エージェントの信念変更アルゴリズムの実装とその実例

奈良女子大学 理学部 情報科学科 ４回生 新出研究室 鈴木恵里

2009 年 2 月 16 日

概要

Alechina らはエージェント向けの効率の良い信念変更アルゴリズムを与えた [2]．しかしこれには，具体的にエージェントで動く実装は与えられていない．そこで本研究では，その実装を与え，実例による有効性を示すと共に，信念変更の複数の可能性から１つが選択されたとき，その理由の説明を与える機構の追加を行った．なお本研究は，奈良女子大学理学部４回生の荒添との共同研究であり，本論文ではそのうち信念変更の実装とその実例について述べる．

1 はじめに

信念に基づく行為決定を行うエージェントの実現にあたって，信念の変更の実現は重要な意義を持つ．信念変更の操作は，拡張，コントラクション，リヴィジョンに大別される [1]．拡張とは，新情報を信念集合に加えることである．コントラクションとは，新情報を付加することなしに信念集合の中から一部の命題を削除し，一貫性を保つことである．リヴィジョンとは，新情報を受け入れる代わりに信念集合中のから一部の命題を削除し，信念全体に矛盾がないようにすることである．コントラクションやリヴィジョンでは，同じ新情報を得た場合でも更新後の信念状態は複数存在する．従って，そのうちのどれを選択するかを効率よく決定することがエージェントの設計にとって重要となる．

エージェント向けの効率の良い信念変更アルゴリズムとして，Alechina らによるものがある [2]．しかしこれには，具体的にエージェントで動く実装は与えられていない．そこで我々は，エージェント記述言語 AgentSpeak の処理系 Jason の上で，その具体的な実装を与えた．また，更新後の信念状態が複数の可能性から選ばれたとき，なぜそれが選ばれたのかを説明できる機構の追加を行った．

本論文では，我々が与えた信念変更の実装について述べ，また，実例によってその有効性を示す．なお，信念状態の選択の理由を説明する機構の追加については [7] で述べる．

2 信念の変更

2.1 信念変更に対する統合性制約

信念とは特定の人の認識の世界のみで真と仮定されることであり，よって変更が可能である．信念の変更を計算機科学の枠組みで扱うためには次のような問題を明確にしておく必要がある．まず，データベース内の信念をどのように表現するか．すなわち多くのデータベースと同様，ファクト（事実）とルール（推論規則）によって記述されることが期待され，必然的に信念とはある論理で規定される命題の集合となる．ここでは，データベースとは１人のエージェントが信じる命題の集合と考えてもよい．

次に，データベース内でファクトとして記述されていることと，そのデータベース内で推論の結果導き出されたものとどう区別するか．すなわち陽に述べられたデータと暗に導き出されたデータとの間の関係をどう考えるか．このことはのちに信念変更の過程で個々のデータの重要性を論じる際に問題になる．

さらに，信念の変更する方法が一意に決まらない場合，その中の選択はどう行うべきか，実際，どのデータを変更すればよいかは応用分野に強く依存する問題であり一般性を論ずることは困難であるが，いくつかの合理的と思われる指針が統合性制約 [1] という名で以下のように定義されている．

(i). 信念データベースは無矛盾性を保つべきである．
(ii). もし信念データベースからある命題が推論されるならば，その命題そのものもデータベース内にすでに含まれているべきである．
(iii). 信念の変更にともなう変更は最小にとどめるべきである．
(iv). 信念データベース中，ある命題が他の命題よりも重要である（あるいは保護されている）ならば，もっとも重要でない命題を先に変更すべきである．

信念変更とは命題の集合に新しい命題 φ が追加できるかできないかの問題である．追加しないとは $\neg\varphi$ を追加することと同義ではない．信念変更における追加しないという概念は，φ も $\neg\varphi$ も信念にないといっているだけであるが，$\neg\varphi$ を追加することが表すものは φ を拒絶するという状況を意味する．

2.2 コントラクションとリヴィジョンの基本原理

信念変更には情報は無料ではないという暗黙の前提があり，今持っている情報をできる限り保持したいという考えがある．このことは前述の統合性制約 (iii) に述べられており，情報の経済性と呼ばれる．この情報の経済性を定量的に実現するのは一般に困難であるが，この問題意識に立った合理的な基本原理が存在する．これらの基本原理は提案した 3 人の名前のイニシャル Alchourrón - Gärdenfors - Makinson から AGM 原理と呼ばれる．以下にコントラクションとリヴィジョンに関する AGM 基本原理を挙げる．

