Vol. 43 No. 5 情報処理学会論文誌 May 2002

議論を計算とコミュニケーションの基本メカニズムとするエージェントシステム

梅田勇一[†] 沢村一[††]

エージェント指向コンピューティングの世界では，それぞれのエージェントが各自の持つ情報を生かしながら協調・合意して問題解決にあたることが求められている．本論文では，議論の導入がこの問題に有効であるとの考えのもとで，次のような機能を持つエージェントシステムを提案し，実際にネットワーク上で現実の問題に適用して有効性を示す．(1) 複数のエージェントが各自の知識ベースをもとに議論・反論を行う．(2) 反論に行き詰まったら，相手の議論への補強を考えることによって協調を試みる．(3) この2つを行っても結果が定まらないとき，弁証法的な合意形成を行う．

An Agent System with Argumentation as a Basic Mechanism of Computation and Communication

YUICHI UMEDA[†] and HAJIME SAWAMURA [††]

In the upcomming networked society, it is desired that several computers on network can resolve conflicting problems or make better solutions through argumentation. In this paper, we propose a novel approach to agent systems where several agents communicate, argue with each other, reinforce other arguments for cooperation and finally make a dialectical agreement through argumentation from distributed knowledge bases. By applying it to a variety of application, we show the potential and practical usefulness of the system.

1. はじめに

一般に情報処理システムが扱う情報は，刻々と変化し流動している．そのため，完全な情報というものは存在せず，無数の不完全な情報が分散した状態となっている．このため，情報の不十分さや矛盾から衝突が生じる．

このように互いの持っている知識が対立している場合，我々は話し合いや交渉を行い，そのうえで相手を説得したり自分が妥協したりするなどして合意に至り，問題を解決したり協力を得たりすることが多い．また，単に対立を解消するだけではなく，相手の発言から何らかの収穫があったり，時には話し合いの中から新しいアイディアが生まれたりといった効果もある．

一方，情報システムの分野では，合理的なエージェントの実現を目的とした研究が多数行われている．ここでは，異なる情報を持ったエージェントどうしが先に述べたような情報の衝突を回避したり，互いの情報を生かした協調動作を行ったりして，問題を解決することが期待されている．

本論文で我々は，エージェントによる合意形成の手段として議論を導入する．一般に人間が行う議論の形式は多様であるが，本論文では特に「主張が衝突し，かつ結論と前提が明確である場合，互いに論争し衝突を解決することによって合意に至る」という形式の議論に着目する．そのうえで，より柔軟な合意形成や協調のための新しいアプローチを示す．さらに，既存の議論に関する研究 1), 3) では触れられていなかった分散ネットワーク環境への応用を試み，分散環境下での新しい総合的交渉手段とすることを目指す．

本論文では，エージェントシステムに計算メカニズムとして議論を導入し，次のように拡張することによって衝突回避・協調動作・合意形成の3つの目的へ応用できることを示す．

(1) 異なる知識を持つ複数のエージェントどうしが，意思決定のために議論を行う．その際，単に反論を返し合うのではなく，自分の目的に有利になるように戦略を用いて議論を作成する．

† 新潟大学大学院自然科学研究科
Graduate School of Science and Technology, Niigata University
†† 新潟大学工学部情報工学科
Department of Information Engineering, Faculty of Engineering, Niigata University

(2) 反論できなくなった場合は単に引き下がるのではなく，相手の議論を補強する議論の作成を考えることによって協調を試みる．

(3) 議論後の結果が定まらないとき，弁証法的な合意案の作成を試みることによって，より柔軟に解を得ることができる．

本論文では，知識表現・議論の作成・反論の作成については，文献4)の内容に基づいている．そこで，2～4章でその概念について説明し，5章以降で我々独自の拡張について述べる．

以下，2章では各エージェントの持つ知識の表現方法について述べる．3章ではエージェントが作成する議論の定義について述べる．4章では他のエージェントが提出した議論への攻撃方法について述べる．5章では，このシステムで扱うエージェントモデルと，エージェントが議論提出時にとる戦略について説明する．6章では，システムが衝突回避・協調動作・合意形成問題の解決に適用できることを示す．7章では，複数のエージェントが議論を行うためのプロトコルについて述べる．8章ではシステムの実装について述べる．9章では，議論システムを実在する問題に適用し評価を行う．

2. 知識表現

本論文では，各エージェントが議論の背景知識として非共有の知識ベースを持っており，かつ矛盾した知識が混在していることを前提とする．ただし，議論として提出する知識は無矛盾でなければならない．

2.1 2種類の否定と規則

本論文では，知識表現に完全/不完全の2種類の否定を持つ拡張論理プログラミング言語を導入している．不完全否定を併用することで，システムは根拠のない不確実な情報をも取り扱うことができる．また，我々が提案する協調のための足掛かりとなる（詳しくは6.2節で述べる）．以下，2種類の否定について述べる．

原子命題Pについて，Pの完全否定（…ではない）を強否定と呼び，“¬ P”で表す．また，Pの部分否定（…とは限らない）を弱否定と呼び，“～ P”で表す．以下，“～ P”と“～ ¬ P”の形をとるリテラルを弱リテラルと呼び，“¬P”の形をとるリテラルを強リテラルと呼ぶ．本論文では“¬ ～ P”という形のリテラルについては考えない．

弱リテラル ～Pは，仮定 ¬Pを持つ．これは，一般に「Pとは限らない」というときにはPでない可能性の仮定が含まれているからである．同様に，～ ¬ Pの仮定はPである．

