ソフトウエア科学会研究会資料シリーズ　ISSN 1341-870X No.19

MACC2001
第 10 回マルチエージェントと協調計算ワークショップ
The Tenth Workshop on Multiagent and Cooperative Computation

MACC2001

第 10 回マルチエージェントと協調計算ワークショップ論文集

武田英明編

ソフトウエア科学会研究会資料シリーズ

MACC2001

第 10 回マルチエージェントと協調計算ワークショップ論文集

2002 年 2 月 1 日発行

編集　武田英明

〒101-8430　東京都千代田区一ツ橋 2-1-2

国立情報学研究所

http://www-kasm.nii.ac.jp/~takeda/

電話 03-4212-2543

FAX　03-3556-1916

発行　日本ソフトウエア科学会

〒113-0032 東京都文京区弥生 2-4-16 学会センタービル内

http://www.jssst.or.jp/jssst/

電話　03-5802-2060

FAX　03-5802-3007

ISSN 1341-870X No.19

目次[1]

はじめに
武田 英明 v
アドバイザリーコミッティとプログラム委員 vi

パネルセッション

1. **社会的エージェントの可能性**
石田 亨, 中西 英之, 長尾 確, 小野 哲雄 1

エージェントモデル

2. **義務と権利に基づく社会行動の即応制御モデル**
大澤 一郎 3

3. **BDI Logic の sequent calculus による演繹体系**
新出 尚之，高田 司郎 10

4. **マルチエージェントによるチーム航行支援 (S)**
安藤 英幸，広野 康平，大和 裕幸 21

5. **Wizard of Oz 法を使ったマルチキャラクターエージェントインタフェースの評価 (S)**
辻本 秀樹，北村 泰彦，辰巳 昭治 26

エージェントプロトコル

6. **エージェント基盤のための協調プロトコル：Agent Platform Protocol**
高橋 健一，鍾 国強，雨宮 聡史，峯 恒憲，雨宮 真人 30

7. **マルチエージェントシステムへの投機的計算導入手法**
山本 京司，佐藤 健 38

8. **モバイルエージェント系における分散制約充足**
沼澤 政信，栗原 正仁，能登 正人 46

9. **ユーザ選考に基づく分散コンテンツキャッシュポリシ (S)**
中塚 康介, 八槇 博史 54

コミュニティのためのエージェント

10. **口コミによる分散型情報収集システム**
吉田 匡志，伊藤 雄介，沼尾 正行 61

11. **知識の階層構造を用いた協調的情報推薦 (S)**
濱崎 雅弘，武田 英明，河野 恭之，木戸出 正継 69

[1] タイトルに (S) とあるものはショートペーパー、その他はロングペーパー

ロボット

12. **RoboCupRescue における最少通話による協調の実現**
森本 武資，河野 健二，竹内 郁雄 71

13. **異種端末間の協調による高齢者・障害者の移動支援、Robotic Communication Terminals(S)**
矢入（江口）郁子、猪木 誠二 78

学習と知識獲得

14. **未来状態の予測を利用したマルチエージェント強化学習法**
長行 康男，柴田 直樹，伊藤 実 85

15. **環境状況の変化に応じて自己の報酬を操作する学習エージェントの構築**
森山 甲一, 沼尾 正行 91

16. **自律学習における CITTA による部分知識の結合 (S)**
山川 宏，宮本 裕二，馬場 孝之，岡田 浩之 99

17. **ネット情報を使った意思決定のための知識獲得エージェント (S)**
藤本 和則，山本 裕 103

特別セッション [ゲーム理論／意思決定／経済学的アプローチに基づくマルチエージェント]

18. **情報財の開発インセンティブについて**
藤山 英樹 107

19. **人工市場における複雑さの発展の分析**
和泉 潔 116

20. **X-Economy - マルチエージェント経済におけるシミュレーションプラットホーム**
川村 秀憲，車谷 浩一，大内 東 122

21. **P2P 電子マーケットにおけるエージェント間の提携に基づく売買交渉機構について**
服部 宏充，伊藤 孝行，新谷 虎松 128

22. **U-MART：経済学と工学をエージェントが結ぶ**
喜多 一、出口 弘、寺野 隆雄 138

23. **エージェントシミュレーションによる「チープ・トーク」モデルの拡張**
小山 友介 144

24. **エージェント系としてのレプリケータダイナミックスとその制御**
國上 真章、寺野 隆雄 152

25. **エージェントシミュレーションによる Prospect 理論と GARCH モデルの関連性の分析**
高橋 大志、寺野 隆雄 158

26. **複数財オークションについて**
松井 知己，渡辺 隆裕 166

27. **セキュアマルチエージェント動的計画法の設計と組合せオークションへの適用**
鈴木 幸太郎．横尾 真 171

はじめに

プログラム委員長 武田 英明

本論文集は日本ソフトウエア科学会「マルチエージェントと協調計算研究会」の第 10 回ワークショップ（**第 10 回マルチ・エージェントと協調計算ワークショップ**，**MACC2001**）で発表されたものを集めたものである． MACC2001 においては投稿された論文はプログラム委員による査読を受けて，著者による加筆・修正ののち発表，本論文集に掲載というプロセスを 経ている．本年度は，通常セッションにロングペーパー9 件，ショートペーパー7 件，特別セッション 10 件の発表が行われた．

MACC2001 は金沢のラポート兼六を中心に 2001 年 11 月 16 日，17 日に開催された．参加人数は約 60 人で，基本的に全員泊りがけで行った．

初日は前日より金沢大学で開催していた電子情報通信学会人工知能と知識処理研究会と共同企画でパネルを行った．その日は懇親会を兼ねた夕食の後，発表者以外の参加者の 30 人弱のポジションペーパー発表が行われた．ポジションペーパー発表とはいえ，興味深い発表もあり，活発な質疑が行われた発表もあった．

翌日からは2トラックのパラレルセッションとして行った．今回は，特別セッションに多数の論文が集まったため，基本的に1つのトラックを特別セッションとして行った．

通常セッションは，ロングペーパー，ショートペーパーを織り交ぜ，「コミュニティのためのエージェント」，「ロボット」，「エージェントモデル」，「エージェントプロトコル」，「学習と知識獲得」というセッションに分けて行った．

特別セッションは「ゲーム理論／意思決定／経済学的アプローチに基づくマルチエージェント」というテーマで 10 件の発表が行われた．時間帯によっては通常セッションよりむしろ特別セッションの方に人があつまり，立ち見がでる状況であった．従来の MACC の発表者が人工知能を中心とする情報工学の分野であったの対して，この特別セッションの発表者はセキュリティの研究者や経済学の研究者など幅広いものであり，この分野が広範な範囲の人々の興味をひきつけており，今後の発展の可能性を示していた．

このようにマルチエージェントの分野は単に人工知能あるいは情報工学の特定の興味でなく，広範な興味・話題を含みようになっている．このような認識のもとで MACC は来年度は電子情報通信学会人工知能と知識処理研究会と共同で，エージェント／マルチエージェントの話題を広く含むような会議として開催されることになった．同じく 11 月中旬に函館での開催が予定されている．

本ワークショップ開催にあたってはプログラム委員のみなさまの協力のもとで成功裏に開催することができた．ここに感謝の意を表する．

Preface

Hideaki Takeda, 国立情報学研究所, National Institute of Informatics.

アドバイザリコミッティ

石田 亨 (京都大学)
奥乃 博 (京都大学)
所 真理雄 (ソニー)
中島 秀之 (産業技術総合研究所)
西田 豊明 (東京大学)
本位田 真一 (国立情報学研究所)

プログラム委員

[委員長] 武田 英明 (国立情報学研究所)
伊藤孝行 (北陸先端科学技術大学院大学)
和泉 潔 (産業技術総合研究所)
磯崎 秀樹 (NTT)
大澤 一郎 (産業技術総合研究所)
大沢 英一 (はこだて未来大学)
大澤 幸生 (筑波大学)
小野 典彦 (徳島大学)
北村 泰彦 (大阪市立大学)
佐藤 理史 (京都大学)
椹木 哲夫 (京都大学)
島津 秀雄 (NEC)
菅沼 拓夫 (東北大学)
菅原 俊治 (NTT)
角 康之 (ATR)
高間 康史 (東京工業大学)
垂水 浩幸 (香川大学)
津田 和彦 (筑波大学)
中内 靖 (防衛大学校)
長尾 確 (名古屋大学)
西村 俊和 (立命館大)
橋田 浩一 (産業技術総合研究所)
平山勝敏 (神戸商船大学)
藤本 和則 (NTT)
松原 繁夫 (NTT)
本村 陽一 (産業技術総合研究所)
八槇 博史 (京都大学)
横尾 真 (NTT)
吉田 哲也 (大阪大学)

パネル: 社会的エージェントの可能性

石田 亨† 中西 英之† 長尾 確‡ 小野 哲雄††

エージェント研究の流れは, コンピュータインタフェースの一形態として人間のコンピュータ利用を支援する「個人的エージェント」から, 社会的役割を持ち, 人と人のインタラクションに介在する「社会的エージェント」へと向かっているように思われる. 本パネルでは, ソフトウェアエージェントからロボットに至るまでの, 我が国で最新の研究を紹介し, 「社会的エージェント」という新しい研究分野に定義を与え, その可能性を議論する.