コントラクションの AGM 基本原理

($\mathrm{K}\dot{-}1$) $\mathrm{K} \dot{-} \mathrm{A} = \mathrm{Cn}(\mathrm{K} \dot{-} \mathrm{A})$ (closure)
($\mathrm{K}\dot{-}2$) $\mathrm{K} \dot{-} \mathrm{A} \subseteq \mathrm{K}$ (inclusion)
($\mathrm{K}\dot{-}3$) If $\mathrm{A} \notin \mathrm{K}$, then $\mathrm{K} \dot{-} \mathrm{A} = \mathrm{K}$ (vacuity)
($\mathrm{K}\dot{-}4$) If not $\vdash \mathrm{A}$, then $\mathrm{A} \notin \mathrm{K} \dot{-} \mathrm{A}$ (success)
($\mathrm{K}\dot{-}5$) If $\mathrm{A} \in \mathrm{K}$, then $\mathrm{K} \subseteq (\mathrm{K} \dot{-} \mathrm{A}) + \mathrm{A}$ (recovery)
($\mathrm{K}\dot{-}6$) If $\mathrm{Cn}(\mathrm{A}) = \mathrm{Cn}(\mathrm{B})$, then $\mathrm{K} \dot{-} \mathrm{A} = \mathrm{K} \dot{-} \mathrm{B}$ (equivalence)

リヴィジョンの AGM 基本原理

($\mathrm{K}\dot{+}1$) $\mathrm{K} \dot{+} \mathrm{A} = \mathrm{Cn}(\mathrm{K} \dot{+} \mathrm{A})$
($\mathrm{K}\dot{+}2$) $\mathrm{A} \in \mathrm{K} \dot{+} \mathrm{A}$
($\mathrm{K}\dot{+}3$) $\mathrm{K} \dot{+} \mathrm{A} \subseteq \mathrm{K} + \mathrm{A}$
($\mathrm{K}\dot{+}4$) If$\mathrm{A} \cup \mathrm{K}$ is consistent, then $\mathrm{K} + \mathrm{A} = \mathrm{K} \dot{+} \mathrm{A}$
($\mathrm{K}\dot{+}5$) $\mathrm{K} \dot{+} \mathrm{A}$ is inconsistent if, and only if, A is inconsistent.
($\mathrm{K}\dot{+}6$) If $\mathrm{Cn}(\mathrm{A}) = \mathrm{Cn}(\mathrm{B})$, then $\mathrm{K} \dot{+} \mathrm{A} = \mathrm{K} \dot{+} \mathrm{B}$

2.3 知識の保護

情報の経済性を尊重し変更を最小にとどめようとするまではいいが，信念データベースのサイズが同等ならどの命題を除去するかということが問題になる．そこで，命題間に重要度をつけておいて，より重要な知識は保護しようとする考え方がある．これを知識の保護（epistemic entrenchment）という．今 2 つの命題 φ と ψ の間で，「ψ は少なくとも φ と同程度には保護されている」という関係を $\varphi \leq \psi$ と書くことにする．特に「ψ は φ より保護されている」場合は $\varphi < \psi$ と書くことにする．

命題間の重要度をつけるには，以下のような基本原理が存在する [1].

(EE1) $\varphi \leq \psi$ かつ $\psi \leq \chi$ ならば $\varphi \leq \chi$（推移性）
(EE2) $\varphi \vdash \psi$ ならば $\varphi \leq \psi$
(EE3) 任意の φ と ψ について $\varphi \leq \varphi \wedge \psi$ であるかまたは $\psi \leq \varphi \wedge \psi$
(EE4) K $\neq K_{\perp}$ のときすべてのすべての ψ で $\varphi \notin$ K のときそのときに限り $\varphi \leq \psi$
(EE5) すべての ψ で $\psi \leq \varphi$ ならば $\vdash \varphi$

3 信念変更アルゴリズム

[2] では，前に述べた信念変更の満たすべき諸性質を満たす効率の良いアルゴリズムを提案している．このアルゴリズムは，信念が有向グラフで表現される場合にのみ利用可能である．エージェントの信念（プランライブラリ）は有向グラフで表現可能なため，このアルゴリズムが有効である．

