知能と複雑系　115 － 6
(1999. 1. 11)

アクション言語 $\mathcal{A}$ における因果関係の学習

鍋島 英知[1]　井上 克巳[1,2]　羽根田 博正[1,2]

[1]神戸大学大学院自然科学研究科
[2]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 情報メディア科学専攻
TEL: (078)881-1212 内線 5243　FAX: (078)881-3193
e-mail: nabesima@cs-ee.eedept.kobe-u.ac.jp

あらまし

動的世界において自律的に行動するエージェントには，これまでに出会ったことのない環境においても，アクションの実行結果を基にして，その環境に対する知識を学習する能力が望まれる．本論文では，知識表現のための言語としてアクション言語 $\mathcal{A}$ を用い，背景知識のない状態から，あるアクションに対する効果を記述する規則を学習することを考える．この学習アルゴリズムは，事象の観測例から，言語 $\mathcal{A}$ による因果関係の記述を決定木を用いて生成する．

キーワード　アクション言語，学習，因果関係，決定木

Learning Causality on Action Language $\mathcal{A}$

Hidetomo Nabeshima*　Katsumi Inoue**,***　Hiromasa Haneda**,***

*Graduate School of Science and Technology, Kobe University
**Department of Electrical and Electronics Engineering,
Faculty of Engineering, Kobe University

*Division of Information and Media Science,
Graduate School of Science and Technology, Kobe University
Rokkodai, Nada Kobe 657-8501 Japan
TEL: +81 78 881 1212 (5243)　FAX: +81 78 881 3193
e-mail: nabesima@cs-ee.eedept.kobe-u.ac.jp

Abstract

Autonomous agents that behave in dynamic world needs the learning ability that can acquire knowledge about the environment in which the agents have never encountered. In this paper, we investigate a learning algorithm which produces rules about effects of actions from observations in action language $\mathcal{A}$.

key words　Action languages, learning, causality, decision tree

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である．そのようなアクション言語の中で最初に提案され，かつ最も基本的とされているのが，Gelfond と Lifschitz[1] による言語 $\mathcal{A}$ である．言語 $\mathcal{A}$ では決定性の効果をもつアクションしか記述できないが，非決定性効果をもつアクション同時発生アクション，フルーエント間の制約，フルーエント間の依存関係などを表現できる言語が提案されている [8, 4].

動的世界において自律的に行動するエージェントには，環境の変化に対する即応性や，長期的な目標に対して頑健なプランを生成する能力が望まれる．そのためには，適切な知識表現が必要となるが，我々はこれに対し，多様なアクションを柔軟かつ簡潔に表現できるアクション言語を用いてアプローチする．

ところで，エージェントには，これまでに出会ったことのない環境においても，アクションを実行した結果を基にして，その環境に対する知識を自律的に学習する能力が必要となる．

本論文では，このアクション言語の学習について焦点をあてて，背景知識のない状態からの学習を考える．対象とするアクション言語も簡潔な言語 $\mathcal{A}$ とする．学習アルゴリズムへの入力は，事象の観測例であり，出力は，言語 $\mathcal{A}$ による因果関係の記述 あるアクションを実行すると，それがどのような影響を及ぼすかという記述である．言語 $\mathcal{A}$ は命題的フルーエントのみを扱うので，学習アルゴリズムとして，決定木を学習するアルゴリズム [6] を採用した．

2 アクション言語 $\mathcal{A}$

アクション言語 $\mathcal{A}$ は，2 種類の記号の集合と，2 種類の命題の集合からなる．すなわち，**アクション名**の集合と**フルーエント名**の集合，**評価命題** (value proposition) の集合と**効果命題** (effect proposition) の集合である．

アクションとは，状態を変化させる行動であり，フルーエントとは，状態に依存する属性である．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．$\neg\neg F = F$ である．単に“フルーエント F”と記述した場合，F は正または負のフルーエントとする．

評価命題は次の形式をしている：

$$F \textbf{ after } A_1;\ldots;A_m. \quad (m \geq 0) \tag{1}$$

ここで F はフルーエント，$A_1,\ldots,A_m$ はアクション名である．$m = 0$ のときは簡潔に

initially F

と記述する．効果命題は次の形式をしている：

$$A \textbf{ causes } F \textbf{ if } P_1,\ldots,P_n. \quad (n \geq 0) \tag{2}$$

ここで A はアクション名，$F, P_1,\ldots,P_n$ はフルーエントである．F をアクションの**効果**といい，$P_1,\ldots,P_n$ を**前提条件**という．$n = 0$ のときは **if** を落して