1 パネルプログラム

- 司会 石田 亨
- 社会心理学実験報告: エージェントは人間関係を操作できるか 中西英之
- 社会的マシンとしての会話ロボット 長尾確
- Robovie:コミュニケーションにおける身体性と関係性 小野哲雄

2 各講演概要

2.1 社会心理学実験報告: エージェントは人間関係を操作できるか

中西英之 (京都大学社会情報学専攻)

ソフトウェアエージェントが人間関係に及ぼす影響を明らかにする第一歩として, エージェントが人間関係を操作できるかどうか確かめる実験を行なった. この実験はバランス理論にもとづいて行われた. バランス理論によると, 2 人の人間がある対象に同じ心情(好意もしくは敵意)を持つ場合の方が, 片方は好意を持ち, もう片方は敵意を持つ場合よりも, 2 人の間の関係が良くなる. エージェントと人間 2 人の関係にこれを当てはめた.

実験ではまず, エージェントが同意の態度を示すことで人間に好意を持たれ, 逆に反意の態度を示すことで敵意を持たれることを確かめた. 次に, バランス理論が人間 2 人とエージェントの関係に成り立つかどうかを観察した. バランス理論が成立するとすれば, エージェントが両方の人間に同意や反意の態度を取る場合の方が, 片方にだけ同意して, もう片方に反意する場合より, 人間同士の関係は良くなる.

実験の結果, 人間同士の会話が無いときは, 人間関係をエージェントが操作できた. しかし, 少しでも会話があると, エージェントは人間関係を操作できなかった. 少しの会話がエージェントの能力を失わせた原因を探るため会話分析を行ったところ, 会話を通して人間 2 人が互いに共感することが原因であると判明した.

2.2 社会的マシンとしての会話ロボット

長尾確 (日本IBM東京基礎研究所)

IBM 東京基礎研究所では、新たなヒューマンインタフェースとして会話ロボットに関する研究を進めている。 IBM の取り組むロボットは、手足の動きなど外観の機能や動きを追求するのではない。「人間にとって役に立つ、便利な助手のような存在」として利用できるようにするための仕組み、つまり知能の部分にフォーカスを当てている。人間とのコミュニケーションの手段は非常に身近な「会話」である。しかしながら、人間が機械に対して話しかける行為に抵抗を感じるという問

Panel: Potential of Social Agents

† Toru ISHIDA, Hideyuki NAKANISHI 京都大学社会情報学専攻 Department of Social Informatics, Kyoto University
‡ Katashi NAGAO 日本アイ・ビー・エム株式会社 東京基礎研究所 IBM Research, Tokyo Research Laboratory
†† Tetsuo ONO 公立はこだて未来大学, Future University Hakodate

題をまずはクリアしなければ役には立たない。人間は無意識のレベルで、存在感があり、動作する対象が自分に注意を向けていることを感じ取ると、それは「自分とコミュニケートする対象なのだ」と直感的に認識していると思われる。IBM は直感的で臨場感のある「会話」によって、人間と機械がスムーズなコミュニケーションを行なうために、「ロボット」という入れ物を利用している。ロボットは、話しかけられると、ユーザーの意図を理解し、必要な情報を集めて自動的に処理し、適切な応答の生成を試みる。さらにはユーザーと行ってきた会話の履歴を記憶する能力を持たせることにより、ユーザーの嗜好にあった行動をロボットが自発的に行なってくれるようになるだろう。また、ユーザーに特化し同調するような仕組みとして感情を理解できるシステムもあらかじめ組み込んでおけば、ユーザーがうれしいとき、ロボットもうれしいし、楽しいとロボットも喜んでくれるだろう。ロボットが知性と感情を持ちあわせるようになったとき、便利さだけでなく、人間を癒してくれる存在となるだろう。人間との会話や知識を処理するインテリジェンスを持たせたとき、ロボットは人間にとって本当に信頼のおけるパートナーになり得るだろう。また、ロボットは、コミュニティを媒介し、人同士のコミュニケーションを促進するシステムとしても機能する。たとえば、会話ロボットは遠隔地にいるもの同士が非同期でコミュニケーションする場合に、物理的な存在感を備えた、擬似的な対面性を実現することができる。また、長い経験を通じて個人情報を獲得したロボットは、その人の外在化された記憶あるいはパーソナリティを持つシステムとして機能できるようになるだろう。これは、ロボット（あるいはエージェント）が文字通り人間の代理人になるということであるが、もしロボットの信用や責任という問題がうまく技術的あるいは社会的に解決できれば、人間は自分のコピーあるいは分身を作って同時に複数の作業ができるようになるだろう。そのようなロボットを社会的マシンと呼び、その可能性と課題について議論する。

2.3 Robovie：コミュニケーションにおける身体性と関係性

小野哲雄（公立はこだて未来大学／ATR メディア情報科学研究所）

近年，エージェントやロボットの社会性に関する議論が行われている．本発表では，コミュニケーションにおける「関係性」と「身体性」に注目し，社会的ロボットの実現可能性とその研究方法について述べる．

エージェントやロボットなどの人工物が社会的存在となるためには，人と自然で円滑なコミュニケーションを実現しなければならない．しかし，いまだに人は，エージェントやロボットとのインタラクションに違和感を持ってしまう．この理由は，これまでのコミュニケーション研究が言語的側面だけを重視してきたことによる．本発表では，自然なコミュニケーションを実現するには，その成立の基盤をなす「関係性」と「身体性」に注目する必要があり，さらにそれらの機能をシステムに取り込む必要があることを主張する．

本発表では，2 つの研究事例を紹介する．まず，Ono [1]は，人とロボットの間に関係性を構築するためのインタフェースとして，エージェントマイグレーションを用いた手法を提案した．具体的には，ユーザとインタラクトしていた携帯端末上のエージェントがロボットへ移動することにより，両者の間に関係を構築することを目指したものである．実験の結果，エージェントが移動したときユーザはロボットの発話を理解できるが，移動しないときは理解できないことが明らかとなった．

次に，Ono [2]は，コミュニケーションにおける空間表象の伝達と身体動作の関係に注目し，共創対話のモデルを提案した．本モデルは，身体の同調的動作による対話者間の関係の構築，および，この関係に基づく情報の伝達のメカニズムを理解するためのモデルである．ヒューマノイド型ロボットを用いた実験の結果，人は身体の同調的な動作により円滑な情報伝達を行っていることが明らかとなった．

これらの研究成果から，コミュニケーションにおける関係性と身体性の重要性が確認できた．さらに，これらの結果は，社会的ロボットの実現方法に示唆を与えるものである．

[1] T. Ono, M. Imai (2000). Reading a Robot's Mind: A Model of Utterance Understanding based on the Theory of Mind Mechanism, AAAI-2000, pp. 142--148.

[2] T. Ono, M. Imai, H. Ishiguro (2001). A Model of Embodied Communications with Gestures between Humans and Robots, CogSci-2001, pp. 732--737.

義務と権利に基づく社会行動の即応制御モデル

大澤 一郎

A new computational model of socially interactive systems driven by obligations and rights is proposed. Obligations and rights are explicitly represented with the operators “O” and “R”, and the internal computation is interactively executed on a reflective architecture based on recall and observation inside. We implement a prototype system to show its performance on some social domains such as business transactions, secure information handling, and driving on the road.

1 はじめに

コンピュータ・システムが社会のあらゆるところに入り込み、政治、経済、金融から、一般の人々の生活までもが、世界中に張り巡らされたコンピュータ・ネットワークによって密接に結び付けられつつある。このような状況下では、ネットワーク内で活躍するエージェント・システムも社会的あるいは組織的に守られた特殊な存在ではなく、社会人と呼ばれる成人のように社会で一人前に行動できなければならない。しかしながら、これまでの大半のエージェント・システムは、BDI モデルで代表されるような、欲求ベースで行動する幼児のような存在でしかなかった。

近年、社会的行為における義務と権利の重要性が明らかになるにつれ、この分野で多くの研究が行われつつある。様相論理の一種である義務論理の多くは、義務と許可を双対なものとみなして体系化しようとして[4]、義務論理のパラドックスを生じさせていた。Castelfranchi [2]は権利を社会的コミットメントに基づいて派生するものと考えたが、Norman [5]は権利を明示的に表現し、権利の概念を柔軟に扱おうとした。Alonso [1]は、義務と権利の関係に着目し、権利の背後にある周囲の義務を含めて権利を定義しようとしている。このような権利の明示的表現という論理的な研究の流れに対応して、Sloman [3]らのグループは、データーベースにおけるセキュリティ管理を一般化して、役割 (ロール) に基づく義務と権利をオブジェクト指向モデルに導入している。また、Wagner [8]はエージェント指向モデルに義務と権利の枠組みを導入しようとしている。

本研究では、このような研究動向を踏まえ、義務と権利に基づいて社会的に自律動作するインタラクティブシステムの内部計算モデルを提案し、実際に実装したプロトタイプによる社会的行動の内部シミュレーションを示す。以下では、2 節でシステム内部における情報の表現形式を説明し、3 節で基本となる計算モデルを集合的概念に基づいて提案する。そして、4 節でマルチエージェントによる実装と、5 節でプロトタイプによるシミュレーションを紹介する。

2 義務と権利の表現

2.1 BDI モデルの言語拡張

義務と権利に関する情報を明示的に扱うために、人工知能における BDI モデル[7]を言語拡張する。義務を O(obligation) で表現し、権利を R(right) で表現する。

A reflective computational model of socially interactive systems

Ichiro Osawa, AIST 情報処理研究部門

基本的な BDI モデルでは、システムの内部状態を信念 B(belief) と欲求 D(desire) と意図 I(intention) によって表現する。信念 B がシステムの信じている情報で、欲求 D がシステムの望んでいる状態あるいは事象、意図 I がシステムの決意した状態あるいは事象である。例えば、システム s が x という命題を信じていて、y という状態を望み、z という事象を意図している場合を以下のように表現する。

