有限オートマトンに基づく非決定性アクション言語

鍋島英知[†] 井上克巳[††] 羽根田博正[††]

状態の変化やアクション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といえる．本論文が対象とするのは，非決定性アクション言語——非決定性効果を持つアクションを記述できる言語である．本研究の目的は，1) 非決定性有限オートマトン（NFA）の観点から新しい非決定性アクション言語 $\mathcal{NA}$ を提案し，2) すでに提案されている非決定性アクション言語の表現能力を形式的に解析する手段を提供することにある．実際に本論文では，言語 $\mathcal{NA}$ と有限オートマトンの表現能力が等しいことを示し，さらに Giunchiglia らによる言語 $\mathcal{AR}$ と言語 $\mathcal{NA}$ との等価性を証明する．

A Nondeterministic Action Language Based on Finite Automata

HIDETOMO NABESHIMA,[†] KATSUMI INOUE[††]
and HIROMASA HANEDA[††]

Recent work on representing action has introduced high-level *action languages* to describe the effects of actions in a systematic and theoretically sound way. In this paper, we define a *nondeterministic action language*, which is an action language for describing nondeterministic actions , and study its properties. We first define a new nondeterministic action language $\mathcal{NA}$ from a viewpoint of *nondeterministic finite automata* (NFA). Then, we provide a formal way to analyze other action languages that have been already proposed in terms of their applicability. In fact, we prove in this paper (a) the equivalence between the expressive power of $\mathcal{NA}$ and that of the class of finite automata, (b) the equivalence between $\mathcal{NA}$ and $\mathcal{AR}$ proposed by Giunchiglia et al.

1. はじめに

状態の変化やアクション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といえる．そのようなアクション言語の中で最初に提案されたのが，Gelfond ら[4]による言語 $\mathcal{A}$ である．言語 $\mathcal{A}$ では，決定性効果を持つアクションのみを記述することができる．以来，$\mathcal{A}$ を基として，$\mathcal{A}$ を拡張し，さらなる表現力を求めた様々なアクション言語が提案されており，非決定性効果を持つアクション[2),3),5),9),11)]，同時発生アクション[1),2)]，フルーエント間の制約[5),8),9),11)]，フルーエント間の依存関係[6),9),11)]などを表現できる言語が存在する．

本論文が対象とするのは，非決定性効果を持つアクションを記述できる言語（**非決定性アクション言語**）である．我々の目的は，1) **非決定性有限オートマトン**（NFA）の観点から新しい非決定性アクション言語 $\mathcal{NA}$ を提案し，2) すでに提案されている非決定性アクション言語の表現能力を形式的に解析する手段を提供することにある．

文献 10) により，最も基本的なアクション言語 $\mathcal{A}$ で表現可能な領域のクラスと，有限オートマトン（FA）で表現できる言語のクラスとが等しいことが分かっている．したがって，非決定性アクション言語を NFA の観点から設計することは，自然な拡張であると考えられる．

一方，すでに提案されている非決定性アクション言語に対し，言語 $\mathcal{NA}$ は次のような利点を持っている．非決定性アクション言語の多くは，言語 $\mathcal{A}$ を拡張し，さらなる表現力を求めて提案されたものである．しかしながら，各言語における非決定性の取り扱い方はまちまちであり，また，言語 $\mathcal{A}$ に比べ客観的にどの程

† 神戸大学大学院自然科学研究科情報メディア科学専攻
Division of Information and Media Science, Graduate School of Science and Technology, Kobe University
†† 神戸大学工学部電気電子工学科
Department of Electrical and Electronics Engineering, Kobe University

度表現力が向上したのか分かっていない．ここで提案する言語 $\mathcal{NA}$ は，FA と同じ表現能力を持っているため，たとえば $\mathcal{NA}$ と他のアクション言語との等価性を示すことで，その言語の適用範囲を形式的に議論できるようになる．実際に本論文では，言語 $\mathcal{AR}$[5]による記述を，それと等価な $\mathcal{NA}$ による記述に変換する手続きを与え，$\mathcal{NA}$ と $\mathcal{AR}$ が同じ表現能力を持つことを示す．

言語 $\mathcal{NA}$ の利点として，このほかにも，アクション言語におけるプランニング問題や問合せを，オートマトンの受理言語を求める問題に帰着できることがあげられる．プランニング問題のすべての可能な解は，正則表現により完全に記述できる．

さらに，オートマトン理論に対し，アクション言語のメリットを適用できるようになる．一般的に，アクション言語のように知識を宣言的に記述する枠組みでは，状態グラフのように知識をその構造の中に埋め込む枠組みに比べて，知識の修正が容易である．現在，オートマトンの応用分野は，テキスト編集，語彙解析，並行プロセス，プログラム検証など幅広く存在する．このような動的な領域の設計を，アクション言語により行えるのならば，それを修正することは容易になると考えられる．

本論文の残りは次のようになっている．まず言語 $\mathcal{NA}$ を続く 2 章で紹介し，3 章で $\mathcal{NA}$ と FA の等価性の結果を示す．4 章では等価性の結果を利用して例題を解く．5 章で $\mathcal{NA}$ と $\mathcal{AR}$ とが相互に変換できることを示す．

2. アクション言語 $\mathcal{NA}$

アクション言語 $\mathcal{NA}$ では，非決定性効果を持つアクションや，間接的効果を持つアクション，実行不可能なアクション，状態に対する制約を記述することができる．

2.1 構 文 論

2.1.1 構成要素

アクション言語 $\mathcal{NA}$ を 5 つ組 $\langle \mathcal{A}, \mathcal{F}, \mathcal{E}, \mathcal{C}, \mathcal{V} \rangle$ で表す．ここで $\mathcal{A}$ は**アクション名**の集合，$\mathcal{F}$ は**フルーエント名**の集合，$\mathcal{E}$ は**効果命題**（effect proposition）の集合，$\mathcal{C}$ は**制約**（constraint）の集合，$\mathcal{V}$ は**評価命題**（value proposition）の集合である．

言語 $\mathcal{NA}$ において，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．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．

言語 $\mathcal{NA}$ では，アクション列の集合を簡単に表すために，正則表現が利用できる．r と s がそれぞれ言語 R と S を表す正則表現のとき，$(r+s)$, $(r;s)$, (r^*) は，それぞれ集合 $R \cup S$, RS, R^* を表す．ここで $RS = \{rs \mid r \in R, s \in S\}$ であり，R^* は R の Kleene 閉包である[7]．正則表現 r に対し，それが表す集合を $L(r)$ と記述する．

2.1.2 命 題

言語 $\mathcal{NA}$ では 3 種類の命題を記述できる．まず，アクションとその効果（非決定的効果かもしれない）を記述する**効果命題**がある：

$$a \textbf{ causes } \phi_1 \mid \cdots \mid \phi_n \textbf{ if } \psi \qquad (1)$$