A **causes** F

と記述する．

言語 $\mathcal{A}$ による**領域記述**は，これら 2 種類の命題の集合である．領域記述 D に含まれるすべてのフルーエント名の集合を $\mathit{fluent}(D)$ で表す．**状態** σ 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．つまり，

$$\sigma = S \cup \{\neg F \mid F \in \mathit{fluent}(D) \setminus S\}.$$

ここで $S \subseteq \mathit{fluent}(D)$ である[1]．

解釈 I は，σ_0 を初期状態，Φ を遷移関数とすると，その対 (Φ, σ_0) である．**遷移関数** Φ は，アクション名 A と状態 σ の対 (A, σ) を状態へ写像する．任意の解釈 I と，任意のアクション列 $A_1;\ldots;A_m$ に対して，$I^{A_1;\ldots;A_m}$ は次の状態を与える：

$$I^{A_1;\ldots;A_m} = (\Phi, \sigma_0)^{A_1;\ldots;A_m} = \Phi(A_m, \Phi(A_{m-1}, \ldots, \Phi(A_1, \sigma_0)\ldots)).$$

式 (1) のような評価命題に対して，$F \in I^{A_1;\ldots;A_m}$ であれば，式 (1) が解釈 I において**真**であるといい，その他の場合を**偽**であるという．

解釈 I が領域記述 D の**モデル**であるのは，(i) D に含まれるすべての評価命題が I において真であり，(ii) すべてのアクション名 A，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) もし D が，状態 σ で前提条件を満たす式 (2) の形の効果命題を含めば，そのとき $F \in \Phi(A, \sigma)$

[1] 文献 [1] では，負のフルーエントを含まないように状態を定義しているが，言語 $\mathcal{A}$ を定義する上で差異は生じない．

である，

(b) もし D が，そのような効果命題を含まなければ，$F \in \sigma \Leftrightarrow F \in \Phi(A,\sigma)$ である．

条件 (b) は**慣性の法則**を表している．上の条件を満たす遷移関数は，多くとも1つしか存在しない．よって，同じ領域記述における異なったモデルは，そのモデル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**であり，モデルが1つしかなければ**完全**であるという．もし，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，D はその評価命題を**帰結する**といい，次のように記述する：

$$D \models (F \textbf{ after } A_1;\ldots;A_m). \quad (m \geq 0)$$

2.1 省略形

本論文では，以下のような省略形を用いる．任意の領域記述 D に対し，

$$D \models (F_1 \textbf{ after } A_1;\ldots;A_m) \wedge \cdots \wedge (F_n \textbf{ after } A_1;\ldots;A_m)$$

であるとき，簡潔に次のように記述する：

$$D \models (F_1 \wedge \ldots \wedge F_n \textbf{ after } A_1;\ldots;A_m)$$

同様に，$n=0$ のときについても

$$D \models (\textbf{initially } F_1 \wedge \ldots \wedge F_n)$$

と記述する．また，次のような効果命題を

$$A \textbf{ causes } \mathit{False} \textbf{ if } P_1,\ldots,P_n$$

次のように記述する：

$$\textbf{impossible } A \textbf{ if } P_1,\ldots,P_n$$

これは，アクション A が条件 $P_1 \wedge \ldots \wedge P_n$ の下で実行不可能であることを表している．

2.2 記述例

例題 1 Yale Shooting 領域 [2]

initially $\neg loaded$.

initially $alive$.

$load$ **causes** $loaded$.

$shoot$ **causes** $\neg alive$ **if** $loaded$.

$shoot$ **causes** $\neg loaded$.

この領域記述のモデルを図 1 に示す．この記述は完全であるので，唯1つのモデルをもつ．

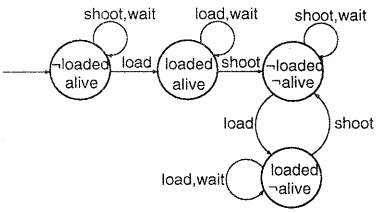

図 1: Yale Shooting 領域

例題 2 人間と狼と山羊とキャベツの問題

図 2 は，この問題の状態遷移図である [3]．最初，男と狼と山羊とキャベツは川の左岸にいる．男の目的は，狼と山羊とキャベツを小舟で右岸へ運ぶことである．ただし，小舟は一度に1つの荷物しか運べない．また，狼と山羊を岸に残していくと狼が山羊を食べてしまい，山羊とキャベツを岸に残していくと山羊がキャベツを食べてしまう．図中の M,W,G,C は，それぞれ男と狼と山羊とキャベツが左岸にいることを表し，¬M,¬W,¬G,¬C が右岸にいることを表している．m,w,g,c は，男と狼と山羊とキャベツの対岸への移動を表す．