この弱リテラルと強リテラルを用いて，規則形式の知識を次の形で定義する[4]．

$$L_0 \Leftarrow L_1 \wedge ... \wedge L_j \wedge \sim L_k \wedge ... \wedge \sim L_n$$

ここで，L_i はすべて強リテラルである．また，∧は「かつ（and）」を表している．右辺が前提，左辺が結論であり，「右辺ならば左辺」という意味になる．前提は空でもよい．

本論文で用いる知識ベースは，この規則の集合によって記述されているものとする．

2.2 規則の種類

規則は絶対規則（strict rule）と撤回可能規則（defeasible rule）の2種類を定義する．

絶対規則は一般法則・事実など，確実なことを表す．確実なことを述べた規則であるので，弱リテラルは持たない．議論の余地のない知識はこの規則で記述する．撤回可能規則と区別するため，左辺と右辺を結ぶ矢印に ← を用いて表す．

撤回可能規則は個人の主張・意見などを記述する．確実なことを述べているわけではないので，前提部分に弱リテラルを持つことができる．絶対規則と区別するため，左辺と右辺を結ぶ矢印に ⇐ を用いて表す．

3. 議論

ここでは，知識ベースから作成される議論の定義を与える．

次の条件を満たす規則の列 $[r_0...r_n]$ を議論と定義する[4],☆．

(1) 各規則の前提にある強リテラルはそれぞれ，自分よりも前にある規則の結論に現れている（この「前にある規則」のことを以後「根拠になる規則」と呼ぶ）．
弱リテラルは確実でないことを表現しているので，根拠になる規則は必要ない．

(2) 1つの議論に同じ結論を持った2つ以上の規則は含まれていない．

たとえば，次に示す規則の列Aは議論である（r_2 のaはすでに r_1 の結論に現れているので根拠になる規則がある．cも同様．b，dには根拠になる規則は不要）．

☆ 以後の文章では「いくつかの意見を交わし，論じ合う行為としての議論（argumentation）」と「本論文で規則の列として定義されている議論（argument）」の2つに同じ「議論」という語が当てられているため，一部混同しやすいところがあるが，「議論する」「議論を行う」など，動詞的な表現は前者の意味であり，「エージェントが議論を作成する」「エージェントが議論を提出する」など，名詞的な表現は後者の意味を意図したものである．

$$A = \left[\begin{array}{lll} r_1: & a & \Leftarrow \\ r_2: & c & \Leftarrow \quad a \wedge \sim \neg b \\ r_3: & e & \Leftarrow \quad c \wedge \sim d \end{array} \right]$$

上の定義によって作られた議論の仮定と結論を次のように定義する．

(1) 議論中に含まれている弱リテラルが持つ仮定（2.1 節参照）の和集合が議論の仮定である．

(2) 議論中に含まれている各規則の結論の和集合が議論の結論である．

先に示した議論 A の場合，仮定は {b, $\neg d$} で，結論は {a, c, e} である．

4. 攻撃方法

各エージェントは，相手が提出した議論に対し，対立する議論を作成し攻撃することができる．以下，攻撃方法について述べる．

4.1 反論と無効化

結論に L を持つ議論（の規則）は次の 2 種類の方法で他の議論（の規則）に攻撃することができる[4]．

(1) 結論に $\neg L$ を持つ議論（の規則）に攻撃できる．これを反論（rebut）という．

(2) 仮定に $\neg L$ を持つ議論（=前提部に $\sim L$ を持つ規則を含む議論）に攻撃できる．これを無効化（undercut）という．

たとえば，次の議論 B と C を考える．

$$B = \left[\begin{array}{lll} r_1: & a & \Leftarrow \\ r_2: & b & \Leftarrow \quad a \end{array} \right]$$

$$C = \left[\begin{array}{lll} r_3: & \neg a & \Leftarrow \quad \sim b \end{array} \right]$$

B は C を反論することも無効化することも可能である．一方，C は B に反論することが可能である．

4.2 defeat の定義

2 つの議論 A_1 と A_2 が以下のいずれかの条件を満たすとき，A_1 は A_2 を defeat しているという[4]．

(1) A_1 が A_2 を無効化している場合．

(2) A_1 が A_2 に反論していて，A_2 が A_1 を無効化していない場合．

また，A_1 が A_2 を defeat していて，A_2 が A_1 を defeat できないとき，A_1 は A_2 を完全に defeat (strictly defeat) しているという．以後,「議論を攻撃する」とは,「相手の議論を defeat できる議論を作成し提出する」ことを指す．

4.1 節の議論 B，C について考えると，B は C を反論したとしても無効化したとしても条件のいずれかを満たすことになり，C を defeat しているといえる．一方，C は B に反論することができるが，そのときは B から無効化されているので条件 (2) を満たせず，B を defeat しているとはいえない．よって，B が C を攻撃したときには完全な defeat になる．

4.3 正当化と却下

ここまでの定義を用いて，我々は議論の正当化を次のように定義する[9]．これは議論の最終状態を決めるものである．なお，文献 4) でも不動点意味論を用いて同様の定義がなされていることを書き記しておく．

- エージェント X の議論 A は，A を攻撃するすべての議論が X の議論によって完全に defeat されているとき，正当化（justified）されているという．
- 正当化された議論から攻撃されている議論は却下（overruled）されているという．