ここで a はアクション名，各 ϕ_i はフルーエントの連言肢，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．この縦棒演算子（|）は，論理式における論理和とも排他的論理和とも異なる．縦棒演算子では，$f \mid \neg f$ という記述が可能であるが，これを（排他的）論理和で記述すると恒真となる．

次に，状態に対する**制約**が記述できる：

$$\textbf{always } \phi \qquad (2)$$

ここで ϕ は式である．この命題は，あらゆる状態において，式 ϕ がつね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：

$$\phi \textbf{ after } r \qquad (3)$$

ここで ϕ は式，r はアクション名から構成される正則表現である．この命題は，$L(r)$ に含まれる任意のアクション列を実行した後，ϕ が成立することを言明する．

2.1.3 省 略 形

いくつか有用な省略形を定義する．ここで *True* は，恒真を意味する特別なフルーエントで，すべての状態で真となる．$False = \neg True$ である．

効果命題 (1) に対し，前提条件 ψ が *True* であるならば，**if** 以下を省略する：

$$a \textbf{ causes } \phi_1 \mid \cdots \mid \phi_n$$

また，a **causes** *False* **if** ψ の形をした効果命題を次のように記述する：

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

評価命題 (3) に対し，$r = \varepsilon$（空列）であるならば，簡単に次のように記述する：

$$\textbf{initially } \phi$$

変数を含む命題は，スキーマとして扱う（例題 2 参照）．

2.2 意味論

言語 $\mathcal{NA}$ では，効果命題・制約・評価命題を使って，状態変化領域を記述する．その命題の集合を**領域記述**と呼ぶ．我々は，その**解釈**を**非決定性状態遷移システム** $\langle Q, \mathcal{A}, \delta, q_0 \rangle$ で与える．ここで Q は状態の集合，$\mathcal{A}$ はアクション名の集合，q_0 は初期状態，δ は $Q \times \mathcal{A}$ から 2^Q への関数であり**遷移関数**と呼ぶ☆．

まず，状態と式の真偽値を定義する．

2.2.1 状態と式

状態 q を，すべ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：

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

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，$f \in q$ であるとき，かつそのときに限り，**状態** q **で式** ϕ **が真**であると定義する．任意の式 ϕ' については，論理演算子の真偽値表に従って拡張する．状態 q で式 ϕ' が真であるならば，$q \models \phi'$ と記述する．

任意の状態 q と制約 **always** ϕ に対し，$q \models \phi$ であるとき，かつそのときに限り，**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すべての制約を充足するならば，それを**正当な状態**と呼ぶ．

2.2.2 正当な遷移関数

正当な遷移関数を定義する前に，補助的な演算を定義する．任意のフルーエント名 F に対し，正のフルーエントを $|F|^+$ で表し，負のフルーエントを $|F|^-$ で表す．すなわち，

$$|F|^+ = |\neg F|^+ = F, \quad |F|^- = |\neg F|^- = \neg F$$

フルーエントの集合についても，同様に定義する：

$$|\{f_1, \cdots, f_n\}|^+ = \{|f_1|^+, \cdots, |f_n|^+\},$$
$$|\{f_1, \cdots, f_n\}|^- = \{|f_1|^-, \cdots, |f_n|^-\}$$

集合族 $A_1, \cdots, A_n$ に対し，次の直積演算を定義する：

$$A_1 \times \cdots \times A_n = \{a_1 \cup \cdots \cup a_n \mid a_1 \in A_1, \cdots, a_n \in A_n\}$$

効果命題 $P = (a$ **causes** $\phi_1 \mid \cdots \mid \phi_n$ **if** $\psi)$ に対し，次の関数 *effect* を定義する：

$$\mathit{effect}(P) = \{c(\phi_1), \cdots, c(\phi_n)\}$$

ただし $\phi_i = f_1 \wedge \cdots \wedge f_m$ に対して，$c(\phi_i) = \{f_1, \cdots, f_m\}$ とする．さらに，効果命題の集合 $\{P_1, \cdots, P_m\}$ に対して拡張する：

$$\mathit{effect}(\{P_1, \cdots, P_m\}) = \mathit{effect}(P_1) \times \cdots \times \mathit{effect}(P_m)$$

ここで $\mathit{effect}(\emptyset) = \emptyset$ とする．

言語 $\mathcal{NA}$ による領域記述を D とする．任意の状態 q と任意のアクション a に対し，遷移関数 δ が次式で定義されるとき，δ を**正当な遷移関数**と呼ぶ．

$$\delta(q, a) = \{q' \in Q \mid q' = (q \setminus (|e|^+ \cup |e|^-)) \cup e, \ e \in \mathit{effect}(\mathcal{P})\}$$

ここで Q はすべての状態の集合であり，$\mathcal{P}$ は次式で与えられる：

$$\mathcal{P} = \{P \in D \mid P = (a \textbf{ causes } \phi_1 \mid \cdots \mid \phi_n \textbf{ if } \psi) \text{ かつ } q \models \psi\}$$

effect($\mathcal{P}$) は，状態 q においてアクション a を実行すると発生するすべての非決定的効果を表している．正当な遷移関数において，アクションの効果 $e \in \mathit{effect}(\mathcal{P})$ 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．

最後に遷移関数 δ を，状態とアクション列の組に対する関数 $\hat{\delta} : Q \times * \to 2^Q$ に拡張する：

- $\hat{\delta}(q, \varepsilon) = \{q\}$
- $\hat{\delta}(q, wa) = \bigcup_{r \in \delta(q,w)} \delta(r, a)$

ここで w はアクション名の列を表し，a はアクション名を表す．$\hat{\delta}(q, a) = \delta(q, a)$ であるので，両者をともに δ で表すことにする．

2.2.3 モデル

モデルを定義するための補助的な定義を与える．

任意の解釈 $M = \langle Q, \mathcal{A}, \delta, q_0 \rangle$ と，任意の式 ϕ に対し，$M(\phi) = \langle Q, \mathcal{A}, \delta, q_0, Q(\phi) \rangle$ 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，**非決定性有限オートマトン**（NFA）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** $\mathcal{L}(M(\phi))$ を以下で定義する：

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

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：

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

$\mathcal{L}(M(\phi))$ は，目標状態に到達する可能性を持っているすべ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すべてのアクション列の集合を表している．$L(M(\phi)) \subseteq \mathcal{L}(M(\phi))$ である．もし $M(\phi)$ が**決定性有限オートマトン**（DFA）であるならば，両者は等しい．

☆ アクション名の集合 $\mathcal{A}$ は，任意の解釈において同じである．したがって，本来ならば解釈の定義に含める必要はない．ここでは，有限オートマトンとの対応のために解釈に加えている．