この問題をアクション言語 $\mathcal{A}$ により記述すると次のようになる [5]：

initially $M \wedge W \wedge G \wedge C$

m	**causes**	M	**if**	$\overline{M} \wedge W \wedge \overline{G} \wedge C$
m	**causes**	$\overline{M}$	**if**	$M \wedge W \wedge \overline{G} \wedge C$
m	**causes**	M	**if**	$\overline{M} \wedge \overline{W} \wedge G \wedge \overline{C}$
m	**causes**	$\overline{M}$	**if**	$M \wedge \overline{W} \wedge G \wedge \overline{C}$
g	**causes**	$M \wedge G$	**if**	$\overline{M} \wedge \overline{G}$
g	**causes**	$\overline{M} \wedge \overline{G}$	**if**	$M \wedge G$
w	**causes**	$M \wedge W$	**if**	$\overline{M} \wedge \overline{W} \wedge \overline{G} \wedge C$
w	**causes**	$\overline{M} \wedge \overline{W}$	**if**	$M \wedge W \wedge \overline{G} \wedge C$
w	**causes**	$M \wedge W$	**if**	$\overline{M} \wedge \overline{W} \wedge G \wedge \overline{C}$
w	**causes**	$\overline{M} \wedge \overline{W}$	**if**	$M \wedge W \wedge G \wedge \overline{C}$
c	**causes**	$M \wedge C$	**if**	$\overline{M} \wedge W \wedge \overline{G} \wedge \overline{C}$
c	**causes**	$\overline{M} \wedge \overline{C}$	**if**	$M \wedge W \wedge \overline{G} \wedge C$
c	**causes**	$M \wedge C$	**if**	$\overline{M} \wedge \overline{W} \wedge G \wedge \overline{C}$
c	**causes**	$\overline{M} \wedge \overline{C}$	**if**	$M \wedge \overline{W} \wedge G \wedge C$

この問題におけるアクションの効果は非決定的である．すなわち，状態によって遷移先が定義されていない場合がある．そこで，遷移先の定義されていないアクションは遷移元に回帰すると仮定した．

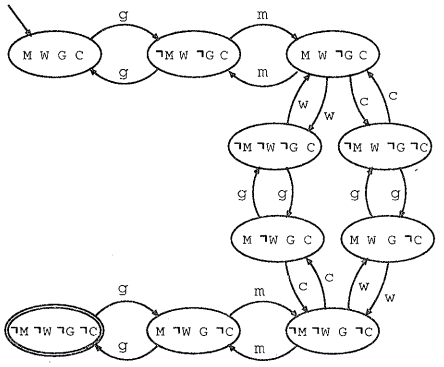

図 2: 人間と狼と山羊とキャベツの問題

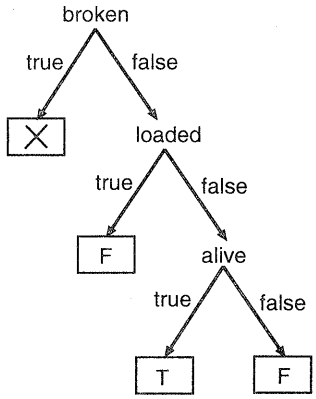

図 3: 決定木 $\langle shoot, alive \rangle$

3 因果関係の学習

本研究では，言語 $\mathcal{A}$ による因果関係の記述を学習するために，決定木の学習アルゴリズム [6] を用いる．まず，学習する決定木を定義する．

決定木は，任意のアクション名 A と任意のフルーエント名 F の対 $\langle A, F \rangle$ に対して定義される．

定義 1 決定木 $\langle A, F \rangle$ の節点と枝を以下で定義する：

- 各非終端節点は，フルーエント名をもつ．その節点から出る枝には，そのフルーエントの真偽値がラベル付けされる．
- 各終端節点は，“T” または “F” または “×” というラベルをもつ．

直感的には，終端節点に到達する経路が，アクション A の前提条件を表しており，終端節点が，その前提条件のもとでアクション A を実行した場合，フルーエント F にどのような影響を及ぼすのかを表している：F が真になる (true) ／偽になる (false) ／ A は実行できない (×)．