いい替えると，相手からの攻撃をすべて退けた場合（異議のない場合）に議論は正当化される．

5. 議論するエージェントのモデル

ここでは，まず我々のエージェントのモデルについて述べる．次に戦略の基礎概念となる「攻撃を受けにくい議論」を定義したうえで，目的に応じた議論の提出戦略について述べる．

5.1 議論するエージェントの構成員

本論文ではエージェント群を，自分の主張を押し通そうとしているエージェント群，その主張に反発するエージェント群，議論を仲介するエージェントの 3 つに分類している．この構成についてまず定義する[9]．

発案者（proponent）側エージェント 議題に沿った議論を作り（これを本論と定義する），本論への反論・異議をすべて退けることで正当化することを目的とする．以下，$P_1, \ldots, P_n$ で表す．

対立者（opponent）側エージェント 発案者側の本論が正当と見なされることを防ぐため，攻撃することを目的とする．以下，$O_1, \ldots, O_n$ で表す．

仲介者（mediator）エージェント 双方が作った議論の送信仲介（7 章参照）と，議題ごとの結果の管理（下記参照）を行う．また，議論過程がすべて終了したとき，最終的な合意案を提案する．

ここで，議題は $A\&B\&C\&\ldots$ の形で与えられる．A,B,C, ... のそれぞれを連言肢と呼ぶ．これは強リテラルである．

仲介者エージェントは，上の形で与えられた議題を A,B,C,... に分解し，最初に A を発案者側エージェントに与える．発案者側エージェントはこれを結論とした本論を作り，これを正当化すべく対立者側の反論と

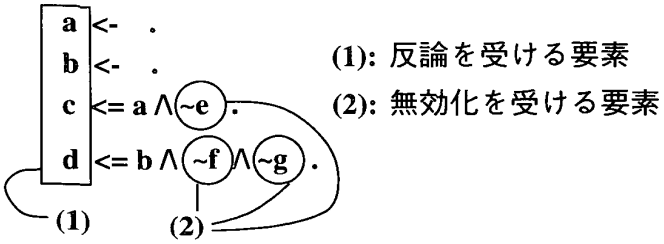

図 1 議論が攻撃を受ける要素
Fig. 1 The literals that may be attacked.

戦う．結局は反論が尽きた時点で，本論は正当化または却下のいずれかの状態になる（4.3 節参照）．同様に，B について，C について ... と議論が行われる．

最後に，仲介者エージェントは一連の結果から最終合意案を生成する．この合意案の生成については 6.3 節で説明する．

5.2 反論を受けにくい議論

発案者側エージェントの目的は本論を正当化することであるが，その定義から反論を受けにくい議論の方が正当化されやすいといえる．そこでここでは反論を受けにくい議論について定義する．

3 章の定義に従って作られた議論は，結論を多く持っているほど，その部分を相手から反論される恐れがある（図 1 の (1)）．また，仮定を多く持っているほど，その部分を相手から無効化される恐れがある（図 1 の (2)）．

そこで本論文では，仮定と結論の要素数の合計が少ないほど，反論を受けにくい議論であると定義する[8]．

5.3 議論の提出戦略

ここでは，それぞれのエージェントが議論を提出する際にとる戦略を与える．本論文では，議論の形式から判断する戦略のみを取り入れ，議論の内容から判断する戦略については今後の課題とする．

- 発案者側は，相手の反論を退ける目的から，最も反論を受けにくい議論を提出する．ただし，議論がループに陥るのを防ぐため，過去に提出したことのある議論は使わない．
- 対立者側は，より多くの反論を行うという目的から，過去に用いた議論を繰り返すことを許す．

なお文献 4) では，法的議論（法廷論争）を対象としていることから，我々のものよりも発案者に厳しい提出戦略を与えていることを書き記しておく．

6. 議論するエージェントの問題解決法

この章では，我々の議論するエージェントシステムを衝突回避・合意形成・協調動作の 3 つの問題に適用する方法を与える．

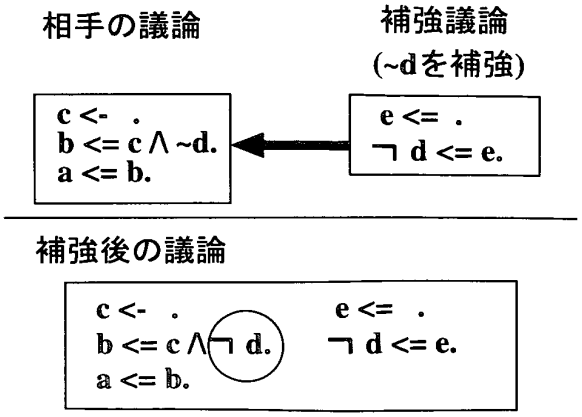

図 2 補強議論
Fig. 2 The reinforced argument.

6.1 衝突回避

提案者側エージェントはここまでで述べた方法によって議論することで自論の正当性を示し，矛盾する案を却下して衝突を回避できることになる．そこで，次のことを定義する．

- すべての連言肢について提案者側が本論を正当化できた場合，議題すべてについて異議なしと見なし，その議題について合意が成立する．

異議を退けられなかった場合，すなわち議題の一部または全部が却下された場合には，双方の主張をなるべく尊重するため，弁証法的合意形成を試みる．詳しくは 6.3 節で述べる．

6.2 協調

エージェント間で合意し目的が一致した以上は，協調的な動作をとることが望ましい．本論文のシステムでは，反論が尽きて提案者側の議論が正当化された場合に，対立者側が自らの持っている情報を相手の議論に付け加え，補強を試みることで協調を実現する[9]．

