目標状態の集合を表している．オートマトンと同様に，$M(\phi)$ により受理される言語 $\mathcal{L}(\mathcal{M}(\phi))$ を以下で定義する．

$$\mathcal{L}(M(\phi))=\{w\in\mathcal{A} \mid \Phi(q_0,w)\cap Q(\phi)\neq\emptyset\}$$

さらに，$M(\phi)$ により確実に受理される言語 $L(M(\phi))$ を以下で定義する．

$$L(M(\phi))=\{w\in\mathcal{A} \mid \Phi(q_0,w)\subseteq Q(\phi)\}$$

$\mathcal{L}(M(\phi))$ は，目標状態に到達する可能性を持っている全てのアクション列の集合を表している．一

方 $L(M(\phi))$ は，必ず目標状態に到達するような全てのアクション列の集合を表している．

次に，評価命題の真偽値を定義する．任意の解釈 $M = \langle Q, A, \Phi, q_0 \rangle$ と，任意の評価命題 $P = (\phi \textbf{ after } r)$ に対し，$L(r) \subseteq L(M(\phi)$ であるとき，かつそのときに限り，M において P が**真である**と定義し，$M \models P$ と記述する．同様に，もし $L(r) \subseteq \mathcal{L}(M(\phi))$ であるならば，かつそのときに限り，M において P が**軽信的 (credulous) に真である**と定義し，$M \models\!\!\sim P$ と記述する．特に比較するとき，$M \models P$ を**懐疑的 (skeptical) に真である**ということもある．

言語 $\mathcal{NA}$ による領域記述を Dとする．解釈 $M = \langle Q, A, \Phi, q_0 \rangle$ が Dの**モデル**であるとは，

1. Q が，全ての正当な状態の集合であり，
2. Φが，Q 上に定義された正当な遷移関数であり，
3. D に含まれる全ての評価命題が M において真であるとき，

かつそのときに限る．評価命題 P が Dの任意のモデルに対して真であるならば，$D \models P$ と書く．同様に，解釈 $M = \langle Q, A, \Phi, q_0 \rangle$ が Dの**軽信的なモデル**であるのは1．2．に加え

3' Dに含まれる全ての評価命題が Mにおいて軽信的に真であるとき

かつそのときに限る．このとき $D \models\!\!\sim P$ と書く．

4 翻訳手続き NAtoA

翻訳手続き **NAtoA** は、言語 $\mathcal{NA}$ による領域記述を受け取り、言語 $\mathcal{A}$ による領域記述を出力する．**NAtoA** は2つの手続き **Determine**，**Release** からなる．**Determine** は非決定的な効果をもつ効果命題の効果それぞれを決定的な効果命題に書き換える手続きである．**Release** は制約を効果命題の効果に組み込むことで制約文を消す手続きである．翻訳手続き **NAtoA** は以下で定義される．

$$\textbf{NAtoA}(D) = \textbf{Release}(\textbf{Determine}(D))$$

4.1 手続き Determine

Determine は言語 $\mathcal{NA}$ の領域記述において非決定的な効果をもつ効果命題 ($n > 1$ である効果命題) を決定的な効果命題にする手続きである．以下の形式をした効果命題と，評価命題に対し，**Determine** は次の順序で行われる．ここで $n > 1$ である．

$$a \textbf{ causes } \phi_1 \,|\, \cdots \,|\, \phi_n \textbf{ if } \xi \tag{4.7}$$
$$\phi \textbf{ after } \vec{a} \tag{4.8}$$

1. $n > 1$である各効果命題 (4.7) に対し，アクションの効果に含まれるフルーエントの連言 $\phi_1, \ldots, \phi_n$ をそれぞれ効果とし，アクションを a_i $(i = 1, \cdots, n)$ とした効果命題の組：
$$\begin{aligned} &a_1 \textbf{ causes } \phi_1 \textbf{ if } \xi \\ &\quad\vdots \\ &a_n \textbf{ causes } \phi_n \textbf{ if } \xi \end{aligned} \tag{4.9}$$
を作り (4.7) と置き換える．
2. 領域記述においてアクション a に関する効果命題が複数存在するとき，これらの効果命題もアクションを a_i $(i = 1, \cdots, n)$ として新しく効果命題を作る．今，(4.7) を除く任意のアクション a に関する効果命題：
$$a \textbf{ causes } \phi^* \textbf{ if } \xi^* \tag{4.10}$$
に対し，アクションを a_i $(i = 1, \cdots, n)$ で置き換え，アクションの効果と前提条件は上と同様にして，新しく効果命題の組：
$$\begin{aligned} &a_1 \textbf{ causes } \phi^* \textbf{ if } \xi^* \\ &\quad\vdots \\ &a_n \textbf{ causes } \phi^* \textbf{ if } \xi^* \end{aligned} \tag{4.11}$$
を作り (4.10) と置き換える．
3. 効果命題 (4.7) から新しく作った効果命題のアクション$a_1, \ldots, a_n$に対し，アクションa_hを$a_1, \ldots, a_n$の中のどれか1つのアクションとする．効果命題 (4.7) のアクション a を含むアクション列を $\vec{a}$ とする．またアクション列 $\vec{a}$ の中のアクション a を a_h に置き換えたアクション列を $\vec{a}_h$ とする．評価命題 (4.8) に対し，アクション列 $\vec{a}$ をアクション列 $\vec{a}_h$ に置き換える．
$$\phi \textbf{ after } \vec{a}_h \tag{4.12}$$
ここで評価命題 Vに対する **Determine** の手続きを **Determine**(V) と定義する．