Example 1

$$Bs.x \wedge Ds.y \wedge Is.z$$

欲求 D は実現すれば望ましい状態であり、将来において成り立つ可能性があると信じていれば、他の欲求と相反していても良い。一方、意図 I は実現を確信して身を委ねるような状態あるいは事象であり、将来において必ず成立すると信じていなければならない。

義務 O は、あるシステム s がシステム t に対して状態あるいは事象の実現の責務を負っていることを表現する。この義務の表現を用いれば、商店主が客から代金を受け取った場合に商品を渡さなければならないという規則を次のように記述することができる。但し、[行為] は $done$(行為者, 行為) の略記である。

Example 2

$Os, t.[give, t, x]$

$\leftarrow \quad Ds.[sell, t, x] \wedge Bs.price(x, p) \wedge Bs.[take, t, p]$

なお、基本的な BDI モデルでは、意図 I は行為の実行を意図する場合と、状態の達成を意図する場合の両方に用いられるが、本モデルでは意図 I を行為の実行意図のみに用い、状態達成の意図は自己に対する義務 $Oi, i.x$ で表現している。

権利 R は、システム s に事象の実行権があることを表現する。例えば、商店主は客に商品を売る義務があれば、その客から代金を受け取る権利があるという規則を以下のように記述できる。

Example 3

$Rs.[take, t, p]$

$\leftarrow \quad Os, t.[sell, t, x] \wedge Bs.price(x, p)$

$\wedge Bs.\neg[take, t, p]$

2.2 メッセージの解釈

オブジェクト指向システムでは、システムがメッセージを受信すると、受信メッセージに対応するメソッドが自動的に起動され、メソッドのプログラムに従ってメッセージの処理が実行される。メッセージの基本的な送受信機構に義務や権利の概念は含まれていないので、送信者によって異なるメッセージ処理を行うようにするには、メソッドのプログラム内に送信者情報に基づいた分岐処理を内在させておかなければならない。

それに対してこのシステムでは、明示的に定義してある規則に基づいて、受信したメッセージを義務 (あるいは情報) として解釈し、義務の履行という形式でメッセージの処理を実行する。例えば、システム tom からの質問メッセージに対して必ず回答しなければならないシステムの場合、以下の規則を定義しておく ($Ms.$内容@t は t から s へのメッセージを表す)。

Example 4

$Oi, s.[ans, s, p] \leftarrow Mi.[ans, s, p]@s \wedge s = tom$

3 自己反映型即応計算モデル

3.1 計算モデル

信念 B、欲求 D、意図 I、および義務 O と権利 R に基づいてシステムをインタラクティブに動作させるために、その計算機構として自己反映型[6]の即応システム RRS を用いる。この節では、この計算機構のベースになっている計算モデルを集合論的な概念に基づいて定義する。

自己反映型即応システム RRS は、時々刻々と変化する状態 RS_t と想起規則 RR とから構成され、時刻 t における状態 RS_t は、観測 SO_t と記憶 SM_t と想起 SR_t の 3 状態から構成される。

Definition 1 (自己反映型即応システム RRS)

$$RRS = (RS_t, RR) \qquad [t = 0, 1, 2, , ,]$$
$$RS_t = (SO_t, SM_t, SR_t)$$

時刻 t におけるシステムの観測としては、外部世界からのメッセージ M あるいはイベント情報 E、若しくはシステム自身がその時点で想起している信念 B と欲求 D と意図 I と義務 O と権利 R に関する想起情報 SR_{t-1} の中から、まだ未観測のものを一つだけ取捨選択することができる。すなわち、システムの観測対象で

ある世界 W_t は、メッセージ M とイベント情報 E から成る外部世界 OW_t と、想起 SR_{t-1} に基づく内部世界から構成されており、時刻 t における観測 SO_t は次のように表現できる。但し、: は $\subset$ とほぼ同義で、右側の集合に含まれる要素の種類 (タイプ) を拡張 BDI モデルのオントロジーに基づいて補足的に表現している。例えば S : $M + E$ は、集合 S の要素が M あるいは E の情報であることを表している (尚、省略形として $M + E$ の代わりに ME、$B + D + I$ の代わりに BDI、$O + R$ の代わりに OR を用いている)。

Definition 2 (観測 SO_t)

$$W_t = OW_t \cup SR_{t-1} : ME + BDI + OR$$
$$SO_t \subset W_t - SM_{t-1} : ME + BDI + OR$$
$$|SO_t| \leq 1$$

記憶 SM_t は、直前の記憶 SM_{t-1} をベースに計算する。SM_t は、SM_{t-1} の情報をほぼ全て継承するが、メッセージ M の場合は、そのメッセージから想起規則 RR で想起され未観測のものがあるときだけ継承される。解釈の終了したメッセージを記憶から抹消するためである。そして観測 SO_t を加えることで時刻 t における記憶 SM_t が計算できる。

Definition 3 (記憶 SM_t)

$$SM_t = SO_t \cup (SM_{t-1} - M)$$
$$\cup \{m | m \in SM_{t-1} \cap M$$
$$\wedge |RR(m, SM_{t-1}) - SM_{t-1}| > 0\}$$
$$SM_t : ME + BDI + OR$$

想起規則 RR は、記憶 SM_t 内の情報に基づいて想起する情報を求める規則である。規則は大きく分類して 4 つのタイプがある。(1) メッセージ M から信念 B あるいは義務 O を想起する規則 RR_1、(2) 欲求 D から別の欲求 D、あるいは意図 I、義務 O、権利 R を想起する RR_2、(3) 意図 I から実行プロセスを想起する RR_3、そして (4) 義務 O から欲求 D あるいは権利 R を想起する RR_4 である。RR_1 でメッセージを解釈し、RR_2 はいわゆるトップダウンのプランニング、RR_3 が意図に対応する行為の実行、RR_4 が義務 O から欲求 D や権利 R を想起する。

Definition 4 (想起規則 RR)

$$RR_1 : M \times SM_t \rightarrow B + O$$
$$RR_2 : D \times SM_t \rightarrow D + I + O + R$$
$$RR_3 : I \times SM_t \rightarrow E$$
$$RR_4 : O \times SM_t \rightarrow D + R$$
$$RR = RR_1 + RR_2 + RR_3 + RR_4$$
$$RR : (M + D + I + O) \times SM_t \rightarrow E + BDI + OR$$

以上のことから時刻 t における想起 SR_t は、記憶 SM_t と想起規則 RR から以下のように定義できる。

Definition 5 (想起 SR_t)

$$SR_t = \cup_{x \in SM_t \cap (M \cup D \cup I \cup O)} RR(x, SM_t)$$
$$SR_t : E + BDI + OR$$

ここで強調しておきたいことが一つある。このモデルの本質的な点であるが、システムが想起していても観測していない情報は、システムに認識されておらず、システムの動作に全く影響を与えていないということである。RR_3 による行為の実行だけは、実行それ自体をシステムが観測していなくても外界に何かしらの影響があるので間接的な情報の観測によってシステムは影響されるかも知れない。しかしながら、それ以外の信念 B、欲求 D、意図 I、義務 O、権利 R の想起に関しては、直にその想起を観測しない限りはシステムに影響を与えることがない。

3.2 記憶の自動調整

システムの信念 B は時々刻々変化している。ある時、活動用のエネルギーが不足してきたという信念に基づいてエネルギーの補充欲求が生じたとしても、その後の信念の変化によって、活動用のエネルギーが不足しているという信念が消えてしまうことがある。このような場合には、エネルギーの補充欲求も消滅してほしい。この思考の自動調整機能は以下のように簡単に実現できる。

Definition 6 (欲求・意図・権利の活性状態)

$$x \text{は} active \Leftrightarrow x \in SR_t \cap SM_t \cap (D \cup I \cup R)$$

システムの欲求 D、意図 I、権利 R は、想起規則に基づいて想起され、それが観測されることで活性化し、システムの動作に影響を与える。しかしながら、SM_t 内の信念等の変化によって、活性化していた欲求 D、意図 I、権利 R の想起がなくなってしまうことがある。この場合、観測によって SM_t 内に加えられている欲求 D、意図 I、権利 R に関して、以下の定義にしたがって活性状態でなくなったものを継承しないように

する。

Definition 7 (修正 SM_t)

$$SM_t = SO_t \cup SM_{t-1} \cap (E \cup B \cup O) \cup \{x | x \in SM_{t-1} \cap (D \cup I \cup R) \wedge x \text{は} active\} \cup \{x | x \in SM_{t-1} \cap M \wedge |RR(x, SM + t - 1) - SM_{t-1}| > 0\}$$

4 マルチエージェントアーキテクチャ

4.1 基本エージェント

前節で述べた自己反映型即応システムを効率よく実装するために、表示と観測に基づいて各エージェントが自律動作するマルチエージェントアーキテクチャ [9] を用いる。各エージェントは、自己および他エージェントの表示している情報を常時観測し、エージェント毎に定義されている規則に基づいて表示情報の更新計算を行う。システム内で変化する情報は全体の僅かな割合なので、この計算は非常に素早く実行できる。全てのエージェントの表示情報が固定した時点で、自己反映型即応システムの一時刻における計算が終了する。

Definition 8 (エージェント)

エージェント$A =$ (観測A_O, 規則A_R, 表示A_D)

各エージェントはローカルに情報を保持し、保持している情報が全てそのまま表示される。自己の表示している情報は全て観測できるが、他エージェントの表示情報に関しては予め観測したいものを宣言しておく。すなわち、観測したい情報の種類と情報源を予め観測 A_O として定義しておく。例えば、 $memory$ エージェント内の意図情報を観測したい場合には以下のように定義しておく。