図 3 に，決定木 $\langle shoot, alive \rangle$ の例を示す．アクション *shoot* の実行により，フルーエント *alive* が真になるのか (true)，偽になるのか (flase)，それとも実行できないのか (×) が表現されている．この決定木を言語 $\mathcal{A}$ による記述に直訳すると次のようになる：

$$shoot \textbf{ causes } \neg alive \textbf{ if } \neg broken \wedge loaded$$
$$\textbf{impossible } shoot \textbf{ if } \neg broken$$

shoot が *alive* に影響を及ぼさない場合は，慣性の法則により記述する必要がない．また，銃が壊れている (*broken*) ならば，*shoot* を実行できないので，前者の命題は，さらに簡単化することができる：

$$shoot \textbf{ causes } \neg alive \textbf{ if } loaded$$

学習アルゴリズムへの入力は，観測された事例の集合である．“例” を以下で定義する．

定義 2 例は，次の形式をしている：

$$(F_1 \wedge \ldots \wedge F_n \textbf{ after } A) \wedge$$
$$(\textbf{initially } P_1 \wedge \ldots \wedge P_m) \quad (n \geq 1, m \geq 0)$$

これは，フルーエント $P_1, \ldots, P_m$ が観測された状態でアクション A を実行したところ，フルーエント $F_1, \ldots, F_n$ が観測されたことを表している．これを簡潔に次のように記述する：

$$F_1 \wedge \ldots \wedge F_n \textbf{ after } A \textbf{ if } P_1 \wedge \ldots \wedge P_m$$

$P_1 \wedge \ldots \wedge P_m$ を**前提条件**と呼び、$F_1 \wedge \ldots \wedge F_n$ を**結論**と呼ぶ．アクションが実行できなかった場合，前提条件を *false* と記述する．

上の定義において，我々は，システムがすべてのフルーエントを観測できるとは仮定しない．観測結果には欠損データが存在するかも知れないし，ノイズが含まれるかも知れない．またアクションの長さを 1 に限定している．これらについては 5 章で触れる．補助的な定義をあといくつか必要とする．

定義 3 例集合 $\mathcal{I}$ とフルーエント名 F に対し，結論

に F が含む例を取り出す演算を定義する：

$$\mathcal{I}_F = \{\ (F_i \textbf{ after } A \textbf{ if } \vec{P}) \mid (F_1 \wedge \ldots \wedge F_n \textbf{ after } A \textbf{ if } \vec{P}) \in \mathcal{I} \ \text{かつ}\ |F_i| = F\ \}$$

同様に，アクション名 A を含む例を取り出す演算を定義する：

$$\mathcal{I}^A = \{\ I \in \mathcal{I} \mid I = (\vec{F} \textbf{ after } A \textbf{ if } \vec{P})\ \}$$

定義 4 P を任意のフルーエントとする．このとき P の値を以下で定義する：

$$Val(P) = \begin{cases} T & P \text{ が正のフルーエント} \\ F & P \text{ が負のフルーエント} \\ \times & P \text{ が } \mathit{false} \end{cases}$$

定義 5 フルーエント F **が例** I **を分類する**とは，前提条件に F を含む例を，結論が等しいグループに分けることをいう．

学習アルゴリズムを図 4 に示す．サブルーチン DTL は，ID3 など決定木を学習するアルゴリズムで一般的な手法である．ここでは簡単のため，欠損データやノイズを考慮していない．

4 例

例題 2 に対し，学習アルゴリズムを適用した結果を示す．ここでは，図 2 において遷移先のないアクションは実行不可能なアクションであると仮定した．入力として，アクション g (山羊を連れて対岸にわたる) に関する観測例を 5 つ与えた：

$$\begin{array}{rlll} \overline{M} & \textbf{after} & g & \textbf{if } M \wedge W \wedge G \wedge C \\ M & \textbf{after} & g & \textbf{if } \overline{M} \wedge \overline{W} \wedge \overline{G} \wedge C \\ \overline{M} & \textbf{after} & g & \textbf{if } M \wedge \overline{W} \wedge G \wedge C \\ \mathit{false} & \textbf{after} & g & \textbf{if } \overline{M} \wedge \overline{W} \wedge \overline{G} \wedge \overline{C} \\ \mathit{false} & \textbf{after} & g & \textbf{if } M \wedge W \wedge \overline{G} \wedge C \end{array}$$

学習結果は次のようになった：

$$\begin{array}{lll} g & \textbf{causes} & \overline{M} \ \textbf{if}\ M \wedge G \\ g & \textbf{causes} & M \ \textbf{if}\ \overline{M} \wedge \overline{G} \\ \textbf{impossible} & g & \textbf{if}\ M \wedge \overline{G} \\ \textbf{impossible} & g & \textbf{if}\ \overline{M} \wedge G \end{array}$$