4.2 手続き Release

領域記述 D に含まれる以下のような形式の効果命題，制約，評価命題：

$$a \textbf{ causes } \phi \textbf{ if } \xi \tag{4.13}$$
$$\textbf{always } \Omega \tag{4.14}$$
$$\phi \textbf{ after } \vec{a} \tag{4.15}$$

に対して，翻訳手続き **Release** を行う．ここで次の関数を定義する．

- **Name**(ϕ)： 式 ϕ に含まれるフルーエント名を取り出す．ここで $f_1, \cdots, f_m$は式 ϕ に含まれるフルーエントである．$\textbf{Name}(\phi) = \{|f_1|^+, \ldots, |f_m|^+\}$
- **Size**(ω)： 集合 ω に含まれる要素の数を取り出す．

翻訳手続き **Release** は次の順序で行われる．

1. 全ての制約 $\Omega_1, \ldots, \Omega_n$ を連結し$\Psi = \Omega_1 \wedge \cdots \wedge \Omega_n$とし，1つの制約 **always** Ψとする．

2. Ψを **DNF**(選言標準形) に変換し，各選言肢を $\psi_1,\ldots,\psi_n$ とする．

$$\Psi = \psi_1 \vee \cdots \vee \psi_n$$

3. 効果命題 (4.13) の効果 ϕ に対し，Ψ の中で ϕ に含まれるフルーエント名と等しいものが存在する連言 ψ_i $(i = 1,\cdots,n)$ のうち ϕ と無矛盾な ψ_i を $\psi'_1,\ldots,\psi'_k$ とする．つまり，$\mathbf{Name}(\phi) \cap \mathbf{Name}(\Psi) \neq \emptyset$ かつ $\phi \wedge \psi'_i$ が無矛盾であり，ある j $(1 \leq j \leq n)$ に対して $\psi'_i = \psi_j$ $(i = 1,\ldots,k)$ である．各無矛盾な式 ψ'_i に対しアクションを a_i，効果を $\phi \wedge \psi'_i$ とした効果命題を新しく作る．

$$\begin{array}{l} a_1 \ \ \mathbf{causes} \ \ \phi \wedge \psi'_1 \ \ \mathbf{if} \ \ \xi \\ \quad\vdots \\ a_k \ \ \mathbf{causes} \ \ \phi \wedge \psi'_k \ \ \mathbf{if} \ \ \xi \end{array} \qquad (4.16)$$

ここで効果命題の数を減らすための戦略として次の手続きを行う．各式 ψ'_i に含まれるフルーエント名から ϕ に含まれるフルーエント名を引いたフルーエント名の個数を m とする．つまり，$\mathbf{Size}(\mathbf{Name}(\phi'_i)) - \mathbf{Size}(\mathbf{Name}(\phi)) = m$ である．$k = 2^m$のとき，つまりアクション a に対して新しく作った効果命題の数が 2^mであるとき m 個のフルーエントの真偽値は，式ϕにより全て満たされる．よってこのとき置き換える必要はないので，新しく作った効果命題の集合 (4.16) を消去する．$k \neq 2^m$ であるときは m 個のフルーエントの真偽値は，式ϕにより全て満たされないので，効果命題 (4.13) と新しく作った効果命題の集合 (4.16) を置き換える．