Example 5

$[observing,\ Ix.y@memory]$　　※xとyは変数

この定義により、 $memory$ エージェントの表示 A_D に $Ia.[eat]$ 等の情報が表示されていれば、その情報を観測することができる。

Definition 9 (規則 A_R)

規則A_R： 表示情報×非表示情報 → 表示情報

規則 A_R は、観測した自己および他エージェントの表示情報から自己の表示 A_D を更新計算する。各規則は 2 つの入力項と 1 つの出力項から構成される。入力項の 1 つ目は表示されていなければならない情報で、2 つ目は非表示でなければならない情報である (他エージェントの表示情報は「情報 @ エージェント」という表現形式によって自己の表示情報と区別している)。出力項には、入力項の条件が満たされた時に自己の表示 A_D に付け加えるべき情報を定義しておく。以下に示した例は、食べたいという情報があり、食べる権利のあるものに関する情報がエージェント m から観測できない場合に、食べる権利のあるものを欲求するという情報を自己の表示 A_D に付け加えるという規則である。

Example 6

$$\{Dx.[eat]\} \times \{Rx.z * [eat, z]@m\} \rightarrow Dx.y * [eat, y]$$

4.2 エージェントの拡張

規則 A_R を拡張し、自己の表示情報を更新するだけでなく、他エージェントに直接作用して他エージェントの表示情報を更新できるようにしてある。すなわち、出力項の情報定義を「情報 @ エージェント」という形式で記述しておくことで、入力項の条件が満たされた時に出力項の情報が別のエージェントの表示情報として用いられる。これはエージェントのモジュール性を損なうようにも思えるが、オブジェクト指向のメッセージに代わるものとして利用することができる。

そして、人間の長期記憶あるいはデーターベースのように持続性をもって蓄えられる情報を処理するために、特殊なエージェント m を用意した。この m エージェントに対して、「+情報@m」という形式で直に表示情報を送ると、内部に情報が書き込まれ持続的に表示される。その後に情報を抹消したい場合は「−情報@m」という形式で m エージェントに抹消希望の情報を送る。

4.3 自己反映型即応システムのアーキテクチャ

自己反映型即応システム RRS は、 m エージェントを含む 12 の主要エージェントから構成される (図 1)。$sensor$ エージェントは、外部からのメッセージあるいはイベント情報、若しくはシステムが想起している BDI+OR 情報の中からまだ未観測のものを観測して m エージェントに書き込んでいく。 stm エージェントは $sensor$ エージェントが観測したメッセージを解釈終了まで一時的に保持する。メッセージ解釈は

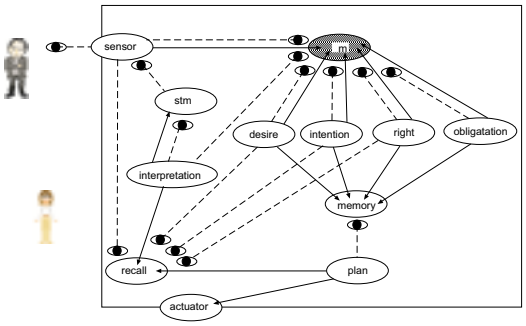


図 1　アーキテクチャ

interpretation エージェントが想起規則 RR_1 に基づいて行い、結果を *recall* エージェント内に表示する。*desire* エージェントと *intention* エージェントと *right* エージェントは欲求 D と意図 I と権利 R を管理し、活性状態のものを *memory* エージェント内に表示し、活性状態でなくなったものを m エージェント内から抹消する。*obligation* エージェントは義務 O を管理し、未達成の義務を *memory* エージェント内に表示し、達成した義務を m エージェントから抹消する。*plan* エージェントは *memory* エージェント内の BDI+OR 情報を観測し、想起規則 (RR_2 と RR_3 と RR_4) に基づき想起された情報を *recall* エージェントあるいは *actuator* エージェントに表示する。

4.4　*sensor* エージェント

sensor エージェントは、外部からのメッセージとイベント情報、およびシステムが想起している BDI+OR 情報を観測し、その中からまだ m エージェントに書き込まれていないものを各単位時間毎に最大で一つ取捨選択し、m エージェントに書き込んでいく。この観測制限は、システム内に雑多なシンボル情報が一度に流れ込んで解釈処理等が混乱するのを避けるためで、どの情報を選択するかはこのエージェントに託されている。以下は現在の取捨選択の優先順序である (同優先度内ではランダム)。

1. 外部の注目情報源 (話相手等) からの情報
2. 内部で想起された BDI+OR 情報
3. 外部からの情報

5　義務と権利に基づく社会行動システム

5.1　プロトタイプ・システム

これまでに、本システムにおける情報表現 (2 節)、計算モデル (3 節)、実装方法 (4 節) に関して説明してきた。この節では、SUN の SICStus Prolog (v.3.8.5) 上に実装したプロトタイプシステムでの実験に基づき、義務と権利をベースにした社会行動の即応制御がどのように実現できるかを概説する。

例題として、このプロトタイプシステムに対し、Tom という人物から Jo という人物の血液型を尋ねられたという簡単な状況を考える。システムは Tom に対する問いに対しては回答する義務があり、Jo はシステムが管理している病院の患者なので、システムは *doctor* 権限があればデーターベース DB に問い合わせて調べることができる。

5.2　想起規則

この例題で必要となる主要な想起規則は、以下の想起規則と次節 (5.3.) で述べる役割に基づく想起規則の二種類になる。

Example 7

想起規則 *(RR_1)*

(1) $Oi, tom.[ans, tom, q] \leftarrow Mi.[ans, tom, q]@tom$

想起規則 *(RR_2)*

(2) $Dx.(y : p) \leftarrow Dx.[ans, z, y : p] \wedge \neg Bx.(y : p > a)$

(3) $Dx.role(doctor) \leftarrow Dx.(y : p) \wedge Bx.patient(y) \wedge \neg Rx.role(doctor)$

(4) $Ix.[ask, DB, q] \leftarrow Dx.[ask, DB, q]$

(1) の想起規則は、Tom からの問い合わせに対する回答義務を表現している。(2) は、対象 y の属性 p 値を対象 z に報告したい欲求があるが、その値を知らない場合に、その値を知りたいという欲求の想起で、(3) は、対象 y の属性 p 値を知りたいという欲求があり、y は患者であるが、*doctor* 権限がない場合に *doctor* 権限が欲しくなるという欲求の想起である。最後の (4) は、データーベース DB に問い合わせたいという欲求があれば、問い合わそうという意図の想起規則である。

その他に、本論文中では範囲外としてあまり言及してこなかったが、義務達成時に義務情報を抹消するための

規則が必要になる。

Example 8

(5) $\neg Ox, y.[ans, y, q]$

$\leftarrow Ox, y.[ans, y, q] \wedge [ans, y, q > a]@x$

5.3 役割に基づく情報管理 (役割エージェント)

義務と権利に関する多くの規則は、部長や医者等の役割 (権限) という枠組みで管理するのが望ましい[3]。このシステムでは、役割に応じて専門の役割エージェントを用意し、役割に応じた義務と権利の定義が可能になっている。さらに、役割によって得た情報を役割エージェント内で管理することで、異なる役割での越権的な情報処理を防いでいる。以下の想起規則は *doctor* という役割エージェント内で定義されている。

Example 9

想起規則 (RR_2)

(6) $Rx/doctor.[ask, DB, y : p]$

$\leftarrow Dx.(y : p) \wedge Bx.patient(y)$

(7) $Ix/doctor.[ans, z, q > a]$

$\leftarrow Dx.[ans, z, q] \wedge Bx/doctor.(q > a)$

想起規則 (6) は、対象 y の属性 p 値を知りたい欲求があり、 y が患者であれば、データーベース DB に問い合わす権利の想起である。 (7) は、対象 z に質問 q の回答を行いたい欲求があり、 *doctor* 権限においてその回答を知っていれば、 *doctor* という役割条件で回答を行う意図が想起されるという規則である。

5.4 シミュレーション (情報管理)

図 2に実行履歴を示す。観測情報の右側は情報源で、想起の○が未観測の情報、●が観測済みの情報を表している。最初 (時刻 320) に Jo が患者であるという情報を観測している。時刻 331 で Tom からの問い合わせメッセージ (Jo の血液型) を受けた。時刻 332 で Tom からのメッセージを回答義務として解釈し、時刻 333 以降で *doctor* 権限欲求を想起している。時刻 416 で *doctor* 権限を得たので DB への問い合わせを実行し、時刻 423 に情報 (A 型) を得ている。その結果、 Tom への回答意図が想起され、時刻 424 でその意図を認識し、 *doctor* 権限で Tom への回答を行っている。

6 結論と今後の方向

義務と権利に基づいて社会的に自律動作するインタラクティブシステムの内部モデルを提案した。表現形式として BDI モデルを拡張し、計算モデルとして自己反映型即応計算モデルを提案した。

マルチエージェントベースのプロトタイプ実装により、例題として取り上げた *doctor* 権限による質疑応答以外に、商取引での基本的な売買における義務と権利の問題や、自動車の運転における左方優先義務を順守した直進等の例題シミュレーションでシステムがうまく動作することをこれまでに確認している。

Example 10

店主の想起規則 (RR_4)

$Rx/shop.[take, y, m]$

$\leftarrow Ox, y.[sell, y, a] \wedge Bx.price(a, m) \wedge \neg[take, y, m]@x$

運転手の想起規則 (RR_2)

$Rx/driver.[move] \leftarrow Dx.[go] \wedge \neg Bi.exist(c, L)$

今後は、システムに複数の役割がある場合の役割同士間の相互関係モデル、および義務の達成条件の一般的なモデル、そして達成不可能な場合の責任 (義務) のモデルを構築し、システムに実装していきたいと考えている。また、 Sloman 流の役割ベースの義務と権利の管理方式と Wagner のダイアグラムを融合させ、義務と権利のダイアグラムから想起規則を自動生成することで、既存の義務と権利に関する外部モデルとこの論文で提案した内部モデルをリンクさせていきたい。

参考文献

[1] Eduardo Alonso. Right and coordination in multi-agent systems. In *UKMAS'98*, 1998.