5 つの状態における観測結果から，残りの 5 つの状態に対しても正しい予測をする記述が得られた．

次に，アクション w (狼を連れて対岸にわたる) に関する全ての観測例を入力として与えたところ，次のような学習結果が得られた (**impossible** 命題は省略)：

$$\begin{array}{lll} w & \textbf{causes} & M \wedge W \ \textbf{if}\ \overline{M} \wedge \overline{W} \wedge \overline{G} \wedge C \\ w & \textbf{causes} & \overline{M} \wedge \overline{W} \ \textbf{if}\ M \wedge W \wedge G \wedge \overline{C} \\ w & \textbf{causes} & \overline{M} \wedge \overline{W} \ \textbf{if}\ M \wedge \overline{G} \\ w & \textbf{causes} & M \wedge W \ \textbf{if}\ \overline{M} \wedge G \end{array}$$

この問題には制約を満たさない状態が 6 つあるが，そのような状態における観測結果は与えられないため，元々の記述より多少簡単になっている．我々が文献 [5] で提案した言語 $\mathcal{A}$ の簡単化手続きでは，この種の簡単化は達成できない．

```
CRL(ISET)
入力：観測された例の集合 ISET
出力：決定木の集合 DTS
begin
  DTS = ∅
  AS = ISET に含まれるアクション名の集合
  FS = ISET に含まれるフルーエント名の集合
  for each A ∈ AS
    for each F ∈ FS
      P = ISET_F^A に含まれる最も数の多い結論
      DTS = DTS ∪ DT(A, F, ISET_F^A, Val(P))
end.

DTL(A, F, ISET, D)
入力：アクション名 A, フルーエント名 F
      例集合 ISET, デフォルトの分類 D
出力：決定木 DT = ⟨A, F⟩
begin
  if ISET = ∅ then
    return D
  if すべての例の結論が等しい then
    return Val(結論)
  FS = すべての例の前提条件に含まれる
       フルーエント名の集合
  最も効率良く ISET を分類できる
    フルーエント P ∈ FS を選ぶ
  P を DT のルート (根) とする
  PS = 前提条件において |P| が真である例の集合
  NS = 前提条件において ¬|P| が真である例の集合
  根 P の正の子ノードを DTL(A, F, PS, D) とする
  根 P の負の子ノードを DTL(A, F, NS, D) とする
end.
```

図 4: 因果関係の学習アルゴリズム

5 考察

上で示した実験では，システムが全てのフルーエントを正しく観測できることを仮定していた．例題 2 は 4 つのフルーエントしか含まない小規模なもの

なので，この仮定は妥当なものかも知れないが，一般に，観測結果はノイズや欠損データを含んでいる．そのようなデータに対し，ID3 では，統計的手法を用いて，仮説を生成するアルゴリズムを提案している．本研究でも，そのような手法を取り入れ，より大きな規模の問題で実験し，定量的な評価を行なう必要がある．

また，本研究では入力例のアクション列の長さを1に制限していたが，より一般的にするためには，複数アクションの列も入力できるよう拡張する必要がある．それにより，単一のアクションの因果関係を学習するだけでなく，複数アクションの列をマクロのような形で学習することを考えている．

単一のアクション，複数のアクションの区別なく，因果関係を学習する枠組として，文献 [9] で紹介されている行動ネットワーク (behavior network) の学習や，文献 [7] で紹介されている信念ネットワーク (belief network) の学習などがある．前者は，アクション間の因果関係　あるアクションが実行された後にどのアクションが実行可能になるのかという関係を学習する．後者は，より一般的に，事象間の依存関係を学習する．今後の課題として，これらの研究との関連を明らかにすることが挙げられる．

参考文献

[1] 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).

[2] Hanks, S. and McDermott, D.: Nonmonotonic Logic and Temporal Projection, *Artificial Intelligence*, Vol. 33, No. 3, pp. 379 412 (1987).

[3] 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)).

[4] Lifschitz, V.: Two components of an action language, *Annals of Mathematics and Artificial Intelligence* (1997).

[5] 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).

[6] Quinlan, J. R.: Induction of Decision Trees, *Readings in Machine Learning* (Shavlik, J. W. and Dietterich, T. G.(eds.)), Morgan Kaufmann (1990). Originally published in *Machine Learning* 1:81 106, 1986.

[7] Russell, S. J. and Norvig, P.: *Artificial Intelligence : A Modern Approach*, Prentice-Hall Intl Edns. (1995). (古川 康一 監訳：エージェントアプローチ 人工知能，共立出版 (1997)).

[8] 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).

[9] 山田誠二: リアクティブプランニングにおける学習, 日本ロボット学会誌, Vol. 13, No. 1, pp. 38 43 (1995).