次に，評価命題の真偽値を定義する．任意の解釈 $M = \langle Q, \mathcal{A}, \delta, q_0 \rangle$ 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，$M \mid\sim P$ と記述する．特に比較するときに，$M \models P$ を，M において P が**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 $M = \langle Q, \mathcal{A}, \delta, q_0 \rangle$ が D の**モデル**であるのは，

i) Q が，すべての正当な状態の集合であり，
ii) δ が，Q 上に定義された正当な遷移関数であり，
iii) D に含まれるすべての評価命題が M において真であるとき，

かつそのときに限る．条件 i) を満たす状態集合は 1 つに決定され，条件 ii) を満たす遷移関数も，たかだか 1 つしか存在しない．したがって，同じ領域記述における異なったモデルは，その初期状態によってのみ異なる．領域記述は，モデルを持てば**無矛盾**である．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 $M = \langle Q, \mathcal{A}, \delta, q_0 \rangle$ が D の**軽信的なモデル**であるのは，i) と ii) に加え，

iii') D に含まれるすべての評価命題が M において軽信的に真であるとき，

かつそのときに限る．評価命題 P が D の任意の軽信的モデルに対して軽信的に真であるならば，$D \mid\sim P$ と書く．

以下に示すのは，帽子を被った Mary が湖に飛び込む例題である[5]．アクション *Jump* は，帽子が飛んでいくかもしれない $(Hat \mid \neg Hat)$ 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持つ．

例題 1　Mary の帽子[5]

Jump **causes** $Hat \mid \neg Hat$ **if** *Hat*
Jump **causes** *Lake*
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 $\neg Lake \wedge \neg Wet \wedge Hat$

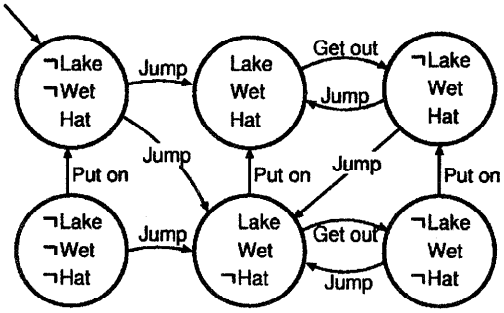


図 1　Mary が湖に飛び込む例題
Fig. 1　Mary jumped into the lake example.

この例題のモデルを図 1 に示す．同図において遷移先の定義されていないアクションは，状態 $\{False\}$ に遷移するものとする．この問題は，初期状態が一意に定まるため，モデルを 1 つしか持たない．この問題では，次のような評価命題が帰結できる．

$D \models \neg Lake \wedge Wet$ **after** *Jump*; *GetOut*
$D \not\models Hat$ **after** *Jump*; *GetOut*
$D \mid\sim Hat$ **after** *Jump*; *GetOut*

3. $\mathcal{NA}$ と有限オートマトン

本章では，$\mathcal{NA}$ と有限オートマトン（FA）の表現能力が等しいことを示す．ここで領域記述に含まれるフルーエント名とアクション名の数は有限であることを仮定する．

最初に，軽信的な帰結関係に基づけば，$\mathcal{NA}$ と FA の等価性は明らかである．すなわち，

定理 1　言語 $\mathcal{NA}$ による領域記述を D とする．D の軽信的モデル全体の集合を $\{M_1, \cdots, M_n\}$ とする．任意の式 ϕ に対し，$\mathcal{L}(M_1(\phi)) \cap \cdots \cap \mathcal{L}(M_n(\phi))$ を受理する FA $M'(\phi)$ を構成できる．このとき，

$$D \mid\sim \phi \textbf{ after } r \quad \Leftrightarrow \quad L(r) \subseteq \mathcal{L}(M'(\phi))$$

証明　$\mathcal{L}(M_i(\phi))$ が正則集合であることは，その定義より明らかである．正則集合の族は共通部分に関して閉じているので[7]，$\mathcal{L}(M_1(\phi)) \cap \cdots \cap \mathcal{L}(M_n(\phi))$ もまた正則集合であり，それを受理する FA を構成することができる．□

したがって，言語 $\mathcal{NA}$ による任意の領域記述から，軽信的帰結関係において等価な FA を構成できることが分かる．この逆の関係についても同様のことがいえる：文献 10) において，任意の FA を言語 $\mathcal{A}$ による領域記述に変換する正当な手続きが定義されている．言語 $\mathcal{A}$ は $\mathcal{NA}$ のサブセットであり，$\mathcal{A}$ による記述は $\mathcal{NA}$ による記述でもある．

よって以下では，懐疑的な帰結関係についても等価

となることを示す．まず，補助的な結果を示す．

補題1 $M = \langle Q, \Gamma, \delta, q_0, G\rangle$ を任意の NFA とする．M により確実に受理される言語

$$L(M) = \{w \in \Gamma^* \mid \delta(q_0, w) \subseteq G\}$$

は，DFA $M' = \langle 2^Q, \Gamma, \delta', \{q_0\}, 2^G\rangle$ の受理言語

$$L(M') = \{w \in \Gamma^* \mid \delta'(\{q_0\}, w) \in 2^G\}$$

に等しい．ここで δ' は以下で与えられる：

$$\delta(q_1, a) \cup \cdots \cup \delta(q_n, a) = \{p_1, \cdots, p_m\}$$

のとき，かつそのときに限り，

$$\delta'(\{q_1, \cdots, q_n\}, a) = \{p_1, \cdots, p_m\}$$

ここで $q_i, p_j \in Q, a \in \Gamma$ である．

証明 最初に，入力列 $w \in \Gamma^*$ の長さに関する数学的帰納法を利用して，$\delta(q_0, w) = \delta'(\{q_0\}, w)$ を示すことができる．次に，M と M' の受理言語が等しいことを示すために，$L(M) = L(M')$ でないと仮定する．すなわち，次式を満たす w が存在すると仮定する：

$$\exists w((w \notin L(M)) \wedge w \in L(M')) \vee ((w \in L(M)) \wedge w \notin L(M'))$$

まず，左辺 $w \notin L(M) \wedge w \in (M')$ を満たす w が存在しないことを示す．受理言語の定義より，この式は，次のように変換できる：

$$(\delta(q_0, w) \not\subseteq G) \wedge (\delta'(\{q_0\}, w) \in 2^G)$$

ここで $\delta(q_0, w) = \delta'(\{q_0\}, w)$ であるので，上式は矛盾する．同様に右辺 $w \in L(M) \wedge w \notin (M')$ の矛盾も証明できる．したがって，$L(M) = L(M')$ である．□

この結果は，“確実に受理される言語” もまた正則集合であり，それを受理するオートマトンを構成できることを示している．この結果より，次の定理を示すことができる．

定理2 言語 $\mathcal{NA}$ による領域記述を D とする．D のモデル全体の集合を $\{M_1, \cdots, M_n\}$ とする．任意の式 ϕ に対し，$L(M_1(\phi)) \cap \cdots \cap L(M_n(\phi))$ を受理する FA $M'(\phi)$ を構成できる．このとき，

$$D \models \phi \textbf{ after } r \quad \Leftrightarrow \quad L(r) \subseteq L(M'(\phi))$$