[2] Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *In Proc. ICMAS-95*, pp. 41–48, 1995.

[3] Emil C. Lupu and Morris Sloman. Towards a role based framework for distributed systems management. *Journal of Network and Systems Manaement*, Vol. 5, No. 1, pp. 5–30, 1997.

[4] J. J. Ch. Meyer and R. J. Wieringa, editors. *Deontic logic in computer science*. Wiley&Sons, 1993.

[5] Timothy J. Norman, Carles Sierra, and Nick R. Jennings. Right and commitment in multi-agent agreements. In *In Proc. ICMAS-98*, pp. 222–229, 1998.

時刻 t	観測 (SO_t)	想起 (SR_t)
00320	Bi.patient(jo) @ i	
00331	[ans,tom,jo:bt] @ tom	○ Oi,tom.[ans,tom,jo:bt]
00332	Oi,tom.[ans,tom,jo:bt] @ i	○ Di.(jo:bt)
00333	Di.(jo:bt) @ i	○ Di.role(doctor) ● Di.(jo:bt)
00416	Ri.role(doctor) @ i	○ Ri/doctor.[ask,db,jo:bt] ● Di.(jo:bt)
00417	Ri/doctor.[ask,db,jo:bt] @ i	○ Di/doctor.[ask,db,jo:bt] ● Ri/doctor.[ask,db,jo:bt] ● Di.(jo:bt)
00418	Di/doctor.[ask,db,jo:bt] @ i	○ Ii/doctor.[ask,db,jo:bt] ● Di/doctor.[ask,db,jo:bt] ● Ri/doctor.[ask,db,jo:bt] ● Di.(jo:bt)
00419	Ii/doctor.[ask,db,jo:bt] @ i	○ [ask,db,jo:bt] @ i/doctor ● Ii/doctor.[ask,db,jo:bt] ● Di/doctor.[ask,db,jo:bt] ● Ri/doctor.[ask,db,jo:bt] ● Di.(jo:bt)
00423	Bi/doctor.(jo:bt>a) @ db	○ Ii/doctor.[ans,tom,jo:bt>a] ● Ri/doctor.[ask,db,jo:bt] ● Di.(jo:bt)
00424	Ii/doctor.[ans,tom,jo:bt>a] @ i	○ [ans,tom,jo:bt>a] @ i/doctor ● Ii/doctor.[ans,tom,jo:bt>a] ● Ri/doctor.[ask,db,jo:bt] ● Di.(jo:bt)
00425	[ans,tom,jo:bt>a] @ i/doctor	

図 2　シミュレーション

[6] Ichiro Osawa. A computational model of an intelligent agent for natural language dialogue systems. In *Proc. of PRICAI'90*, pp. 209–214, 1990.

[7] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a bdi-architecture. Technical Report 14, AAII, 1991.

[8] Gerd Wagner. Towards agent-oriented information systems. Technical report, Freie University Berlin, 1999.

[9] 大澤一郎. 人間と対話する知能エージェントのモデル. コンピュータ・ソフトウェア, Vol. 13, No. 2, pp. 35–44, 1996.

BDI Logicのsequent calculusによる演繹体系

新出尚之　高田司郎

In this paper, we give a sound and complete deduction system for BDI logic whose base is restricted to propositional CTL, using Gentzen's sequent calculus. In this system we can construct a proof of a formula using Wang's algorithm with some extension. In this way, we can check the provability of a formula straightforwardly, and in addition, can give an easily understandable proof of the formula.

1　はじめに

BDI Logic [2] は, branching time temporal logic の一種である CTL* [1] に, BEL(信念), DESIRE(願望), INTEND(意図) といった心的状態を表す様相を導入して拡張した時相述語論理であり, マルチエージェント環境を含む合理的エージェントの仕様記述 [2] [5] などに用いられている.

BDI Logic を基礎概念においた合理的エージェントの実現手法として, BDI アーキテクチャ [4] [6] が提案されている. しかし, 現状の BDI アーキテクチャは, 動作の基本となるループや心的状態の更新が手続き的に行われているなど, 論理体系との整合性が不十分な箇所が多いため, 仕様の自動検証が行いにくい, 論理的記述力への制約が多いなどの問題点がある.

我々はそれを解決する方法として, BDI アーキテクチャの Logic based な実現を検討している. そのためには, BDI Logic の公理化や, 一階述語論理の導出原理に相当するものの構築が必要となる. しかし, 現状では BDI Logic の公理化は十分とはいえず, ベースを命題論理の CTL に制限した BDI Logic の subset に対する健全で完全な演繹体系が Rao らによって与えられている [3] 程度であるが, これは Hilbert style のもので, 論理式の証明を見通しよく行うには向いておらず, BDI logic で書かれた仕様の論理的検証などにも向いていない.

本論文では, Rao らと同じく命題論理の CTL をベースとする BDI Logic の subset について, Gentzen の sequent calculus [7] による健全で完全な演繹体系を示す. これは, 彼らが与えた tableau 法による BDI logic の論理式の充足可能性の判定アルゴリズム [3] に, ほぼそのまま対応するものとなっている. また, この体系では, 命題論理の証明手続きとして知られる Wang のアルゴリズムを拡張したものを用いて, 論理式の証明可能性のチェックを見通しよく行え, 証明も人間にとって直感的に理解しやすいものが得られる.

2　論理式の定義

命題記号の集合 P をあらかじめ決めておく. 命題論理の CTL をベースとする BDI Logic の論理式は, 以下のように定義される.

- $p \in P$ ならば p は論理式 (これを原子命題という)
- ϕ と ψ が論理式ならば $\phi \vee \psi$, $\neg\phi$, $\mathsf{AX}\,\phi$, $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$, $\mathsf{E}(\phi\ \mathsf{U}\ \psi)$, $\mathsf{BEL}(\phi)$, $\mathsf{DESIRE}(\phi)$, $\mathsf{INTEND}(\phi)$

Deduction Systems for BDI Logics Using Sequent Calculus.

NIDE Naoyuki, 奈良女子大学理学部情報科学科. Nara Women's University.

Shiro Takata, ATR メディア情報科学研究所. ATR Media Information Science Lab.

もそれぞれ論理式

必要に応じて括弧で曖昧さを除去. $true$ は $q \vee \neg q$ の略記 (但し q は適当に選んで固定した P の要素), $false$, $\phi \wedge \psi$, $\phi \supset \psi$, $\phi \Leftrightarrow \psi$ はそれぞれ $\neg true$, $\neg((\neg\phi) \vee (\neg\psi))$, $(\neg\phi) \vee \psi$, $(\phi \supset \psi) \wedge (\psi \supset \phi)$ の略記, また $\mathsf{EX}\,\phi$, $\mathsf{AF}\,\phi$, $\mathsf{EF}\,\phi$, $\mathsf{AG}\,\phi$, $\mathsf{EG}\,\phi$ はそれぞれ $\neg\,\mathsf{AX}\neg\phi$, $\mathsf{A}(true\ \mathsf{U}\ \phi)$, $\mathsf{E}(true\ \mathsf{U}\ \phi)$, $\neg\,\mathsf{EF}\neg\phi$, $\neg\,\mathsf{AF}\neg\phi$ の略記. 優先順位は, 単項論理記号 ($\neg$, AX など), $\wedge$, $\vee$, $\supset$ の順とし, 冗長な括弧は適宜省く.

3 意味論

3.1 BDI^{K}_{CTL}-structure

以下のものを定めておく.

- 可能世界の集合 $W(\neq \emptyset)$
- W の各要素 w に対して, state の集合 $St_w(\neq \emptyset)$ とその上の関係 $R_w \subset St_w \times St_w$ (但し, R_w は total であること. すなわち, 全ての $t \in St_w$ に対し, ある $t' \in St_w$ が存在して $t\,R_w\,t'$ を満たさねばならない)
- W の各要素 w と St_w の要素 t の組に対し, 命題記号への真偽値割り当て $L(w,t) \subseteq P$
- $W, \bigcup_{w\in W} St_w$, W 上の 3 項関係 $\mathcal{B},\mathcal{D},\mathcal{I} \subset W \times \bigcup_{w\in W} St_w \times W$. 但し, 条件
  - $(w,t,w') \in \mathcal{B}$ かつ $t \in St_w$ ならば $t \in St_{w'}$
  - $(w,t,w') \in \mathcal{D}$ かつ $t \in St_w$ ならば $t \in St_{w'}$
  - $(w,t,w') \in \mathcal{I}$ かつ $t \in St_w$ ならば $t \in St_{w'}$

  が満たされねばならない[†1]

このとき, tuple $M = \langle W, \{St_w \mid w \in W\}, \{R_w \mid w \in W\}, L, \mathcal{B},\mathcal{D},\mathcal{I}\rangle$ を BDI^{K}_{CTL}-*structure* と呼ぶ. また, それがさらに条件

- (**B-*D***) 任意の $w \in W$ に対し, $(w,t,w') \in \mathcal{B}$ を満たす $w' \in W$ が存在する
- (**B-*4***) 任意の $w,w',w'' \in W$ に対し, $(w,t,w') \in \mathcal{B}$ かつ $(w',t,w'') \in \mathcal{B}$ ならば $(w,t,w'') \in \mathcal{B}$
- (**B-*5***) 任意の $w,w',w'' \in W$ に対し, $(w',t,w) \in \mathcal{B}$ かつ $(w',t,w'') \in \mathcal{B}$ ならば $(w,t,w'') \in \mathcal{B}$
- (**D-*D***) 任意の $w \in W$ に対し, $(w,t,w') \in \mathcal{D}$ を満たす $w' \in W$ が存在する
- (**I-*D***) 任意の $w \in W$ に対し, $(w,t,w') \in \mathcal{I}$ を満たす $w' \in W$ が存在する

を全て満たせば $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-*structure* と呼ぶ.