4. 領域記述においてアクション a に関する効果命題が複数存在するとき，これらの効果命題もアクションを a_i $(i = 1,\cdots,k)$ として新しく効果命題を作る．今，(4.13) を除く任意のアクション a に関する効果命題 :

$$a \ \ \mathbf{causes} \ \ \phi^* \ \ \mathbf{if} \ \ \xi^* \qquad (4.17)$$

に対し，アクションを a_i $(i = 1,\cdots,k)$ で置き換え，アクションの効果と前提条件は上と同様にして，新しく効果命題の組 :

$$\begin{array}{l} a_1 \ \ \mathbf{causes} \ \ \phi^* \ \ \mathbf{if} \ \ \xi^* \\ \quad\vdots \\ a_k \ \ \mathbf{causes} \ \ \phi^* \ \ \mathbf{if} \ \ \xi^* \end{array} \qquad (4.18)$$

を作り，(4.17) と置き換える．

5. 効果命題 (4.13) から新しく作った効果命題のアクション $a_1,\ldots,a_k$ に対し，アクション a_g を $a_1,\ldots,a_k$ の中どれか 1 つのアクションとする．効果命題 (4.13) のアクション a を含むアクション列を $\vec{a}$ とする．またアクション列 $\vec{a}$ の中のアクション a を a_g に置き換えたアクション列を $\vec{a}_g$ とする．評価命題 (4.15) に対し，アクション列 $\vec{a}$ をアクション列 $\vec{a}_g$ に置き換える．

$$\phi \ \ \mathbf{after} \ \ \vec{a}_g$$

ここで評価命題 Vに対する **Release** の手続きを **Release**(V) と定義する．

4.3 NAtoA の正当性

NAtoA により翻訳されたの領域記述を元の領域記述に戻す逆変換手続きを $\mathbf{NAtoA^{-1}}$とする．$\mathbf{NAtoA^{-1}}$は以下のような形式の評価命題 :

$$\phi \ \ \mathbf{after} \ \ \vec{a_k} \qquad (4.19)$$

におけるアクション列 $\vec{a}_h$ をもとのアクション列 $\vec{a}$ に置き換える手続きである．また評価命題 Vに対するの $\mathbf{NAtoA^{-1}}$の手続きを $\mathbf{NAtoA^{-1}}$(V) とする．

定理 1 Dを領域記述，V,V'を評価命題とする．このとき以下が成立する．

1. $D \vdash\sim V$ ならば，$\mathbf{NAtoA^{-1}}(V') = V$ となる V' が存在し，$\mathbf{NAtoA}(D) \models V'$ である
2. $\mathbf{NAtoA}(D) \models V'$ ならば $\mathbf{NAtoA^{-1}}(V') = V$ となる Vに対し， $D \vdash\sim V$である．

定理 1 を証明することにより，**NAtoA** の正当性は証明できる．証明は [8] を参照のこと．

5 プランニング

言語 $\mathcal{NA}$ 上のプランニングは次の手順で行われる．

1. 翻訳手続き **NAtoA** を用いて言語 $\mathcal{NA}$ の領域記述を言語 $\mathcal{A}$ の領域記述に変換する．
2. アクション言語処理系 AMP により言語 $\mathcal{A}$ のプランニングを行う．
3. 求まったプランは言語 $\mathcal{A}$ による記述となっているので **NAtoA** の逆変換手続き $\mathbf{NAtoA^{-1}}$を用いて言語 $\mathcal{NA}$ のプランにする．

5.1 Mary's hat

この例題は，帽子を被った Mary が湖に飛び込む例題である [5]．ここで，*hat*,*lake*,*wet* は，それぞれ，帽子を被っている，湖の中にいる，濡れているを表すフルーエントである．アクション *jump* は，帽子が飛んで行くかもしれないという非決定的効果と，湖に飛び込むと濡れる (*lake* $\supset$ *wet*) という間接的効果をもっている．アクション *getout*,*puton* は，それぞれ，湖から出る，帽子を被るを表している．この例題のモデルを図 1 に示す :

$$\begin{array}{l} jump \ \ \mathbf{causes} \ \ lake \\ jump \ \ \mathbf{causes} \ \ hat \mid \neg hat \ \ \mathbf{if} \ \ hat \end{array}$$

getout **causes** $\neg$*lake*
puton **causes** *hat*
impossible *jump* **if** *lake*
impossible *getout* **if** $\neg$*lake*
impossible *puton* **if** *hat*
always *lake* $\supset$ *wet*
initially $\neg$*lake* $\wedge$ $\neg$*wet* $\wedge$ *hat*

ここで次式を満たすプラン Xを求めてみる：

$D_{\mathcal{NA}} \models$ *wet* $\wedge$ $\neg$*lake* $\wedge$ $\neg$*hat* **after** X