協調は，相手の議論から弱リテラル（議論の不完全な部分）$\sim L$ を探し，そこに自分の知識ベースから $\neg L$ を結論とした議論（以下，補強議論と呼ぶ）を作って $\sim L$ と置き換えることで実現する（図 2 参照）．

協調要求を受ける提案者側は，補強を受けることによって自分の議論の不完全な部分を減らすことが期待できる．ただし，補強を受けた場合に自分の議論の弱リテラル部分が増えてしまう場合は，かえって議論が弱くなってしまうので協調要求を拒否する．逆に協調を申し出る対立者側は，補強議論が複数作成できる場合，弱否定部分が少ない方から提出する．

なお，提案者側は協調は行わないが，弁証法的合意形成（6.3 節参照）のプロセス中に攻守交代して再度議論する場合があり，このときには協調する可能性がある．

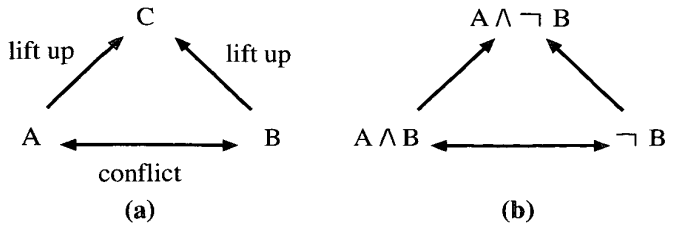

図 3 高次の合意 (a) と弁証法的合意 (b)
Fig. 3 (a): Aufheben / (b): A dialectical agreement.

6.3 弁証法的合意形成

ここでは，弁証法的☆ 合意形成を定義する[6),11)]．

まず，定義に必要となる Aufheben の概念について説明する．

2 つの相容れない命題 A，B が与えられ，$A \to C$ および $B \to C$ のいずれも成り立たないとき，命題 C を Aufheben（高次の合意，止揚）という（**図 3**(a) 参照）．ただし C は A または B と命題変数を共有していなければならない．

この Aufheben の概念を用いて，弁証法的合意形成を次のように定義する．

議題を $J_1\&\ldots\&J_n\&O_1\&\ldots\&O_n$ として議論を行った結果，連言肢 $J_1,\ldots,J_n$ については正当化され，連言肢 $O_1,\ldots,O_n$ については正当化されなかったとする．このとき，仲介者エージェントが $\neg O_1,\ldots,\neg O_n$ について対立者側に議論させる（攻守交代して議論することになる）．対立者側がこれらをすべて正当化できたら，$J_1,\ldots,J_n$ についても $\neg O_1,\ldots,\neg O_n$ についても異議がないことから，この結果に先の Aufheben を適用し $J_1\&\ldots\&J_n\&\neg O_1\&\ldots\&\neg O_n$ を結論とする．正当化できない連言肢 O_i があれば，O_i と $\neg O_i$ が両方とも正当だといえないことから，合意なしとして終了する．

たとえば，議題が $A\&B$ で A が正当化され B が正当化されなかった場合，弁証法的合意形成は図 3(b) のようなプロセスとなり，対立者側が $\neg B$ を正当化できれば最終的な合意案は $A\&\neg B$ となる．この具体例については 9.2 節で触れる．

7. 議論プロトコル

本論文で提唱するシステムは，単にエージェントが 1 対 1 で対立する場合だけでなく，n 対 n で対立する場合においても 6 章で述べたような議論・協調を試み，弁証法的合意を形成することができる．これによりネットワーク上の多くのエージェントが議論に参加できる．この章ではそのための議論プロトコルを全体の実行の流れに沿って述べる．

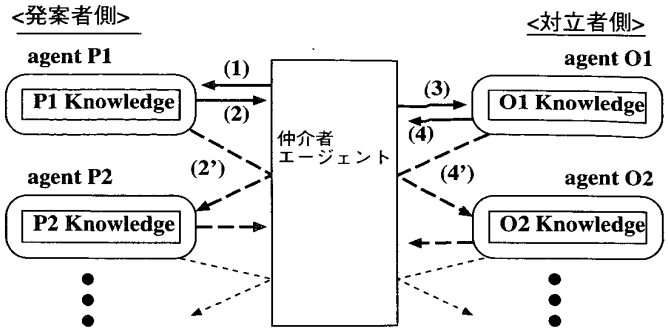

図 4 議論の流れ
Fig. 4 The flow of argumentation.

ここでは簡単のため，「2 人の発案者対 2 人の対立者」が仲介者エージェントの仲介で議論する場合を例にとって説明する．

以下，発案者側のエージェントを agentP_1，P_2，対立者側のエージェントを agentO_1，O_2 とする．また，発案者側は $P_1 \to P_2$ の順に発言権が与えられ，いずれか 1 つのエージェントが発言した時点で相手側に発言権が移るものとする．対立者側も同様である．

(1) ユーザがシステムを起動させ議論開始指令を送ると，まず仲介者エージェントが議題を連言肢に分割する（5.1 節参照）．そして最初の連言肢を最初の議題として agentP_1 に議論開始メッセージを送り，議論の作成を促す（**図 4** の (1)）．

(2) agentP_1 は議題を受け取ると，5.3 節の戦略に基づき議題に沿った議論を自分の知識ベースから作成する．これが本論となる．もし作成できたら，その議論を仲介者に返し手順 (3) へ進む（図 4 の (2)）．本論を作成することができなかったときは，議題を仲介者に返し，これを次の順番の発案者側エージェント（ここでは agentP_2）に渡してもらう（図 4 の (2′)）．次の順番の発案者側エージェントも同様に議論の作成を試み，できなければさらに次の順番のエージェントに発言権を譲る（以下同様）．