直感的には, state が一般の時相論理での「点時刻」, あるいは一般の様相論理の可能世界意味論での「可能世界」にあたり, R_w が時間の前後関係, $\mathcal{B},\mathcal{D},\mathcal{I}$ が様相としての信念・願望・意図を表す可視関係である.

3.2 論理式の解釈と恒真性

$w \in W$ に対し, St_w に属する state の無限列 $(t_0,t_1,\cdots)$ で, 条件 $t_0\,R_w\,t_1$, $t_1\,R_w\,t_2,\cdots$ を満たすものを, t_0 で始まる w 上の *path* という.

BDI^{K}_{CTL}-structure $M = \langle W, \{St_w \mid w \in W\}, \{R_w \mid w \in W\}, L, \mathcal{B},\mathcal{D},\mathcal{I}\rangle$ と世界 $w \in W$, state $t \in St_w$ に対し, 論理式 ϕ が真である ($(M,w,t) \models \phi$ と書く) ための条件を, 以下のように定義する.

- ϕ が primitive proposition である ($\phi \in P$) 場合, $(M,w,t) \models \phi$ iff $\phi \in L(w,t)$
- $(M,w,t) \models \neg\phi$ iff $(M,w,t) \not\models \phi$
- $(M,w,t) \models \phi \vee \psi$ iff $(M,w,t) \models \phi$ or $(M,w,t) \models \psi$
- $(M,w,t) \models \mathsf{AX}\,\phi$ iff for all t' s.t. $t\,R_w\,t'$, $(M,w,t') \models \phi$
- $(M,w,t_0) \models \mathsf{A}(\phi\ \mathsf{U}\ \psi)$ iff
  t_0 で始まる w 上の任意の path $(t_0,t_1,\cdots)$ に対し, その中のある state t_i があって $(M,w,t_i) \models \psi$ かつ for all j s.t. $0 \le j < i$, $(M,w,t_j) \models \phi$
- $(M,w,t_0) \models \mathsf{E}(\phi\ \mathsf{U}\ \psi)$ iff
  t_0 で始まる w 上のある path $(t_0,t_1,\cdots)$ とその中のある state t_i があって, $(M,w,t_i) \models \psi$ かつ for all j s.t. $0 \le j < i$, $(M,w,t_j) \models \phi$
- $(M,w,t) \models \mathsf{BEL}(\phi)$ iff for all w' s.t. $(w,t,w') \in \mathcal{B}$, $(M,w',t) \models \phi$
- $(M,w,t) \models \mathsf{DESIRE}(\phi)$ iff for all w' s.t. $(w,t,w') \in \mathcal{D}$, $(M,w',t) \models \phi$
- $(M,w,t) \models \mathsf{INTEND}(\phi)$ iff for all w' s.t. $(w,t,w') \in \mathcal{I}$, $(M,w',t) \models \phi$

論理式 ϕ が, 全ての BDI^{K}_{CTL}-structure M の全ての世界 w, state t に対し $(M,w,t) \models \phi$ を満たすと

†1 Rao らの論文 [2] [3] ではこの条件を明示的に要請してはいないが, BEL などの様相記号の意味の定義が整合的であるためには明らかに必要であろう.

き, ϕ は BDI^K_{CTL}-*valid* であるといい, $\mathrm{BDI}^K_{CTL} \models \phi$ と書く. また, 全ての $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure M について同様の条件を満たすとき, ϕ は $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-*valid* であるといい, $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL} \models \phi$ と書く (いずれも, 誤解のおそれのない場合単に $\models \phi$ と書き, 単に valid であるということがある).

4 演繹体系

本章では演繹体系 $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ および BDI^K_{CTL} を導入する. なお本稿では, sequent の $\rightarrow$ の両側は論理式の multi set とする (従って exchange の推論規則は必要ない). また本稿では, 文中で sequent の範囲を明示するために, sequent 全体を [ ] でくくることがある.

以下, ギリシア大文字は論理式の multi set, ギリシア小文字は単一の論理式を表す. また, 論理式の multi set Γ と単項論理演算子 K に対し, $K(\Gamma)$ で Γ の各論理式に K を付加して得られる multi set を表す.

4.1 推論規則

$(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ の演繹体系は, 表 1 の推論規則のうち BEL–K, DESIRE–K, INTEND–K 以外の全て (これらを含めてもよいが冗長), また BDI^K_{CTL} の演繹体系は同表のうち BEL–KD45, DESIRE–KD, INTEND–KD 以外の全てを推論規則として持つ.

これらの規則は, 実は Rao らの BDI logic の決定アルゴリズム [3] の pseudo tableau の構築の各 step に対応している. 例えば規則 BEL–KD45 は, [3] の 6.2 節, pseudo-$(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-tableau の構築における (d) に相当する.

この体系では, 前提が結論より大きくなる規則が存在するので, Wang のアルゴリズムをそのまま適用はできない. しかし, 6 章で述べるように, Weak の形を制限することによって, 証明図が必ず有限で止まるかループし, 無限に伸びないようにすることができる. 従って証明可能性の判定は有限の手間で行える.

4.2 証明可能性の定義

ここでは, BDI^K_{CTL} および $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ の演繹体系での, sequent の証明可能性の定義を与える.

定義 4.1 S を sequent とする. 各ノードが 1 つの sequent をラベルとして持ち, 根のラベルが S であり, かつ葉以外の任意のノード N が条件

> N の子ノードが $N_1, \cdots, N_i$ であり, N および $N_1, \cdots, N_i$ のラベルが $S', S_1, \cdots, S_i$ であるならば, $\dfrac{S_1 \quad \cdots \quad S_i}{S'}$ という推論規則が存在する

を満たす木を, S **の推論木**という.

以下では, $\rho = \mathsf{A}(\phi\ \mathsf{U}\ \psi)$ の場合は $\mathsf{AX}\rho$ を, また $\rho = \mathsf{E}(\phi\ \mathsf{U}\ \psi)$ の場合は $\mathsf{EX}\rho$ を, Æ$\mathsf{X}\rho$ と表記する.

定義 4.2 S の推論木 T の葉でないノード N_0 から, その子孫の葉ノード L までの経路 $N_0, N_1, \cdots, N_n$ $(= L)$ が, 以下の条件を全て満たすとき, この経路を**導出のループ**という.

1. N_0 と L は等しいラベルを持つ
2. $N_1, \cdots, N_{n-1}$ の中には L と等しいラベルを持つものはない

定義 4.3 ある論理式 $\rho = \mathsf{A}(\phi\ \mathsf{U}\ \psi)$ あるいは $\rho = \mathsf{E}(\phi\ \mathsf{U}\ \psi)$ が存在して, 導出のループ $N_0, N_1, \cdots, N_n$ とその各ノードのラベル $S_0, S_1, \cdots, S_n (= S_0)$ が以下の条件を全て満たすとき, これを**終局性論理式 ρ を持つ導出のループ**という.

1. $S_i = [\rho, \Gamma \rightarrow \Delta]$ かつ $S_{i+1} = [\phi, \text{Æ}\mathsf{X}\,\rho, \Gamma \rightarrow \Delta]$ を満たす整数 $i\,(0 \leq i < n)$ と Γ, Δ が存在する. また, $S_i = [\rho, \Gamma \rightarrow \Delta]$ かつ $S_{i+1} = [\psi, \Gamma \rightarrow \Delta]$ を満たす整数 $i\,(0 \leq i < n)$ と Γ, Δ は存在しない. (大まかには, S_0 から S_n までの過程のどこかで, ρ に対する AU 左か EU 左が適用されており, かつ, 導出のループはその前提のうち右へ行く経路)
2. $S_i = [\Gamma, \Gamma' \rightarrow \Delta, \Delta']$ および $S_{i+1} = [\Gamma \rightarrow \Delta]$ を満たす全ての整数 $i\,(0 \leq i < n)$ と $\Gamma, \Gamma', \Delta, \Delta'$ に対し, Γ' に含まれる全ての論理式は
   - 原子命題
   - トップレベルのオペレータが BEL, DESIRE, INTEND のいずれかである論理式
   - Γ に含まれる論理式

   のいずれかである. また, Δ' に含まれる全ての論理式は
   - 原子命題
   - トップレベルのオペレータが BEL, DESIRE, INTEND のいずれかである論理式