[2] では，エージェントの信念ベースを有向グラフで表現し，信念と正当化をグラフで表す．また，プランの前後関係を示すための支持リストを保持している．支持リストからは，ある信念を導くために必要な信念や，コントラクションを行うときに矛盾となる信念が読み取れるようになっている．そこで，ある信念を削除するにはプランをたどって，その信念を導く他の信念をも削除することになる．図 1 は，有向グラフで表現された信念ベースの例であり，

(i). （A, [B, C]）… A は B かつ C のとき導かれる．
(ii). （A, [D]） … A は D から導かれる．
(iii). （E, [ ]） … E は [ ] から導かれる．
(iv). （F, [A]） … F は A から導かれる．
(v). （F, [E]） … F は E から導かれる．

であることを示している．(i) より，[B , C]，[D] がある限り再度 A が導かれる．よって，A をコントラクトするだけでなく，B，C のどちらかと D を共にコントラクトしなければならない．ここで，B，C のいずれをコントラクトするかは，リスト [B, C] から最も重要性の低いメンバを選ぶ関数 w があるものとし，それを用いて決める．次に示すアルゴリズムのうち w(s) の部分がそれに該当する．

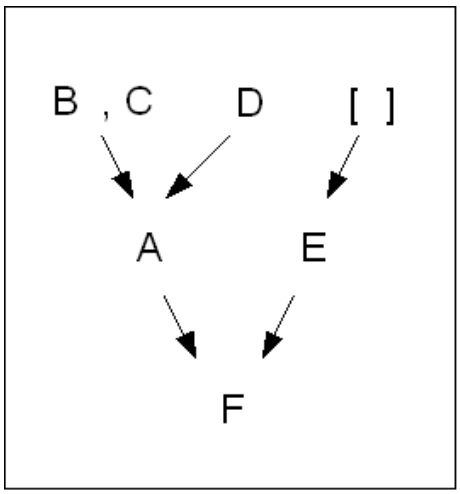

図 1　信念ベースの有向グラフ

以下に信念 A のコントラクションのアルゴリズムを示す.

For each of A ' s outgoing edges
　to a justification (C , s) ,
　remove (C , s) from the graph.
For each of A ' s incoming edges
　from a justification (A , s) ,
　　if s is empty :
　　　remove(A , s) ;
　　else :
　　　contract by w(s) ;
Remove A.

前半では，信念 A から出ていくエッジの管理を行っている．A をコントラクトした際，A から導かれる信念もコントラクトしなければならない．後半では，A に入ってくるエッジの管理を行っている．もし A が空リストから導かれるなら，A のみをコントラクトすれば良い．そうでないなら，A を導く信念がある限りコントラクトを行い，A が導かれないようにしなければならない.

なお，[2] ではコントラクションとして「一貫性スタイル」と「理由維持スタイル」の 2 種類を導入している．前者は AGM 基本原理を満たすものであり，後者は「ある信念が根拠を失ったら，その信念も削除する」というものである．例えば信念 p と q があり，p が q の唯一の根拠となっているとして，q をコントラクトする場合，統合性制約の立場から p は削除しないのが前者，p の根拠がなくなるので p も削除するのが後者である．上述のアルゴリズムは前者の実装をしているが，後者を実装するには，上述の手順の後さらに，入るエッジのなくなった信念を削除すればよい．我々は今回前者の実装を行ったが，後者の実装も容易に追加できる.

以下に信念 A のリヴィジョンのアルゴリズムを示す.