証明 補題1より $L(M_i(\phi))$ は正則集合である．したがって，定理1と同様に，$L(M_1(\phi)) \cap \cdots \cap L(M_n(\phi))$ もまた正則集合であり，それを受理する FA を構成することができる．□

以上のことから，$\mathcal{NA}$ と FA で表現可能な領域のクラスが等しいことが分かる．

4. 例　　題

以下に示すのは，金庫の中にある宝石を狙う泥棒について記述したものである．

例題2 宝石を狙う泥棒

Open **causes** *Opened*
Open **causes** *True* | *Ringing* **if** *On(Alarm1)* ∨ *On(Alarm2)*
Get **causes** *Jewel* **if** *Opened*
Cut(X) **causes** ¬*On(X)*
impossible *Open* **if** *Opened*
impossible *Get* **if** ¬*Opened* ∨ *Jewel*
impossible *Cut(X)* **if** ¬*On(X)*
initially ¬*Open* ∧ ¬*Ringing* ∧ ¬*Jewel*
initially *On(Alarm1)* ∧ *On(Alarm2)*

屋敷には警報器が2つある．泥棒は，警報を鳴らさずに，金庫の扉をうまく開けることができるかもしれない．この領域記述を D として，ここでは，警報を鳴らさずに宝石を手に入れるためのアクション列 r_1, r_2 を求めてみる：

$$D \mid\!\sim \textit{Jewel} \wedge \neg \textit{Ringing} \textbf{ after } r_1$$
$$D \models \textit{Jewel} \wedge \neg \textit{Ringing} \textbf{ after } r_2$$

r_1 は目標状態に到達する可能性を持っているアクション列であり，r_2 は必ず目標状態に到達するようなアクション列である．

このプランニング問題を解くために，まず D のモデル M を求める（**図2**）．M は D の唯一のモデルであり，M に目標状態を与えた $M(\textit{Jewel} \wedge \neg \textit{Ringing})$ は NFA である．定理1より，この NFA の受理言語 $\mathcal{L}(M(\textit{Jewel} \wedge \neg \textit{Ringing}))$ が，r_1 を満たすアクション列である．

一方 r_2 を求めるためには，定理2より，M を決定性状態遷移システム M' に変換すればよい．M' を**図3**に示す．M から M' への変換は補題1を利用する．M' に目標状態を与えた $M'(\textit{Jewel} \wedge \neg \textit{Ringing})$ は DFA であり，その受理言語 $L(M'(\textit{Jewel} \wedge \neg \textit{Ringing}))$ は，必ず目標状態に到達するアクション列 r_2 を表している．

このようにして，アクション言語におけるプランニング問題を，オートマトン理論における受理言語を求める問題に帰着できる．同様にして，

- ある状態においてあるフルーエントが成立するかしないか（予測）
- ある状態でどういうフルーエントが成立しているか（推定）

などを問い合わせることも可能である．

5. $\mathcal{AR}$ と $\mathcal{NA}$

アクション言語 $\mathcal{AR}$[5]は，$\mathcal{A}$ を基にして拡張された

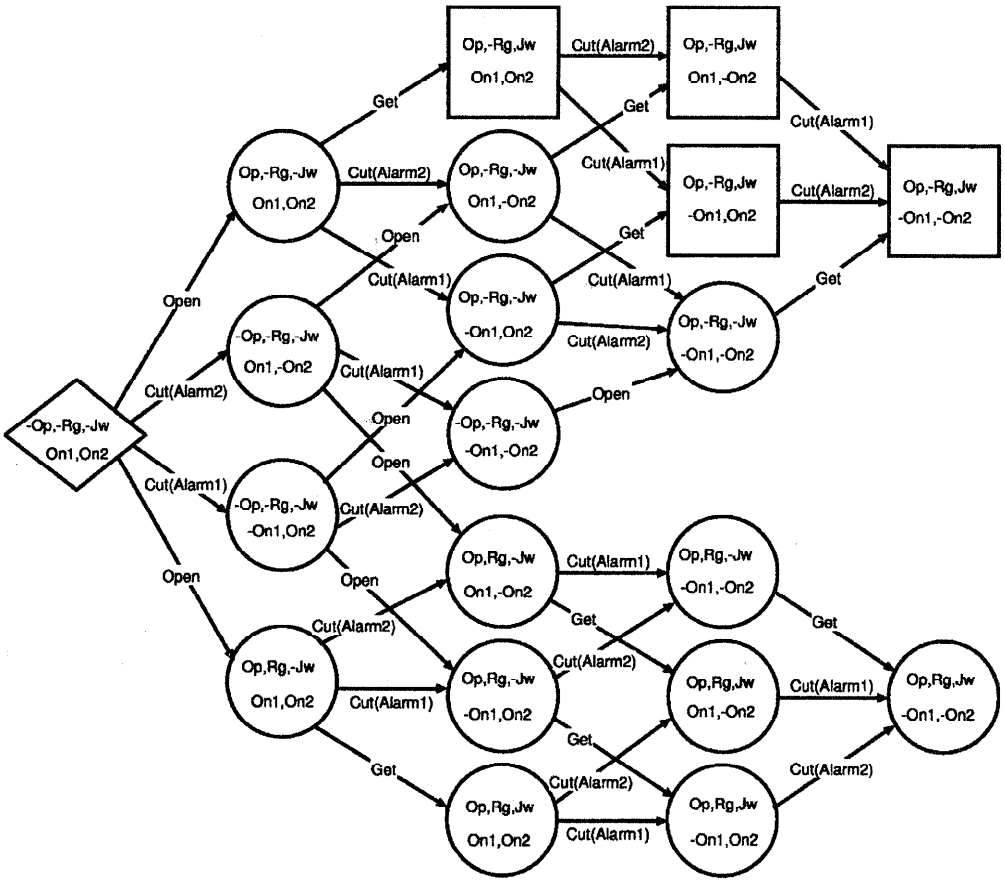


図2 宝石を盗め
Fig. 2 Steal the jewels.