表 1　$(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$, BDI^{K}_{CTL} **の推論規則**

$$\frac{\Gamma \to \phi, \Delta}{\neg\phi, \Gamma \to \Delta}\ (\neg\text{左}) \qquad \frac{\phi, \Gamma \to \Delta \quad \psi, \Gamma \to \Delta}{\phi \vee \psi, \Gamma \to \Delta}\ (\vee\text{左}) \qquad \frac{\phi, \Gamma \to \Delta}{\Gamma \to \neg\phi, \Delta}\ (\neg\text{右}) \qquad \frac{\Gamma \to \phi, \psi, \Delta}{\Gamma \to \phi \vee \psi, \Delta}\ (\vee\text{右}) \qquad \frac{\Gamma \to \Delta}{\Gamma, \Gamma' \to \Delta, \Delta'}\ (\text{Weak})$$

$$\frac{\psi, \Gamma \to \Delta \quad \phi, \mathsf{AX}\,\mathsf{A}(\phi\ \mathsf{U}\ \psi), \Gamma \to \Delta}{\mathsf{A}(\phi\ \mathsf{U}\ \psi), \Gamma \to \Delta}\ (\mathsf{AU}\ \text{左}) \qquad \frac{\Gamma \to \psi, \phi, \Delta \quad \Gamma \to \psi, \mathsf{AX}\,\mathsf{A}(\phi\ \mathsf{U}\ \psi), \Delta}{\Gamma \to \mathsf{A}(\phi\ \mathsf{U}\ \psi), \Delta}\ (\mathsf{AU}\ \text{右})$$

$$\frac{\psi, \Gamma \to \Delta \quad \phi, \mathsf{EX}\,\mathsf{E}(\phi\ \mathsf{U}\ \psi), \Gamma \to \Delta}{\mathsf{E}(\phi\ \mathsf{U}\ \psi), \Gamma \to \Delta}\ (\mathsf{EU}\ \text{左}) \qquad \frac{\Gamma \to \psi, \phi, \Delta \quad \Gamma \to \psi, \mathsf{EX}\,\mathsf{E}(\phi\ \mathsf{U}\ \psi), \Delta}{\Gamma \to \mathsf{E}(\phi\ \mathsf{U}\ \psi), \Delta}\ (\mathsf{EU}\ \text{右})$$

$$\frac{\Gamma \to \Theta}{\mathsf{AX}(\Gamma) \to \mathsf{AX}(\Theta)}\ (\mathsf{AX}\text{–KD}) \qquad \frac{\Gamma \to \phi}{\mathsf{BEL}(\Gamma) \to \mathsf{BEL}(\phi)}\ (\mathsf{BEL}\text{–K}) \qquad \frac{\Gamma, \mathsf{BEL}(\Gamma) \to \mathsf{BEL}(\Delta), \Theta, \mathsf{BEL}(\Theta)}{\mathsf{BEL}(\Gamma) \to \mathsf{BEL}(\Delta), \mathsf{BEL}(\Theta)}\ (\mathsf{BEL}\text{–KD45})$$

$$\frac{\Gamma \to \phi}{\mathsf{DESIRE}(\Gamma) \to \mathsf{DESIRE}(\phi)}\ (\mathsf{DESIRE}\text{–K}) \qquad \frac{\Gamma \to \Theta}{\mathsf{DESIRE}(\Gamma) \to \mathsf{DESIRE}(\Theta)}\ (\mathsf{DESIRE}\text{–KD})$$

$$\frac{\Gamma \to \phi}{\mathsf{INTEND}(\Gamma) \to \mathsf{INTEND}(\phi)}\ (\mathsf{INTEND}\text{–K}) \qquad \frac{\Gamma \to \Theta}{\mathsf{INTEND}(\Gamma) \to \mathsf{INTEND}(\Theta)}\ (\mathsf{INTEND}\text{–KD})$$

(但し, Θ は高々1 つの論理式とする)

- Δ に含まれる論理式

のいずれかである.

(大まかには, S_0 から S_n までの過程で Weak を適用して, AX, AU, EU をトップレベルに持つ論理式が消されることはない)

定義 4.4　sequent S が**証明可能**であるとは, 以下の条件を全て満たす S の推論木 T が存在することをいう. ここで *initial sequent* とは, $[\Gamma, \phi \to \Delta, \phi]$ という形の sequent である.

1. T の全ての葉は, initial sequent をラベルとして持つか, あるいは導出のループの終点である
2. T の全ての導出のループを合わせたグラフ (導出のループに含まれるノードと枝全てからなるグラフ) の各連結成分 T'(これは T の部分木になる) に対し, ある論理式 ρ があって, T' に含まれる T の導出のループは全て, 終局性論理式 ρ を持つ

BDI^{K}_{CTL} および $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ の演繹体系において, $[\to \phi]$ という sequent が証明可能であるとき, 論理式 ϕ が証明可能であるという.

4.3　例

図 1 は, $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ における論理式 $\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(q)) \supset \mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(\mathsf{BEL}(q)))$ の証明である (スペースの都合で図の一部を持ち上げている). この論理式は $\neg\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(q)) \vee \mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(\mathsf{BEL}(q)))$ の略記であるため, 証明の一番下はそれに $\to$ を前置したものになっている. なお, 下から 3 段目の $\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(q)) \to \mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(\mathsf{BEL}(q)))$ から一番右の葉までが, 終局性論理式 $\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(q))$ を持つ導出のループになっていることに注意.

この証明の下から 3 段目以上は, 直感的には「$\mathsf{BEL}(q)$ ならば $\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(\mathsf{BEL}(q)))$」と, 「『$\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(q))$ ならば $\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(\mathsf{BEL}(q)))$』がある時刻で成り立つならばその直前の時刻でも成り立つ」を示すことによって, 数学的帰納法に近い感覚で「全ての時刻で『$\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(q))$ ならば $\mathsf{A}(p\ \mathsf{U}\ \mathsf{BEL}(\mathsf{BEL}(q)))$』が成り立つ」を示していることになる. このように, 得られる証明は, Hilbert style の演繹体系に比べ, 直感的な理解しやすさで優るものになる.

また, 全く同様にして, 例えば $\mathsf{A}(\mathsf{INTEND}(\mathsf{AF}\,\phi)\ \mathsf{U}\ \mathsf{BEL}(\phi)) \supset \mathsf{A}(\mathsf{INTEND}(\mathsf{AF}\,\phi)\ \mathsf{U}\ \mathsf{BEL}(\phi) \vee \neg\,\mathsf{BEL}(\mathsf{EF}\,\phi))$ の証明もできる. これは, コミットメント戦略 [2] の blind commitment が single-minded commitment より強いことの証明になっている.

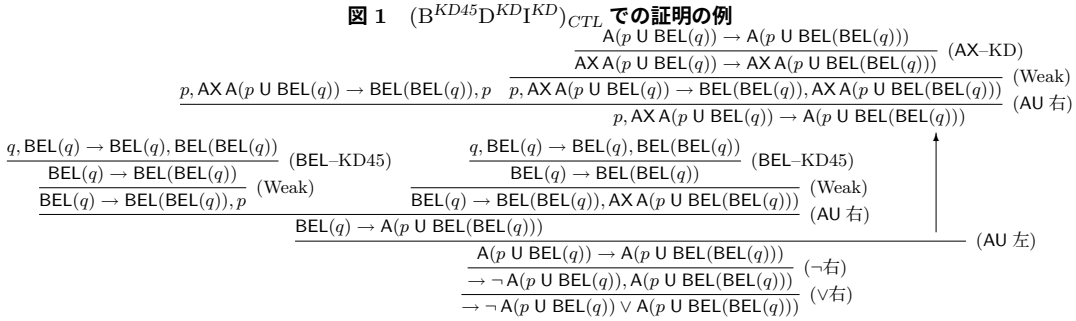
5　健全性

本章では $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ の健全性を示す. BDI^{K}_{CTL} の健全性もほぼ同様に示せる.

sequent $S = [\phi_1, \phi_2, \cdots, \phi_n \to \psi_1, \psi_2, \cdots, \psi_m]$ $(n \geq 0 \leq m)$ に対し, 論理式 $\neg\phi_1 \vee \cdots \vee \neg\phi_n \vee \psi_1 \vee \cdots \vee \psi_m$ (但し $n = m = 0$ のときは *false*) を, S に対応する論理式と呼ぶ. S に対応する論理式が (M, w, t) で真のとき, (M, w, t) で S が真であるといい, $(M, w, t) \models S$ と書く. 特に, sequent S が valid であるとは, S に対応する論理式が valid であることと定義する.

補題 5.1　導出のループの中には, 空 sequent $[\to]$ は現れない.

図 1　$(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ **での証明の例**



証明　定義から, 導出のループは長さ 2 以上でなければならないが, 結論を空, 前提を非空にできる推論規則は存在しない. □

補題 5.2　導出のループの中では, 推論規則 AX–KD か BEL–KD45 のいずれかが少なくとも 1 回適用されており, かつ, その両方は使われていない. また, DESIRE–KD, INTEND–KD は使われていない.

証明　導出のループの中では, 前提が結論より小さくない推論規則が使われていなければならない. 従って, AU/EU 左·右のいずれかか, または BEL–KD45 が使われている.

もし AU/EU 左·右が使われているならば, そこで AX より内側にない AU(または EU) を 1 つ減らしている (ここで「減る」とは, S_i に対して S_{i+1} で, すなわち結論に対して前提で減っていることを意味する. 以下も同様). よって, ループであるためにはそれをどこかで増やさなければならないが, それができる規則は AX–KD か BEL–KD45 だけである. 従って AX–KD[†2] あるいは BEL–KD45 が使われている.