まず手続き **Determine** により非決定的なアクションを含む効果命題を決定的なアクションの効果命題に書き換える：

*jump*1 **causes** *lake*
*jump*2 **causes** *lake*
*jump*1 **causes** *hat* **if** *hat*
*jump*2 **causes** $\neg$*hat* **if** *hat*

次に手続き **Release** により制約を効果命題に組み込む

*jump*1 **causes** *lake* $\wedge$ *wet*
*jump*2 **causes** *lake* $\wedge$ *wet*

書き換えられた領域記述は次のようになる

*jump*1 **causes** *lake* $\wedge$ *wet*
*jump*2 **causes** *lake* $\wedge$ *wet*
*jump*1 **causes** *hat* **if** *hat*
*jump*2 **causes** $\neg$*hat* **if** *hat*

書き換えられた領域記述を AMP の入力をしてプランニングを実行すると，出力として次のプランが求まる．

$\neg$*lake* $\wedge$ $\neg$*hat* **after** *jump*1; *getout*

これを言語 $\mathcal{NA}$ の記述に戻すために $\mathbf{NAtoA^{-1}}(V)$ とすると，次のようなプランが求まる．

wet $\wedge$ $\neg$*lake* $\wedge$ $\neg$*hat* **after** *jump*; *getout*

6 おわりに

本論文では非決定性アクション言語 $\mathcal{NA}$ 上でのプランニングアルゴリズムを提案した．これにより非決定的な効果を持つアクションを含む領域に対して軽信的なプラン (目標状態の到達する可能性があるプラン) を求めることができた．また AMP のプランニングアルゴリズムより考えると，入力として受け取る領域記述に含まれる効果命題の数が少ない方が効率良くプランニングを行うことができる．今回提案したアルゴリズムでは前章の例題 (Mary's hat) について考えてみると効果命題の数がほとんど増えていない．このことより，今回提案した言語 $\mathcal{NA}$ 上のプランニングアルゴリズムはプランニングを効率良く行っていると考えられる．今後の課題としては，今回提案したプランニングアルゴリズムは軽信的 (credulous) なプラン，つまり目標状態に到達する可能性のあるプランしか求めることはできない．このため懐疑的 (skeptical) なプラン，つまり必ず目標状態に到達するプランを求めるプランニングアルゴリズムを提案する必要があると考えられる．

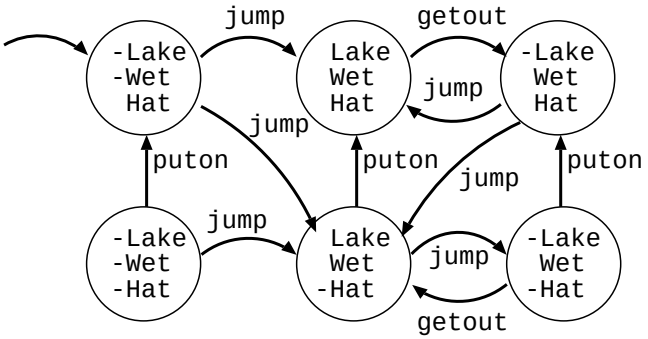

図 1 ：例題 Mary's hat のモデル

参考文献

[1] Gelfond, M. and Lifschitz, V.: Representing action and change by logic programs, *Journal of Logic Programming*, Vol.17, Nos.2–4, pp.301–321 (1993).

[2] Bornscheuer, S.–E. and Thielscher, M.: Explicit and Implicit Indeterminism: Reasoning about Uncertain and Contradictory Specifications of Dynamic Systems, *Journal of Logic Programming*, Vol.31, No.1–3, pp.119–155 (1997).

[3] Boutilier, C. and Friedman, N.: Nondeterministic Actions and the Frame Problem, *AAAI Spring Symposium Series* (1995).

[4] Giunchiglia, E., Kartha, G. N. and Lifschitz, V.: Representing Action: Indeterminacy and Ramifications,*Artificial Intelligence*, Vol.95, pp.409–443 (1997).

[5] Giunchiglia, E. and Lifschitz, V.: Dependent Fluents, *Proceedings of IJCAI–95*, pp.1964–1969 (1995).

[6] 鍋島 英知，井上 克巳， 羽根田 博正: 有限オートマトンに基づく非決定性アクション言語，情報処理学会論文誌，vol.40, No.10, pp.3661–3671 (1999).

[7] Nabeshima, H., Inoue, K. and Haneda, H.: Implementing an Action Language Using a SAT Solver, *Proceedings of ICTAI–2000*, pp.96–103 (2000).

[8] 奥澤 望: 非決定性アクション言語 $\mathcal{NA}$ 上のプランニング実装に関する研究，神戸大学工学部電気電子工学科，卒業論文 (2001).