Algorithm : revision by A

Add A to K ;
apply all matching plans ;
while there is a pair (B , B) in K :
　contract by the least preferred member of the pair

まず信念集合 K に信念 A を加え，全プランを適用する．A を加えたことによって矛盾が生じたなら，その矛盾を取り除くために再度コントラクションを行う．

4　Jason における信念変更の実装

4.1　AgentSpeak について

最もよく知られているエージェント開発のアプローチの１つは，BDI（Beliefs-Desires-Intentions）アーキテクチャである．特に，エージェント指向プログラミング言語の中で，AgentSpeak が BDI アーキテクチャに基づいた最も有力で抽象的な言語である．AgentSpeak[3] は，BDI エージェント構築のための論理プログラミング言語であり，BDI エージェントをプログラムするための抽象的なフレームワークを提供する．また，信念（B:Belief），願望（D:Desire），意図（I:Intention）の様相とその時間的変化を記述する論理体系の BDI 論理に基づく．BDI 論理で扱う様相演算子には，信念，願望，意図を表わす BEL，DESIRE，INTEND がある．例えば，BEL(p) は「p を信じている」という解釈をする．同様に，未来方向の時間分岐を表わす時相演算子がある．例えば，AX(p ∨ q) は「全ての未来において，その次の時刻に p または q が成り立つ」を意味する．表 1 に BDI 論理の演算子を示す．

表 1　BDI 論理の演算子

	演算子	意味
様相	BEL	信じている
	DESIRE	望んでいる
	INTEND	意図している
時相	A	全ての未来において
	E	ある未来において
	X(○)	次の時刻に
	G(□)	現在を含み永遠に
	F(◇)	現在を含み未来のいつか
	U	条件が成立するまで

Bratman は著書である「意図と行為」で人間のような合理的な主体の心的状況の中での意図の重要性を指摘した．Bratman の意図の理論を BDI 論理により実現するものに Rao らの BDI アーキテクチャがある．BDI アーキテクチャでは願望世界は信念世界の一部，意図世界は願望世界の一部であるような分岐時間可能世界モデルを定式的に用いている．Rao らはこの世界で，

(i). 願望を達成したという信念を持つまで
(ii). 願望を達成できる手段があるという信念を持っている限り

(iii). 願望を達成する状況でなくなるまで

などの条件を組み合わせて，幾つかの意図持続のタイプを提案している．

4.2 Jason について

Jason[4] は AgentSpeak の拡張版のためのインタプリタであり，AgentSpeak の操作上の意味論を実装している．また，相互エージェントコミュニケーションに基づくスピーチアクトを含んでいる．SACI[5] を使用することにより，Jason マルチエージェントシステムは，容易にネットワーク上で共有できる．BDI システムには他にも様々な実装が存在しているが，AgentSpeak は BDI アーキテクチャの理論に基づいているという重要な特性がある．また，他の BDI エージェントとの比較における Jason のもう 1 つの重要な特性は，それが Java で実装されており，利用可能なオープンソースであるということである．我々は今回の実装にこの Jason を用いた（図 2）．

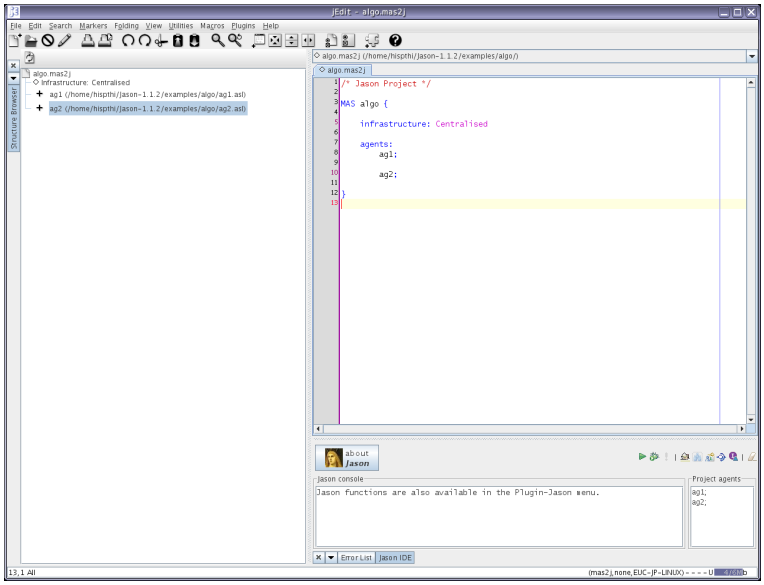

図 2　Jason のプラットホーム

4.3 実装方法

信念変更アルゴリズムでは，プランの前後関係を示すための支持リストを保持している．Jason は論理型言語である AgentSpeak を使用しているため，信念のデータ構造は支持リストと複合項を用いることにより実現できる．図 1 の有向グラフを元に信念ベースを示すと以下のようになる．

```
data(A, [B, C]).
data(A, [D]).
data(E, [ ]).
data(F, [A]).
data(F, [E]).
```