全員が本論を作成できなかった場合は，その議題は却下され，仲介者エージェントが次の議題（連言肢）を用意する（**図 5** の (b)）．

以下で展開される議論過程は，ここで提出された本論が正当といえるか否かという点について争う．

(3) 仲介者は，発案者側から受け取った議論を対立

☆ 弁証法は，一般的には “対話的思考方法，または対話による真理探求方法” といわれているが，様々な見解があり，1 つに定まった定義はない．
本論文では紙面の都合上，詳細についての説明を割愛する．これについては，文献 6), 11) を参照されたい．

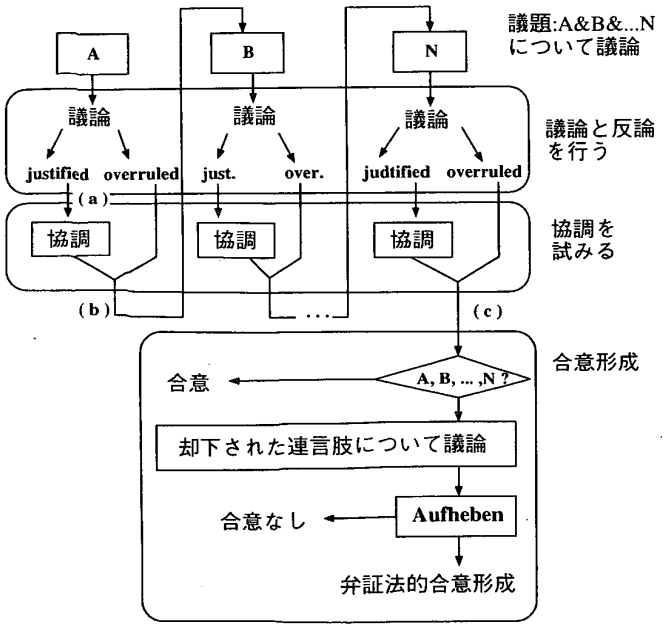

図 5 システム全体の流れ

Fig. 5 The overall flow of the argument-based agent system.

者側に送信し，反論を促す（図 4 の (3)）．

(4) 対立者側で順番が最初のエージェント agentO_1 は，5.3 節の戦略に基づき，この議論を攻撃できる議論を自分の知識ベースから作成することを試みる．もし作成できたら，その議論を仲介者に返し手順 (5) へ進む（図 4 の (4)）．
作成できなければ，送られてきた議論をそのまま仲介者に返し，次の順番の対立者側エージェント（ここでは agentO_2）に渡してもらう（図 4 の (4′)）．次の順番の対立者側エージェントも同様に攻撃を試み，できなければさらに次の順番のエージェントに発言権を譲る（以下同様）．対立者側エージェントの全員が攻撃に失敗した場合は，提案者側の議論にこれ以上攻撃する手段がないということになり，本論は正当であったとして議論過程はいったん終了する．
その後，agentO_1 から順に協調（6.2 節参照）の機会が与えられる（図 5 の (a)）．誰かが補強議論を作り提案者側に受け入れられれば，補強後の議論が正当であるとして協調の機会は終了する．受け入れられなければ成功するまで他の対立者側エージェントが順に協調を試み，全員が協調に失敗した場合は協調終了となる．いずれの場合もその議題についての議論過程は終了となるので，仲介者が次の議題（連言肢）を用意する（図 5 の (b)）．

(5) 仲介者エージェントは，対立者から受け取った議論を発案者側に送信し，反論を促す．

(6) 発案者側は受け取った議論を調べ，5.3 節の戦略に基づいてこれを攻撃できる議論を自分の知識ベースから作成することを試みる．作成できれば，その議論を仲介者に返し手順 (3) へ進む．作成することができなければ手順 (2) と同様，自軍の誰かが議論が作成できるまで後続のエージェントに順番を譲り続ける．ここで全員が攻撃に失敗した場合は本論が却下され，その議題についての議論過程は終了となる．この場合も仲介者が次の議題（連言肢）を用意する（図 5 の (b)）．

(7) 以下，いずれかが相手を攻撃する議論の作成に失敗するまで手順 (3)〜(6) を繰り返す．本論が正当化または却下されたら，仲介者エージェントは結果を記録し，次の連言肢を新しい議題として agentP_1 に送信する．以後，手順 (2) に戻り，新しい議題について再度議論を始める (5.1 節参照)．
すべての議題について議論過程が終了した場合（図 5 の (c)），全議題が正当化されていればそのまま合意となる（6.1 節参照）．そうでなければ却下された議題の否定について再度議論を行い，その結果に弁証法的合意形成を適用する (6.3 節参照)．最終結論が決まったところでシステムは終了する．

8. 実　　装

この章では，ここまでで述べてきたシステムの実装について説明する．

8.1 議論エージェント

各議論エージェントは Prolog で構成された推論エンジンを持っている．推論エンジンが行うのは主に以下の 3 つである．

(1) 本論（最初に提案する議論），または相手を攻撃できる議論を知識ベースから抽出し作成する．

(2) どのような種類の議論を作成したか（またはできなかったか）の状態を出力する．

(3) 作成した議論を英文または木構造に変換して出力する．

(1) の議論の抽出は，主張すべき要素（議題，または相手を攻撃できる要素）が含まれた規則を検索し，以下その規則の根拠となる規則を再帰的に検索し組み込んでいくことで行われる．組み込んだすべての規則について，検索した根拠が事実または弱リテラル（それ以上の根拠不要．3 章参照）となったら議論作成成功とし，出力する．

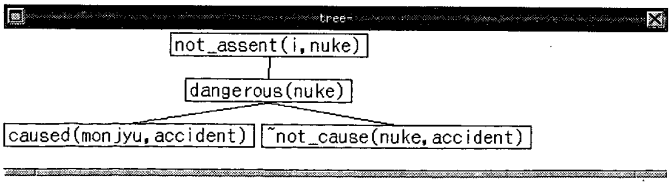

図 6 木構造出力例
Fig. 6 The example of tree output.