言語であり，非決定的効果を持つアクションや間接的効果を持つアクションを記述できる．言語 $\mathcal{NA}$ における間接的効果の記述方法は，$\mathcal{AR}$ のものを参考にしている．言語 $\mathcal{AR}$ の特徴を以下に示す．これらは $\mathcal{NA}$ にはない特徴である．

- 慣性に従うフルーエントと従わないフルーエント（**非慣性フルーエント**）がある．また，フルーエントは，真偽の2値だけでなく，複数の値をとることができる．
- アクションの非決定的効果は，命題 **possibly changes** により記述する．遷移後の状態は，極小変化の概念により定義される．

逆に，$\mathcal{AR}$ にはないが，$\mathcal{NA}$ が持っている特徴として，

- アクションの非決定的効果は，効果命題において縦棒演算子（|）により分けて記述する．
- $\mathcal{NA}$ では記述できるが，$\mathcal{AR}$ では直接記述できない領域が存在する（5.6節，図4）．

本章では，言語 $\mathcal{AR}$ がこれらの特徴を持つのにもかかわらず，$\mathcal{NA}$ による等価な領域記述に変換できることを示す．まず，言語 $\mathcal{AR}$ の構文について説明する．

5.1 構 文 論

言語 $\mathcal{AR}$ において，フルーエント名の集合 $\mathcal{F}$ は，慣性フルーエントの集合 $\mathcal{F}_I$ と非慣性フルーエントの集合 $\mathcal{F}_N$ の2つに分かれている．フルーエント名 F に対し，その値域を集合 Rng_F で表す．もし F が命題的であるならば，$Rng_F = \{True, False\}$ である．フルーエント名 F が値 $V \in Rng_F$ を持つことを $(F \textbf{ is } V)$ と記述し，これを**原子式**と呼ぶ．原子式を論理演算子で結合したものを式と呼ぶ．

言語 $\mathcal{AR}$ では次の4種類の命題を記述できる：

$$a \textbf{ causes } \phi \textbf{ if } \psi \quad (4)$$

$$\textbf{always } \phi \quad (5)$$

$$a \textbf{ possibly changes } F \textbf{ if } \psi \quad (6)$$

$$\phi \textbf{ after } a_1; \ldots; a_n \quad (7)$$

ここで $a, a_1, \cdots, a_n$ はアクション名，ϕ，ψ は式，F はフルーエント名である．効果命題(4)では，効果 ϕ

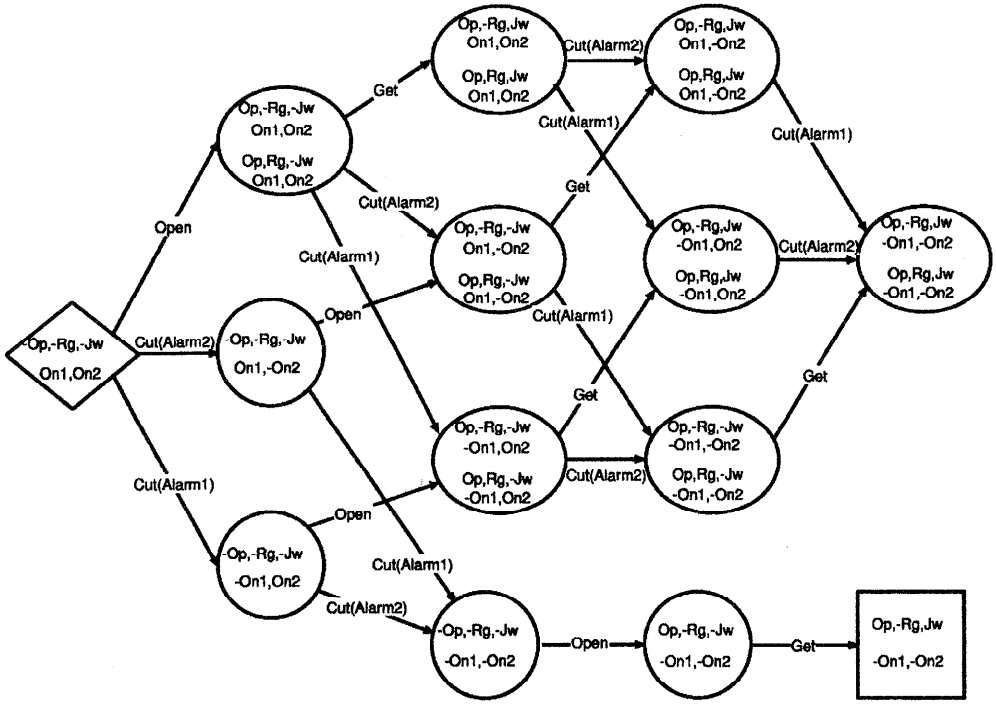


図 3 慎重に宝石を盗め
Fig. 3 Steal the jewels carefully.

に，連言だけでなく選言も記述することができる．命題 (6) は，前提条件 ψ の下でアクション a を実行すると，フルーエントの値が変化するかもしれないことを言明する．省略形については，$\mathcal{NA}$ と同様である．

例題 1 を言語 $\mathcal{AR}$ により記述すると，先頭の命題のみが，言語 $\mathcal{NA}$ による記述と異なり，

Jump **possibly changes** *Hat* **if** *Hat*

となる．ここで *Hat* は，原子式 *Hat* **is** *True* の省略形である．

5.2 意 味 論

5.2.1 状 態

言語 $\mathcal{AR}$ ではフルーエントが複数の値を持つので，フルーエント名 F をその値 $V \in Rng_F$ に写像する関数 σ により状態を定義する．この関数を，**評価関数**と呼ぶ．評価関数を，さらに原子式に対する関数に拡張する：

$$\sigma(F \textbf{ is } V) = \begin{cases} \mathit{True}, & \text{if } \sigma(F) = V, \\ \mathit{False}, & \text{otherwise} \end{cases}$$

任意の式については，論理演算子の真理値表に従って拡張する．任意の制約 **always** ϕ に対し，評価関数 σ が式 ϕ を充足するとき，かつそのときに限り，**評価関数 σ が制約 always ϕ を充足する**と定義する．

言語 $\mathcal{AR}$ による領域記述を D とする．**状態**とは，D に含まれるすべての制約 (5) を充足する評価関数である．

5.2.2 遷 移 関 数

遷移関数 Res_D を定義するために，補助的な関数 Res^0_D と New^a_D を定義する．任意のアクション名 a と任意の状態 σ に対し，Res^0_D を次式で定義する：

$$Res^0_D(\sigma, a) = \{\sigma' \mid$$
すべての (a **causes** ϕ **if** ψ) $\in D$ に対し，
σ が ψ を充足するならば，σ' は ϕ を充足する状態である $\}$

この関数は，慣性の法則を持っていない遷移関数のようなものである．すなわち，アクションの効果は遷移後の状態において成立するが，アクションの影響を受けていないフルーエントは変化するかもしれない．そこで，慣性の法則を実現するために，関数 New^a_D を定義する．任意のアクション名 a と任意の状態 σ，σ' に対し，$New^a_D(\sigma, \sigma')$ は，次の条件のいずれかを満たす原子式 (F **is** $\sigma'(F)$) の集合を返す：