信念の拡張は，Jason の「+」オペレータを使用することにより実現できる．以下のように表記することにより，「+」の後ろに書いた信念の拡張ができるため，信念 B あるいは C から信念 A を導くことができるという信念の拡張を行ったことになる．

```
+data(A, [B, C]).
```

信念の削除は，追加と同様に Jason の「-」オペレータを使用することにより実現できる．

```
-data(A, [B, C]).
```

この削除の機能を利用し，図 1 における信念 A のコントラクションを以下に示す．信念のコントラクションを行うためには，contraction1 のゴールを呼び出すことによって達成できる．そのために，削除された信念を deletion の引数とし，contraction1 を呼び出す．

```
/*deletion*/
+deletion(A1)[source(P)]
    <-  .print(A1, " is deleted.");
        .send(P, tell, contraction1(A1)).
```

contraction1 では，第 2 引数が A である信念に着目することで，信念 A から派生される信念の削除を行っている．図 1 の例では信念 F がそうであるため，data(F, [A]) の削除を行う．

```
/*contraction1*/
+contraction1(A1)[source(P)]: data(D1, [D2, D3]) & D2 = A1 | D3 = A1
    <-  -data(D1, [D2, D3]);
        .print("-data(", D1, ",[", D2, ",", D3,"])");
        +contraction2(A1).

+contraction1(A1)[source(P)]: data(D1, [D2]) & D2 = A1
    <-  -data(D1, [A1]);
        .print("-data(", D1, ",[", A1,"])");
        +contraction2(A1).

+contraction1(A1)[source(P)]: data(D1, [D2])
    <-  +contraction2(A1).
```

contraction1 では，信念 A から派生される信念の削除のみしか行っていないため，次に信念 A を派生する信念の削除を行うための contraction2 を呼び出す．contraction2 では，第 1 引数が A である信念に着目することで，信念 A を派生する信念の削除を行っている．信念 A が空リストから派生する場合は，信念 A の削除のみを行う．それ以外の場合は，信念 A を派生する信念を全て削除しなければ，再度信念 A が導かれてしまうため，第 2 引数について再度コントラクトを行う．そのためには，contraction2 の第 2 引数を，contraction1 の引数として与えることにより実現できる．図 1 の例では，B と C のいずれかのコントラクト，および D のコントラクトを再帰的に行う．B と C のいずれをコントラクトするかは，我々の実装では，信憑度を設けておいてそれの 1 番低いものを選ぶことで決めるようにした．なお，contraction3 は，引数の微調整を行って contraction2 を呼び出すだけの述語であるので，ここでは省略する．

```
/*contraction2*/
+contraction2(A1): data(D1, D2) & D1 = A1 & D2 = []
    <-  -data(A1, D2);
        .print("-data(", A1, ",", D2,")").
```

```
+contraction2(A1): data(D1, [D2, D3]) & D1 = A1
    <-  -data(A1, [D2, D3]);
        .print("-data(", A1, ",[", D2, ",", D3, "])");
        +contraction3(A1);
        +contraction1(D2).

+contraction2(A1): data(D1, [D2]) & D1 = A1
    <-  -data(A1, [D2]);
        .print("-data(", A1, ",[", D2,"])");
        +contraction3(A1);
        +contraction1(D2).
```

図 1 の例題で信念 A を削除した場合の Jason でのプロジェクト実行結果を図 3 に示す.

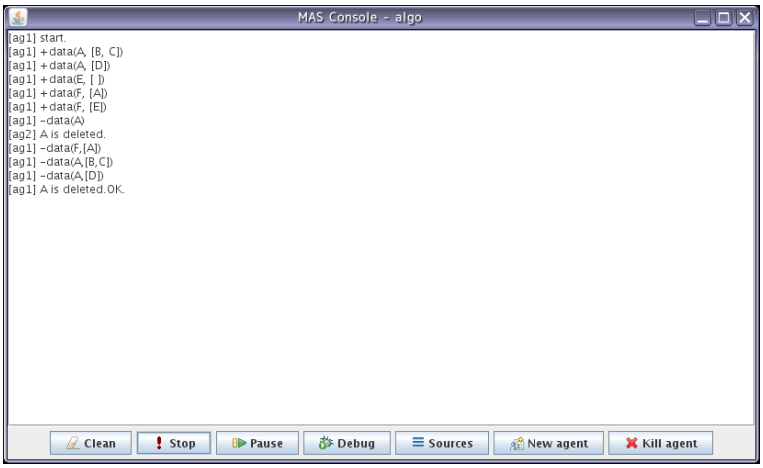

図 3　実行結果

リヴィジョンの操作は，拡張とコントラクションの組み合わせにより実現できる.