(2) の状態出力の種類には,

- 議論/反論/補強議論の作成成功,
- パス（議論の作成に失敗し，次の順番のエージェントに発言権を譲る場合に出力),
- 反論/補強議論の作成失敗（自軍の全員が議論の作成に失敗した場合に出力),
- 協調の申し出が来た場合，受理したか受理しなかったか（6.2 節参照),

がある．この状態出力によって，仲介者エージェントがメッセージの送信先を判断する（8.2 節参照).

(3) の変換出力機能は，作成された議論をユーザに分かりやすく提示するために導入している．以下に英文への変換例を示す．木構造変換の結果については**図 6** を参照.

推論エンジンの議論出力

```
caused(monjyu,accident) <=    .
dangerous(nuke) <= caused(monjyu,accident),
                   ~ ¬ cause(nuke,accident).
¬ assent(i,nuke) <= dangerous(nuke).
```

↓

英文変換の結果

```
It is true that monjyu caused accident.
If monjyu caused accident and there is
  no evidence that nuke doesn't cause
    accident then nuke is dangerous.
If nuke is dangerous then I don't assent nuke.
```

8.2 仲介者エージェント

仲介者エージェントは，各エージェントの推論エンジンが出力した状態の内容からメッセージを送るべき相手を以下のように判断し，コンテンツを作成して送信する．また，結果の管理をし弁証法的合意形成を行う（6.3 節参照)．これらの機能は Java で記述されている.

- 議論が作成されている場合，作成者の相手側の先頭エージェント（P_1 または O_1）に議論を送り，反論の作成を促す.
- パスの場合，次の順番のエージェントに議論作成要求を送る.

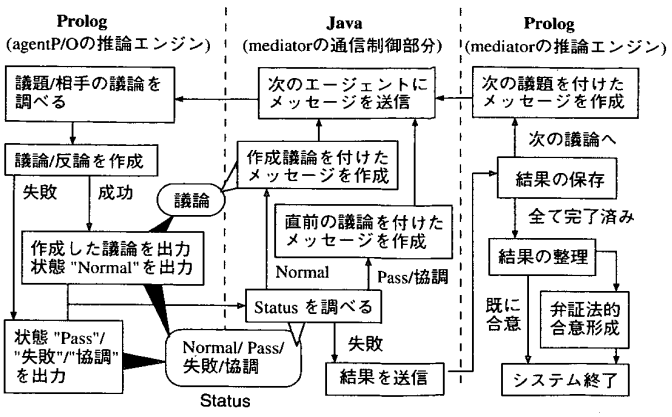

図 7 システム構成図
Fig. 7 The structure of the argument-based agent system.

- 議論作成失敗の場合，協調の余地が残されている場合は補強議論の作成要求を送る.
- 補強議論は送ったが相手に拒否された場合，次の順番のエージェントに補強議論作成要求を送る.
- 協調の余地がない場合・協調が無事終了した場合・協調に失敗した場合は結果を記録し新しい議題を用意する．すべての議題について議論過程が終わっている場合は，最終合意案を生成して終了する.

ここまでの節で述べた各推論エンジンの関係をまとめると**図 7** のようになる.

9. 評　　価

ここまでで述べてきたシステムを評価するにあたり，我々はまずシステムが現実の問題に適用できるかどうか，いくつかの問題を与えて実験した．次に，現実の分散ネットワーク環境に適用できるかどうかの実験を行った．以下，それぞれの実験と，システムの有効範囲に関する考察について述べる.

9.1 適用例 1：原発建設問題

これは「地方都市に原発を建設することに賛成か反対か」に関する知識ベースを新聞などの資料に現れた記事を利用して作成したうえで，発案者側に原発賛成に関する知識を，対立者側に原発反対に関する知識を与えて議論を行わせたものである．知識ベースの内容は省略するが，以下に「原発建設に反対」を議題として与えシステムを起動したところ，議論の過程がどのように進行したかを示す.