条件 (1) F は慣性フルーエントであり，かつ $\sigma'(F) \neq \sigma(F)$ である．

条件 (2) 任意の (a **possibly changes** F **if** ψ) $\in D$

に対し，σ が ψ を充足する．

$New^a_D(\sigma,\sigma')$ の意味するところは，σ を遷移前の状態，σ' を遷移後の状態としたとき，遷移前と遷移後の状態間の変化分を集めることにある．そして遷移関数 Res_D の定義において，遷移前と遷移後の慣性フルーエントの状態変化が極小になるように定義する．これにより慣性の法則を実現する：

$$Res_D(a,\sigma) = \{\sigma' \in Res^0_D(a,\sigma) \mid \neg\exists\sigma'' \in Res^0_D(a,\sigma) \left(New^a_D(\sigma,\sigma'') \subset New^a_D(\sigma,\sigma')\right)\}$$

5.3 モデル

言語 $\mathcal{AR}$ による領域記述を D とすると，その**解釈**は，遷移関数と初期状態の対 (Res_D,σ_0) で与えられる．任意の解釈に対し，評価命題の真偽値を定義する．解釈 $M=(Res_D,\sigma_0)$ と評価命題 $P=(\phi$ **after** $a_1;\cdots;a_n)$ に対し，次のような列（履歴と呼ぶ）を考える．

$$\sigma_0,a_1,\sigma 1,\cdots,a_n,\sigma_n$$

ここで各 σ_k は状態であり，$\sigma_i \in Res_D(a_i,\sigma_{i-1})$ である．任意の履歴に対し，σ_n が式 ϕ を充足するならば，かつそのときに限り，**解釈 M において評価命題 P が真**である．ある解釈が，領域記述 D に含まれるすべての評価命題を充足するならば，その解釈を D の**モデル**と定義する．

5.4 $\mathcal{AR}$ から $\mathcal{NA}$ へ

言語 $\mathcal{AR}$ による領域記述を $D_{\mathcal{AR}}$ で表し，$\mathcal{NA}$ による領域記述を $D_{\mathcal{NA}}$ で表す．$D_{\mathcal{AR}}$ から $D_{\mathcal{NA}}$ への変換は，次の4つの段階を経て行われる．ここで，領域記述に含まれるフルーエント名，アクション名，命題の数，フルーエントの値域は有限であると仮定する．

(1) **非慣性フルーエントの処理**

$D_{\mathcal{AR}}$ に含まれるすべての非慣性フルーエントの集合を $\mathcal{F}_N$，すべてのアクション名の集合を $\mathcal{A}$ とする．すべての $F\in\mathcal{F}_N$，$a\in\mathcal{A}$ に対し，次の命題

a **possibly changes** F

を $D_{\mathcal{AR}}$ に加える．これ以後，$D_{\mathcal{AR}}$ に含まれるフルーエントは，すべて慣性フルーエントとして扱う．

(2) **多値フルーエントの処理**

$D_{\mathcal{AR}}$ に含まれるすべてのフルーエント名 F に対し，F が真/偽以外の値を持つフルーエントならば，次の操作を行う．ここで $Rng_F=\{V_1,V_2,\cdots,V_n\}$ とする．$D_{\mathcal{AR}}$ に含まれるすべての原子式 F **is** V_i を，真/偽の値を持つフルーエント F_{V_i} で置き換える．さらに $D_{\mathcal{AR}}$ に次の制約を加える：

always $F_{V_1}\dot\vee F_{V_2}\dot\vee\cdots\dot\vee F_{V_n}$

ここで $\dot\vee$ は排他性を表す演算子で，次式で定義される：

$$\phi_1\dot\vee\cdots\dot\vee\phi_k \equiv (\phi_1\vee\cdots\vee\phi_k)\wedge\bigwedge_{1\le i\ne j\le k}\neg(\phi_i\wedge\phi_j)$$

(3) **possibly changes 命題と効果命題の処理**

$D_{\mathcal{AR}}$ に含まれるすべてのアクション名 a について，次の処理を施す．まず，a に関する効果命題と **possibly changes** 命題を集める：

a **causes** ϕ_1 **if** ψ_1

$\vdots$

a **causes** ϕ_n **if** ψ_n

a **possibly changes** F_1 **if** θ_1

$\vdots$

a **possibly changes** F_m **if** θ_m

これらの命題から構成される領域記述を D とする．D のすべての状態 s に対し，$Res_D(a,s)=\{s_1,\cdots,s_k\}$ を求め，次の命題を D_{NA} に加える：

a **causes** $c(s_1\setminus s)\mid\cdots\mid c(s_k\setminus s)$ **if** $c(s)$ (8)

ただし，$t=\{F_1,\cdots,F_l\}$ に対し，$c(t)=F_1\wedge\cdots\wedge F_l$ とする．

(4) **評価命題と制約の処理**

$D_{\mathcal{AR}}$ に含まれる評価命題と制約は，$D_{\mathcal{NA}}$ でもそのまま利用できるので，すべて $D_{\mathcal{NA}}$ にコピーする．

この手続きの正当性を示す．

定理3 言語 $\mathcal{AR}$ による領域記述を $D_{\mathcal{AR}}$ とする．$D_{\mathcal{AR}}$ を，上の変換手続きにより，言語 $\mathcal{NA}$ による領域記述 $D_{\mathcal{NA}}$ に変換する．このとき，

$$D_{\mathcal{AR}}\models\phi\ \textbf{after}\ a_1;\cdots;a_n \Leftrightarrow D_{\mathcal{NA}}\models\phi\ \textbf{after}\ a_1;\cdots;a_n$$

証明 各ステップについて証明の概略を示す．

(1) 言語 $\mathcal{AR}$ による領域記述を D とし，ステップ1の処理を施した後の領域記述を D' とする．このステップの正当性の証明は，任意のアクション名 a と任意の状態 σ に対し，$Res_D(a,\sigma)=Res_{D'}(a,\sigma)$ を示すことである．

遷移関数 Res_D は，遷移前と遷移後の慣性フルーエントの変化を極小にするように定義されている．したがって，遷移前の状態 σ に含まれる任意の非慣性フルーエント F は，遷移後の状態 $\sigma'\in Res_D(a,\sigma)$ において任意の値 $V\in Rng_F$ をとることができる．

D' において F は，**possibly changes** 命題に変換され，さらに慣性フルーエントとして扱われる．ここで，(a) $Res^0_D(a,\sigma)=Res^0_{D'}(a,\sigma)$ であることと，(b) New^a_D の定義の条件 (2) より，F は，遷移関数

$Res_{D'}$ においても，極小化の対象にはならない．したがって，F は，遷移後の状態 $\sigma' \in Res_{D'}(a,\sigma)$ において，任意の値 $V \in Rng_F$ をとることができる．

(2) $\mathcal{AR}$ におけるフルーエントは，複数の値を持つことができるが，特に変数が利用できるわけではなく，それらは命題的に振る舞っている．したがって，このような書き換えを行っても，モデルが本質的に変化することはない．