5　例題の作成

実装した信念変更アルゴリズムの有効性を示すために，エージェントがネットショッピングを行う例を作成した[*1]．流れとしては，買い手が欲しいと思っている物と，売り手が公開している商品とを比較し，一致している場合は売り手の信頼性を見極め，商品を購入する．エージェントは，商品を購入するというゴールを目指して行動する.

エージェントとしては，買い手と売り手が必要となるため，構成ファイルは以下のようになる.

```
/* Jason Project */
MAS netshopping{
  infrastructure: Centralised
  agents:
    purchaser;
```

[*1] この例題は，実装の試作段階として作成した例であるため，4.3 節で示した実装と，述語名などが同じではない．しかし，アルゴリズムは同じであるので，実装の有効性を十分示すものにはなっている.

```
    seller;
}
```

買い手は 8000 円位の白い服が買いたいという信念を持っている．また，売り手は 7000 円位のグレーの服を在庫に持っているという信念を持っている．買い手は商品情報の比較を行い，購入するかどうかの判定を行う．両者の信念を表 2 に示す．

```
purchaser : my("M", 8000, white, casual).
seller : data("M", 7000, gray, casual).
```

表 2　商品情報の比較

エージェント	サイズ	値段	色	ジャンル
買い手	M	8000	白	カジュアル
売り手	M	7000	グレー	カジュアル

判断基準としては，

(i). 商品のサイズとジャンルが一致している
(ii). 色が異なっても，希望価格より安い

である．以上の条件を満たしているとき，買い手は「その商品が欲しい（want)」という信念を拡張する．

```
//*********(examination)*********//
+examination(S, P, CO, J, C)[source(A)]:
  C>0 &
  my(Ms, Mp, Mco, Mj) &
  S = Ms &
  P < Mp &
  J = Mj |
  CO = Mco
  <-  .print("Commodity wanting it.");
      .print("Size is ", Ms);
      .print("Price is ", Mp);
      .print("Color is ", Mco);
      .print("Janle is ", Mj);
      .print("The color may be different. ");
      +want(C);
      .print("I want this goods.");
      .send(A,tell,eseller).
```

「欲しい（want)」という信念が拡張されると，次に買い手は，売り手が信頼できるかどうかの判定を行う．売り手にはそれぞれ１０段階で信頼度を与えている．今回はその信頼度が８以上であれば，「その売り手は信頼できる（trust)」という信念を拡張する．信頼度が８未満の場合には，先程拡張した「欲しい（want)」という信念を削除し，「その売り手は信頼できない（~trust)」という信念を拡張する．

```
//*********(value) >= 8*********//
+value(E, C)[source(A)]: C>0 & E>=8 & E<=10
  <-  +trust(A);
      .print(A, " can be trusted.");
```

```
      .send(purchaser,tell,decision(C)).

//*********(value) < 8*********//
+value(E, C)[source(A)]: C>0 & E<8
  <-  -want(C);
      +~trust(A);
      .print(E, " is low value!");
      .send(purchaser,tell,decision(C)).
```

以上の行動を繰り返し，「欲しい（want)」かつ「信頼できる（trust)」の場合にのみ，商品の購入を行う．

```
//*********(decision)*********//
+decision(C)[source(seller)]: want(C) & trust(seller)
  <-  +buy(C);
      .print("It bought.");
      .send(seller,tell,bought(C)).
```

エージェントがネットショッピングを行う例題で，上記に示した行動をとり，商品を購入した場合の実行結果を図 4 に示す．また，デバッグ画面（図 5）では最終的に残る信念を確認することができる．