P_1：「政府は原発にトラブルが起こっても隠蔽する．そのため原発が信頼できず，ゆえに建設には賛成できない」(本論)

O_1：「（将来にわたって）エネルギーが供給できるとは限らない．ゆえに原発は必要であり，建設に賛成」

P_1：「(原発の排水によって) 魚が汚染されないとは限らず，するとそれを食べる子供が汚染される．ゆえに建設には賛成できない」
O_1：(前回の反論の繰返し)
P_1：「事実としてもんじゅ事故があり，事故が起こらないとは限らないのであれば原発は危険であるといえる．ゆえに建設には賛成できない」
O_1：(前回の反論の繰返し)
P_1：「放射性廃棄物の処理が明確になっていない．さらに，今後明確になるとは限らないのであれば，原発は危険であり，建設には賛成できない」
O_1：(前回の反論の繰返し)
P_1：反論の作成に失敗 (同じ議論を 2 度繰り返せないため)，P_2 に反論作成を託す．
P_2：「エネルギーは節約ができる．ゆえに (当面は) エネルギーを供給していくことができる」
O_1：反論の作成に失敗，O_2 に反論作成を託す．
O_2：反論の作成に失敗．

- 対立者側の全員が反論に失敗したので本論は正当化された．次に，対立者側は協調を試みた．

O_1：補強議論の作成に失敗．
O_2：補強議論の作成に失敗．

- 提案者側の最後の議論には弱否定部分がないため補強議論が作れず，協調は行われなかった．
- 議題はすべて正当化されたので，最終結論は「原発反対に両者合意」となり，システムは停止した．

9.2 適用例 2：プログラミング言語選択問題

この適用例では，いくつかのプログラミング言語についてその性質を記述した知識ベースを製作し，そのうえで「(議論システムの製作に) Java と Lisp が適用できるか」について議論を行わせたものである．ここで，議題は “Java が適用可&Lisp が適用可” とした．また，“¬Java を適用 = C++ を適用，¬Lisp を適用 = Prolog を適用” を仮定した．

知識ベースは割愛するが，実行させたところ「Java が適用可」については正当化され「Lisp が適用可」については却下された．また，協調は行われなかった．ここで弁証法的合意形成が行われ (6.3 節参照)，対立者側が「Lisp が適用可能ではない」を正当化した．よって結果は “Java を適用&¬Lisp を適用” となり，先の仮定と合わせると “Java を適用&Prolog を適用” となった (**図 8** 参照)．

9.3 その他の適用例

我々がシステムを拡張したうえで適用したその他の問題として，次のようなものがある．

スケジュール調整問題 ミーティングを行おうとする 2 者がスケジュールを調整するための交渉を議論で行う．

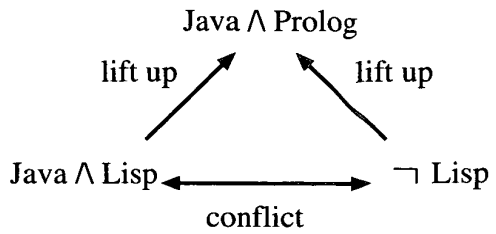

図 8 適用例 2 の結果
Fig. 8 The result of application 2.

Electronic Commerce ネットワーク上の商店をエージェントがめぐり，客と店主，または店と競合店での交渉を議論で行う．

これらの詳細については，文献 7)，10) を参照されたい．

9.4 通信実験

本論文で提唱したシステムを分散ネットワーク環境に適用するためには，実際にエージェントどうしが分散している状態でも動作することを実証しなければならない．

そこで我々は，DS'99 (“Discovery Science 1999”) と IAT'99 (“Intelligent Agent Technology 1999”) の 2 つの会議において，会議会場に提案者側エージェント，新潟大学に対立者側エージェントが動作している状態で議論システムを動作させることで，通信を必要とする環境下での実験を行った．その結果，両会場ともシステムが通信して動作することが確認でき，分散ネットワーク環境に適用できることが示された．

ただし現状ではネットワーク負荷や経路の問題により，通信区間によっては応答が返ってくるまでの時間にばらつきが出ることがあった．このため，タイムアウト時の処理について考える必要が出てくるが，ここでは今後の研究課題とする．

9.5 システムの有効範囲

本論文で提案した手法は，すべての問題に対して有効であるわけではない．ここでは，システムがどのような問題に適しており，どのような問題に適していないかを考察する．

まず，本論文のシステムが適していると思われる問題には，主に次のようなものが考えられる．

- 対立的状況で構成される問題．
- エージェントのゴールが始めから決まっており，二者択一命題 (またはその連言肢) で表せる問題．
- 背景知識に，弱否定で対応できる範囲でのあいまいな表現が含まれている問題 (議論システムで単に扱えるというだけでなく，補強議論による協調

の効果も期待できる).

また，本論文の手法そのままでは対応できないが，拡張次第で対応できる可能性が出てくる問題には，以下のようなものが考えられる．

- 相手が提出した議論からの効果的な学習（自分の知識として吸収する際に，どの知識が自分にとって有用で，どの知識が有用でないかの指標が必要であると思われる．ただし，自分の知識として吸収せず，単に「相手の発言」として記憶しておき，後の議論に活用するという手法もある[10]).
- 議論過程からの学習（本論文では複数回議論した場合の効用について触れていないが，たとえば相手の戦略や発言の傾向を記録しておき，後の議論に活用していくことなどが考えられる).
- 互いに提案を出し合い，その1つ1つについて検討するようなモデル（議題という形で提案を出し合い，それについて議論するという形で実現可能である[10]).
- 参加するエージェントが動的であるようなモデル（議論プロトコルを，参加者・発言順固定となっている現状から，もっと柔軟なものに拡張する必要がある．たとえば，契約ネットプロトコルを用いるという手法が考えられる[2),5)]).
- 議論の提出戦略を柔軟に変化させていくことが重要となるようなモデル（戦略を決定するためのパラメータなどが必要だと思われる[2),5)]).

最後に，本論文の手法で対応させるのが難しいと思われる問題には，次のようなものが考えられる．

- 価格交渉などの，定量的な問題（システムの現状を考えると，数値的なアプローチの方が有効であると思われる).
- エージェントの参加数が極端に多いモデル（n対nエージェントで議論できるプロトコルとはなっているが，発言権がつねに順番に与えられるため，効率が悪くなったり収束に時間がかかることが考えられる).