(3) 言語 $\mathcal{AR}$ による領域記述を D とし，ステップ3の処理を施して得られる言語 $\mathcal{NA}$ による領域記述を D' とする．この時点で D に含まれるフルーエントは，すべて慣性フルーエントであり，それらは真/偽の値を持つ命題的フルーエントである．したがって，D における状態を原子式の集合ではなく，$\mathcal{NA}$ のように，すべ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考えることができる．ここでの正当性の証明は，任意のアクション名 a と任意の状態 σ に対し，$Res_D(a,\sigma) = \delta(\sigma,a)$ を示すことである．ここで δ は D' の正当な遷移関数である．$Res_D(a,\sigma) = \{\sigma_1,\cdots,\sigma_n\}$ とすれば，式 (8) と $\mathcal{NA}$ の遷移関数の定義より，$\delta(\sigma,a) = \{\sigma_1,\cdots,\sigma_n\}$ を示すことができる．

(4) 言語 $\mathcal{AR}$ においても，言語 $\mathcal{NA}$ においても，制約は，状態を同様に制限するだけである．また，これまでのステップで両領域の遷移関数が等しいことが分かっている．したがって，同じ評価命題を持つ両領域のモデルは等しい． □

定理3より，言語 $\mathcal{AR}$ による任意の領域記述は，あらゆる評価命題の帰結関係において等価な言語 $\mathcal{NA}$ による記述に変換できることが分かる．

5.5 変換例

言語 $\mathcal{AR}$ で記述した例題1を，前節の変換手続きを用いて，言語 $\mathcal{NA}$ による記述に戻してみる．

Jump **causes** *Lake* ∧ *Hat* | *Lake* ∧ ¬*Hat*
if ¬*Lake* ∧ *Hat*
Jump **causes** *Lake* **if** ¬*Lake* ∧ ¬*Hat*
GetOut **causes** ¬*Lake* **if** *Lake*
PutOn **causes** *Hat* **if** ¬*Hat*
impossible *Jump* **if** *Lake* ∧ *Hat*
impossible *Jump* **if** *Lake* ∧ ¬*Hat*
impossible *GetOut* **if** ¬*Lake*
impossible *PutOn* **if** *Hat*
always *Lake* ⊃ *Wet*
initially ¬*Lake* ∧ ¬*Wet* ∧ *Hat*

元々の領域記述と比べると多少冗長な記述になっているが，この領域記述は図1と同じモデルを持つ．

5.6 $\mathcal{NA}$ から $\mathcal{AR}$ へ

言語 $\mathcal{NA}$ による任意の領域記述は，以下のようにして，それと等価な言語 $\mathcal{AR}$ による記述に変換することができる．

言語 $\mathcal{NA}$ による領域記述を $D_{\mathcal{NA}}$ とする．定理2による変換を用いて，$D_{\mathcal{NA}}$ をそれと等価な FA M に変換し，さらに M を DFA M' に等価変換する[7]．文献10) の手続きにより，M' を言語 $\mathcal{A}$ による記述に変換したものを $D_{\mathcal{A}}$ とする．このとき，

定理4 任意の評価命題 ϕ **after** $a_1;\cdots;a_n$ に対して☆，

$$D_{\mathcal{NA}} \models \phi \textbf{ after } a_1;\cdots;a_n \Leftrightarrow D_{\mathcal{A}} \models \phi \textbf{ after } a_1;\cdots;a_n$$

証明 定理2および文献10) の定理6を用いて証明できる． □

言語 $\mathcal{A}$ は言語 $\mathcal{AR}$ のサブセットであるので，$\mathcal{A}$ による記述は，$\mathcal{AR}$ による記述でもある．したがって，$\mathcal{NA}$ による任意の領域記述は，あらゆる評価命題の帰結関係において等価な言語 $\mathcal{AR}$ による記述に変換できる．

定理3と定理4より，$\mathcal{NA}$ で表現できる領域のクラスと，$\mathcal{AR}$ で表現できる領域のクラスとは等しいことが分かる．このように，$\mathcal{NA}$ は，他のアクション言語の適用範囲を形式的に議論するのに利用できる．

ところで，言語 $\mathcal{NA}$ では直接記述できるが，言語 $\mathcal{AR}$ では直接記述できない領域が存在する（**図4**）☆☆．以下でそのような領域が存在することを示す．

言語 $\mathcal{AR}$ による領域記述を D とする．ここで，次のような遷移を考える：

$$Res_D(\sigma_1,a) = \{\sigma_1,\sigma_2\} \quad (9)$$

$f,g \in \sigma_1$，$\neg f,\neg g \in \sigma_2$ とする．もし D が a に関する **possibly changes** 命題を含んでいないと仮定すれば，極小変化の定義より，遷移先に σ_2 が含まれることはない．よって仮定は矛盾する．したがって，D は次のような命題を含んでいる必要がある：

$$a \textbf{ possibly changes } f \textbf{ if } \psi \quad \textbf{always } \neg f \Rightarrow \neg g \quad (10)$$

または

☆ 本来，言語 $\mathcal{A}$ では式を認めていないが，たとえば，$D_{\mathcal{A}} \models f_1 \wedge f_2$ **after** $\vec{a}$ を，$(D_{\mathcal{A}} \models f_1 \textbf{ after } \vec{a}) \wedge (D_{\mathcal{A}} \models f_1 \textbf{ after } \vec{a})$ と解釈することで，式を利用することができる．

☆☆ ある領域を直接記述できるとは，その領域で使用されているフルーエント名を変更せずに記述できることをいうものとする．

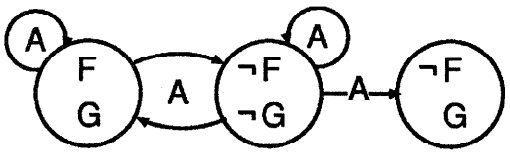


図4 $\mathcal{AR}$ では直接記述できない例
Fig. 4 An example which can not be represented in $\mathcal{AR}$.

$$\begin{array}{l} a \textbf{ possibly changes } f \textbf{ if } \psi \\ a \textbf{ possibly changes } g \textbf{ if } \psi \\ \textbf{always } (f \wedge g) \vee (\neg f \wedge \neg g) \end{array} \quad (11)$$

ここで ψ は状態 σ_1 で真となる前提条件である．ここでさらに次のような遷移を考える：

$Res_D(\sigma_2, a) = \{\sigma_1, \sigma_2, \sigma_3\}$

$\neg f, g \in \sigma_3$ とする．しかし，D は式 (10) または (11) の命題を含んでいるので，σ_2 から σ_3 へ遷移することはできない．逆に，D が式 (10) または (11) の命題を含んでいない場合，式 (9) の遷移を実現できない．すなわち，言語 $\mathcal{AR}$ では直接記述できない領域が存在する．ところで，言語 $\mathcal{NA}$ では次のように直接記述できる：

$$\begin{array}{l} a \textbf{ causes } (f \wedge g) \mid (\neg f \wedge \neg g) \textbf{ if } f \wedge g \wedge \psi \\ a \textbf{ causes } (f \wedge g) \mid (\neg f \wedge \neg g) \mid (\neg f \wedge g) \\ \qquad\qquad\qquad\qquad\qquad\qquad \textbf{if } \neg f \wedge \neg g \wedge \psi' \end{array}$$