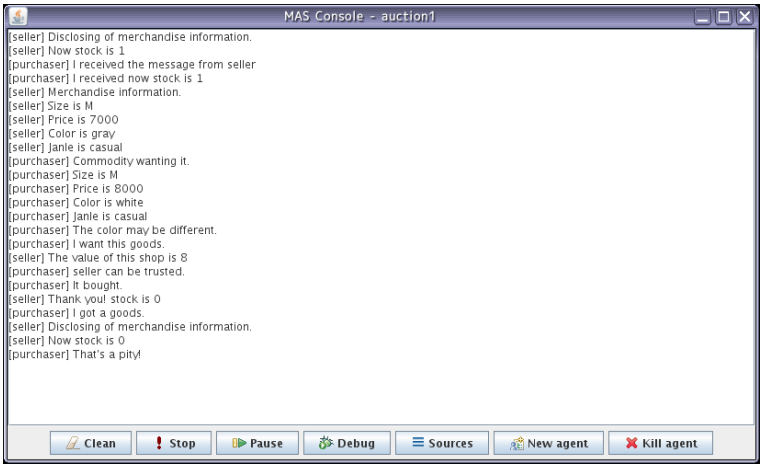

図 4　実行結果

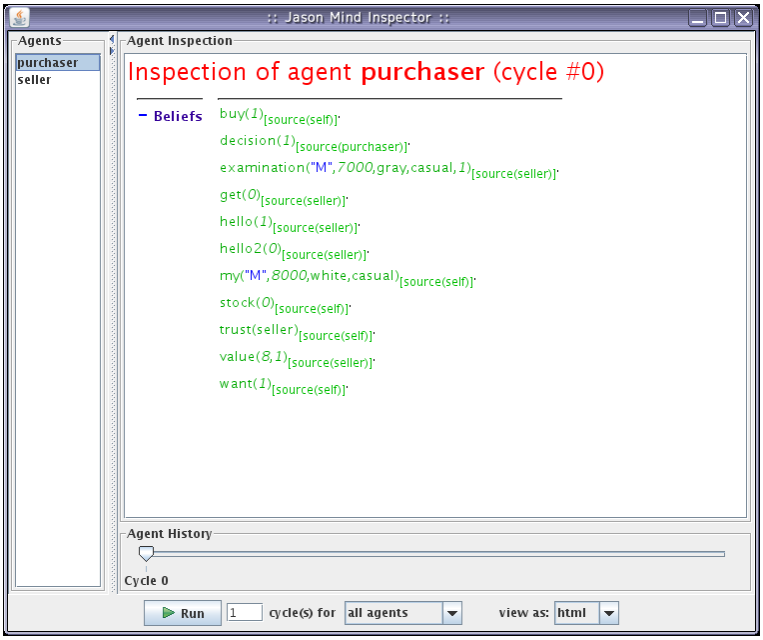

図 5　デバッグ画面

6　おわりに

本研究では，信念変更アルゴリズムの実装を与え，実例による有効性を示すと共に，信念変更の複数の可能性から 1 つが選択されたとき，その理由の説明を与える機構の追加を行った．今後はこの機構を実際的なエージェントに組み込んで利用し，改良に関する知見を積んでいきたい．

参考文献

[1] 東条敏: 知の科学 言語・知識・信念の論理, オーム社 (2006).

[2] Natasha Alechina, Rafael H. Bordini, Jomi F. Hubner, Mark Jago and Brian Logan: *Automating Belief Revision for AgentSpeak*, *Proc. of DALT2006*, pp. 1–16(2006).

[3] Anand S. Rao: *AgentSpeak(L): BDI Agents speak out in a logical computable language*, *Proc. of MAAMAW-96*, pp. 42–55(1996).

[4] Rafael H. Bordini and Jomi F. Hubner: *Jason: a Java-based interpreter for an extended version of AgentSpeak*, `http://jason.sourceforge.net/JasonWebSite/Jason%20Home.php`.

[5] Hubner, J. F. , Sichman, J. S. : *SACI Programming Guide*, *Technical report, Universidade de Sao Pãulo*, (2000).

[6] Rafael H. Bordini, Michael Wooldridge and Jomi F. Hubner: *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley series in Agent Technology)*, (2007).

[7] 荒添理央: エージェントの信念更新に関する理由説明機能の作成について, 奈良女子大学理学部情報科学科 2008 年度卒業論文, (2009).