10. おわりに

本論文では，エージェントシステムに議論を導入し拡張することによって，衝突回避・協調動作・合意形成の3つの目的へ応用できることを示した．また，実際にシステムを試作し現実の問題に適用することによって，分散ネットワーク環境における合意形成・意思決定のための新しい方法となりうることを示した．

今後の展望としては，9.5節で触れたような議論の経過や結果からの有益な情報の取り込みによる学習の実現，妥協・譲歩への応用[10]，グループウェアなどの分野への応用が考えられる．

以上のことから，議論するエージェントシステムが分散ネットワーク社会の新しい通信手段，新しいエージェント交渉の方法となる可能性があると考える．

しかし現状ではシステムがまだ基礎的なものにすぎず，多くの拡張の可能性が残されている．

まず，各エージェントに最初から役割を振ってしまっていることがあげられる．議論開始時にいずれかのグループに属していなければならないという形よりも，参加の意思のあるエージェントのみが議論を行い，反論できるものには反論し，協調できるものには協調するといった自由度が高い形式の方が望ましい．

また，各エージェントが議論に参加する際の運用的な面について，本論文の範囲では触れられていない．

これらの問題点を解決する1つの方法として，議論するエージェントに契約ネットプロトコルを用いる方法[2),5)]がある．このシステムでは，マネージャとなるエージェントがタスクを告知し，それに入札してきたエージェントの中から一定の評価基準以上のエージェントを落札する方法をとっている．この方法であれば，最初から役割を決めることなく参加可能なエージェントだけを集めて議論することができる．

次に，戦略がグループによって固定されており，基本的な戦略は議論中で不変である．これも個々のエージェントで戦略に差が出たり，相手や自分の状況から戦略を変えたりするという形式の方が，エージェントがより自律的に振る舞うことができる．以上のことは今後の課題である．

最後に，本システムで作成された議論（結果）は論理的には正しいといえるが，出た結果を社会的にどう扱うかについては社会学上の見解を必要とする．この点に関する考察も深めていきたい．

謝辞 弁証法に関して数々のご教示をしていただいたオーストラリア国立大学のR.K. Meyer教授に感謝の意を表します．

参考文献

1) Loui, R.P.: Defeat among Arguments: A System of Defeasible Inference, *Computational Intelligence*, Vol.2, pp.100–106 (1987).
2) Maeda, S., Guan, C. and Sawamura, H.: An Argument-based Agent System with the Contract Net Protocol, *Proc. 1st Asia-Pacific Conference on Intelligent Agent Technology Systems, Methodologies and Tools*, pp.99–103, World Scientific (1999).

3) Pollock, J.L.: Defeasible reasoning, *Cognitive Science*, Vol.11, pp.481–518 (1987).
4) Prakken, H. and Sartor, G.: Argument-based extended logic programming with defeasible priorities, *Journal of Applied Non-Classical Logics*, Vol.7, pp.25–75 (1997).
5) Sawamura, H. and Maeda, S.: An Argumentation-Based Model of Multi-Agent Systems, *Proc. 10th European-Japanese Conference on Information Modelling and Knowledge Bases*, pp.96–109 (2000).
6) Sawamura, H., Umeda, Y. and Meyer, R.K.: Computational Dialectics for Argument-based Agent Systems, *Proc. 4th International Conference on MultiAgents Systems*, pp.271–278, IEEE (2000).
7) Sawamura, H., Yamashita, M., Inagaki, M. and Umeda, Y.: Agents Meet Dialectics, *Proc. International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA2000)*, pp.354–360, ICSC Academic Press (2000).
8) Umeda, Y. and Sawamura, H.: Towards an Argument-Based Agent System, *Proc. 3rd Int. Conf. on Knowledge-Based Intelligent Information Engineering Systems*, pp.30–33, IEEE (1999).
9) Umeda, Y., Yamashita, M., Inagaki, M. and Sawamura, H.: Argumentation as a Social Computing Paradigm, *Proc. 3rd Pacific Rim International Workshop on Multi-Agents, PRIMA 2000*, pp.46–60, Springer-Verlag (2000).
10) Yamashita, M., Umeda, Y. and Sawamura, H.: Applications of the Argument-Based Agent System with Dialectical Reasoning Capability, *Proc. 4th International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, pp.301–304, IEEE (2000).
11) 沢村 一：議論の論理と議論するマルチエージェント—論理から議論へ，人工知能学会誌，Vol.16, pp.482–487 (2001).

(平成 13 年 3 月 2 日受付)
(平成 14 年 2 月 13 日採録)

梅田 勇一

1999 年新潟大学工学部情報工学科卒業．2001 年新潟大学大学院自然科学研究科博士前期課程修了．現在，同研究科博士後期課程在学中．エージェントコミュニケーションに興味を持つ．人工知能学会学生会員．

沢村 一（正会員）

1978 年北海道大学大学院工学研究科情報工学専攻単位修得退学．1980 年～1996 年（株）富士通，(株）富士通研究所情報社会科学研究所で，主任研究員，室長，主管研究員等を歴任．1989 年～1990 年および 2000 年オーストラリア国立大学 Visiting Fellow．1996 年より新潟大学工学部情報工学科助教授，博士（工学）．計算論理学，ソフトウェア基礎論，人工知能分野に興味を持つ．人工知能学会，ソフトウェア科学会，日本科学哲学会各会員．