ここで ψ' は状態 σ_2 で真となる前提条件である．

6. 関連研究

言語 $\mathcal{AR}$ と同じ極小変化の概念を持つアクション言語に，Lifschitz の DL_{if}[9]や Turner の $\mathcal{AC}$[11]がある．$\mathcal{AC}$ は $\mathcal{AR}$ の拡張版として提案されている．DL_{if} も $\mathcal{AC}$ も，非決定性アクションや制約を記述することができる．さらに，これらの言語では，状態間の依存関係を表現することができる：

Lake **suffice for** *Wet*

これは状態に対する規則として解釈される．制約 **always** *Lake* $\supset$ *Wet* とは異なり，対遇をとることはできない．

アクションの非決定的効果の取り扱い方には，いくつかのアプローチがある．上にあげた極小変化の概念に基づくアプローチもその1つであり，言語 $\mathcal{NA}$ のように排他的に効果を記述するアプローチもその1つである．Boutilier らによる言語 $\mathcal{A}^{\mathrm{ND}}$ も $\mathcal{NA}$ と同様の縦棒演算子を用いて非決定的効果を排他的に記述している[3]．ただし $\mathcal{A}^{\mathrm{ND}}$ では，その解釈をプロセス論理の一種で与えており，また効果命題に対し，アクションの前提条件が相互に充足してはならないという強い制限を課している．Borncheuer らによる $\mathcal{A}_N$ も，非決定的効果を複数の効果命題を排他的に用いて記述する[2]：

Jump **alternatively caused** *Lake* $\wedge$ *Hat*

Jump **alternatively caused** *Lake* $\wedge \neg$*Hat*

さらに彼らは，$\mathcal{A}_N$ を同時発生アクションを扱えるように拡張し（$\mathcal{A}_{NCC}$），それを論理プログラムに変換する完全で健全な手続きを示している．$\mathcal{A}^{\mathrm{ND}}$ や $\mathcal{A}_N$ による記述を $\mathcal{NA}$ による記述に変換することは，非決定性の取り扱い方が似ていることから，比較的容易であると考えられる．

Baral らによる $\mathcal{A}_C$ もまた同時発生アクションを扱うことができる[1]．彼らは，$\mathcal{A}_C$ による記述を拡張論理プログラムに変換する完全で健全な手続きを示している．同時発生アクションを表現できるように $\mathcal{NA}$ を拡張することも，今後の重要な課題の1つである．

7. 結　論

本論文では，NFA に基づく新しいアクション言語 $\mathcal{NA}$ を提案した．$\mathcal{NA}$ を利用することで，他のアクション言語の適用範囲を形式的に議論したり，アクション言語におけるプランニング問題や問合せをオートマトン理論における受理言語を求める問題に帰着したりすることができる．また，オートマトン理論に対し，アクション言語のメリットを適用できるようになる．

今後の課題には，モデルの効率的な計算方法の探求や，アクション言語とオートマトンにおける記述量の形式的解析，さらなる表現力の向上などがある．

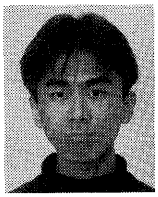
謝辞 本研究に対して，貴重なご助言をいただいた豊橋技術科学大学情報工学系の中島浩教授に深く感謝します．なお，本研究の一部は，文部省科学研究費補助金（特別研究員奨励賞）による．

参考文献

1) Baral, C. and Gelfond, M.: Reasoning About Effects of Concurrent Actions, *Journal of Logic Programming*, Vol.31, No.1-3, pp.85–117 (1997).
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).

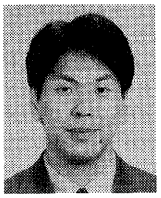
4) Gelfond, M. and Lifschitz, V.: Representing action and change by logic programs, *Journal of Logic Programming*, Vol.17, No.2–4, pp.301–321 (1993).
5) Giunchiglia, E., Kartha, G.N. and Lifschitz, V.: Representing action: indeterminacy and ramifications, *Artificial Intelligence*, Vol.95, pp.409–443 (1997).
6) Giunchiglia, E. and Lifschitz, V.: Dependent Fluents, *Proc. IJCAI-95*, pp.1964–1969 (1995).
7) Hopcroft, J.E. and Ullman, J.D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Massachusetts (1979). 野崎昭弘ほか（訳）：オートマトン 言語理論 計算論（I, II），サイエンス社 (1984).
8) Kartha, G.N. and Lifschitz, V.: Actions with Indirect Effects (Preliminary Report), *Proc. KR-94*, pp.341–350 (1994).
9) Lifschitz, V.: Two components of an action language, *Annals of Mathematics and Artificial Intelligence* (1997).
10) 鍋島英知，井上克巳：アクション言語 $\mathcal{A}$ のためのオートマトン理論，情報処理学会論文誌，Vol.38, No.3, pp.462–471 (1997).
11) Turner, H.: Representing actions in logic programs and default theories: A situation calculus approach, *Journal of Logic Programming*, Vol.31, No.1–3, pp.245–298 (1997).

(平成 10 年 7 月 27 日受付)
(平成 11 年 9 月 2 日採録)



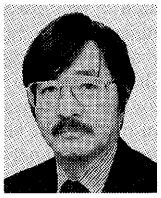
鍋島 英知（学生会員）

1973 年生．1998 年豊橋技術科学大学大学院工学研究科情報工学専攻修士課程修了．現在，神戸大学大学院自然科学研究科情報メディア科学専攻に在学中．日本学術振興会特別研究員．アクション理論等の人工知能，オートマトンの研究に従事．



井上 克巳（正会員）

1959 年生．1982 年京都大学工学部数理工学科卒業．1984 年同大学院工学研究科数理工学専攻修士課程修了．松下電器産業（株），(財) 新世代コンピュータ技術開発機構，豊橋技術科学大学工学部情報工学系を経て，1997 年より神戸大学工学部電気電子工学科助教授．京都大学博士（工学）．人工知能，論理プログラミング，計算機科学に関する教育研究に従事．人工知能学会，AAAI 各会員．



羽根田博正（正会員）

1939 年生．1972 年カリフォルニア大学（バークレイ校）大学院博士課程修了．同年神戸大学工学部助教授．1982 年同教授．1997 年 IEEE 産業エレクトロニクス部門学会会長．計算機援用設計・解析に関する研究に従事．(Ph.D.) 計測自動制御学会，システム制御情報学会，電気学会，電子情報通信学会，ACM，IEEE（フェロー）各会員．