仮に, その両方が使われているとする. BEL–KD45 の結論は補題 5.1 より空ではないので, それに現れる論理式のうち, BEL を最も多く持つもの全ての集合を BEL(Ξ) とする. すると, この先 Weak が適用されるまでは,「sequent に現れる論理式のうち BEL を最も多く持つもの全ての集合は BEL(Ξ) である」という事実は変わらない.

しかし, AX–KD を適用するには, Weak で BEL(Ξ) を消さねばならず, この時, sequent に現れる各論理式が持つ BEL の個数の最大は減る. そして, それを増やせる規則は存在しない. 従って, BEL–KD45 が使われているならば AX–KD は使われていない.

また, DESIRE–KD が使われているとすると, sequent に現れる各論理式が持つ DESIRE の個数の最大がそこで減るが, これを増やす規則はない. 従って, DESIRE–KD は使われていない. INTEND–KD も同様である. □

補題 5.3　終局性論理式 ρ を持つ導出のループの中では, 推論規則 AX–KD が少なくとも 1 回適用されており, かつ BEL–KD45, DESIRE–KD, INTEND–KD は使われていない. また, AX–KD のある適用から次の適用までの間には, ρ に対する AU/EU 左が少なくとも 1 回適用されている.

証明　終局性論理式 ρ を持つ導出のループの中には, AU/EU 左が適用されて ρ が ÆXρ になっている箇所がある. Weak の形の制限 (定義 4.3 の 2) から, ρ や ÆXρ は Weak では消せないので, sequent に現れる論理式のトップレベルを BEL ばかりにすることはできず, BEL–KD45 は使えない. 従って補題 5.2 から, AX–KD が使われており, DESIRE–KD, INTEND–KD は使われていない.

また, 同じ理由から, ρ は, AU/EU 左の適用で ÆXρ に変化しては, AX–KD の適用で ρ に戻ることを繰り返す. 従って AX–KD の適用から適用までの間には, ρ に対する AU/EU 左が必ず適用されている. □

補題 5.4　AX–KD, BEL–KD45, INTEND–KD,

†2　但しこの証明では, ループを 1 周して戻ってきた時の ρ は, AU/EU 左の適用で得られた ÆXρ の ÆX を AX–KD で除去して得たものであるとは必ずしも保証していないことに注意. 他の論理式の部分論理式であった ρ が分解されて出てきた可能性を排除していないからである.

DESIRE–KD 以外のある推論規則の前提が $S_1, \cdots, S_n$, 結論が S であり, かつ $(M,w,t) \not\models S$ となる (M,w,t) が存在するならば, $(M,w,t) \not\models S_i$ となる $i\,(1 \leq i \leq n)$ が存在する.

また, AX–KD の前提が S_1, 結論が S であり, かつ $(M,w,t) \not\models S$ となる (M,w,t) が存在するならば, $t\,R_w\,t'$ かつ $(M,w,t') \not\models S_1$ となる t' が存在する.

証明　論理式の解釈の定義より直ちに従う. □

補題 5.5　ρ が $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ か $\mathsf{E}(\phi\ \mathsf{U}\ \psi)$ のいずれかであるとする. $(M,w,t) \not\models [\rho, \Gamma \rightarrow \Delta]$ かつ $(M,w,t) \models [\psi, \Gamma \rightarrow \Delta]$ ならば, $(M,w,t) \models \rho$ かつ $(M,w,t) \not\models \psi$ である.

証明　論理式の解釈の定義より直ちに従う. □

補題 5.6　終局性論理式 $\rho = \mathsf{E}(\phi\ \mathsf{U}\ \psi)$ を持つ導出のループの中で, ノード N と N' の間で AX–KD が適用されており (すなわち, N のラベル S が AX–KD の結論で, N' のラベル S' がその前提), しかも $(M,w,t) \not\models S$ とする. すると, $t\,R_w\,t'$ なる任意の t' に対し, $(M,w,t') \not\models S'$ iff $(M,w,t') \models \rho$ となる.

証明　ある Γ が存在して $S = [\mathsf{AX}(\Gamma) \rightarrow \mathsf{AX}\neg\rho]$, $S' = [\Gamma \rightarrow \neg\rho]$ と書けることから, 論理式の解釈の定義より直ちに従う. □

定理 5.7　体系 $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ は健全である.

証明　sequent S の推論木 T の全ての導出のループを合わせたグラフ G の連結成分の個数 j に関する帰納法で示す. $j = 0$ の場合は trivial なので, $j < k\,(k > 0)$ の場合を仮定して $j = k$ の場合を示す.

S が valid でないとする. 全ての推論規則は, 前提が全て valid ならば結論も valid であるので, 結論が valid でないなら前提のどれかが valid でない. 従って T の根から子へ, ラベルが valid でないノードを選んでたどっていくことができる. するといつかは, G のある連結成分 T' の根 R に着く. なぜなら, そうでないと導出のループの終点でない葉に着いてしまい, 矛盾するからである.

T' に含まれる T の導出のループが共通に持つ終局性論理式を ρ とする. $\rho = \mathsf{A}(\phi\ \mathsf{U}\ \psi)$ または $\rho = \mathsf{E}(\phi\ \mathsf{U}\ \psi)$ である.

R からさらに子へ, ラベルが valid でないノードを選んでたどっていく. このとき T' の外に出ることはない. なぜなら, R の子孫のうち T' に属さないノードのラベルは, 帰納法の仮定から valid だからである.

するといつかは, あるノード N' とその子ノード N_0 の間で, ρ に対する AU 左か EU 左が適用されている箇所に着く.

N', N_0 のラベルをそれぞれ S', S_0 とすると, S' が valid でないことから, ある (M,w,t_0) が存在して $(M,w,t_0) \not\models S'$ である. また, ここで適用される AU 左か EU 左の前提のうち左は, 定義 4.3 の 1 から T' の外であり, 従って valid である. よって補題 5.5 から $(M,w,t_0) \models \rho$ かつ $(M,w,t_0) \not\models \psi$ である. また, $(M,w,t_0) \not\models S_0$ も成り立つ.

補題 5.3 より, N_0 からその子ノードへの過程で使われている推論規則は BEL–KD45, DESIRE–KD, INTEND–KD 以外のいずれかである. 補題 5.4 より, ここで使われている推論規則が AX–KD 以外であれば, $(M,w,t_0) \not\models S_1$ を満たすラベル S_1 を持つ N_0 の子ノード N_1 があり, また, AX–KD であれば, N_0 の唯一の子ノード N_1 のラベル S_1 に対し, $t_0\,R_w\,t_1$ かつ $(M,w,t_1) \not\models S_1$ となる t_1 が存在する. いずれの場合も N_1 は T' の中にある.

以下, N_0 から N_1 への過程と同様にして, N_1 からそのある子ノード N_2 へ, さらに子ノード N_3 へ…とたどることを繰り返し, T の葉 (T' の葉でもある) にたどり着いたら, それと同じラベルを持つ T' の (葉でない) ノードに戻ってさらに続けていくことにより, ノードの無限列 $N_0, N_1, N_2, \cdots$ とそのラベル $S_0, S_1, S_2, \cdots$, および w 上の path $t_0, t_1, \cdots$ を, $(M,w,t_{applAX(i)}) \not\models S_i$ となるようにとることができる. ここで $applAX(i)$ とは, S_0 から S_i までの間に AX–KD が適用された回数である.

補題 5.3 より, S_0 の $\rightarrow$ の左にある $\text{Æ}\mathsf{X}\rho$ は, AX–KD を適用されて ρ に変わっては, AU 左か EU 左の適用によって $\text{Æ}\mathsf{X}\rho$ に戻ることを繰り返す. N' から N_0 への過程における議論と同様にして, 任意の自然数 ℓ に対し, $(M,w,t_\ell) \models \rho$ かつ $(M,w,t_\ell) \not\models \psi$ となる. すなわち, path $t_0, t_1, \cdots$ において, ずっと ρ が真かつ ψ が偽である. $\rho = \mathsf{A}(\phi\ \mathsf{U}\ \psi)$ であれば, これは論理式の解釈の定義に反する.

一方, $\rho = \mathsf{E}(\phi\ \mathsf{U}\ \psi)$ であれば, path $t_0, t_1, \cdots$ が

上記のようにして取った path であることと, path $t_0, t_1, \cdots$ が任意の自然数 ℓ に対し $(M, w, t_\ell) \models \rho$ を満たすこととは同値であることが, 補題 5.6 を繰り返し適用することにより示せる. しかも, 上記のようにして取った path では, 任意の自然数 ℓ に対し $(M, w, t_\ell) \not\models \psi$ となるので, t_0 から始まる path のうち, ρ が偽にならないうちに ψ が真となるものは存在しない. 従ってやはり論理式の解釈の定義に反する.

以上から, S は valid である. □

6 完全性

本章では $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ の完全性を示す. BDI^{K}_{CTL} の完全性も同様に示せる.

まず 6.1 節で, 証明可能でない sequent S を与えたとき, ある $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure を構築するアルゴリズムを与える. 続いて 6.2 節で, この $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure のある世界において S が偽となることを示す.

6.1 $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure の構築アルゴリズム

S を, 証明可能でない sequent とする. 以下の作業を順に行う.

6.1.1 有限木の構築

まず $S = S_0$ とおき, S_0 に次の '手順A' を適用する.

手順A begin

S_0 に Wang のアルゴリズムを準用し, $\neg/\vee/\mathsf{AU}/\mathsf{EU}$ 左·右を可能な限り適用する. その後, もし各 sequent の $\rightarrow$ の左同士か右同士に同じ論理式が複数回現れていれば, Weak でそれを 1 つに減らす.

手順A end

この結果得られる sequent のうち, 証明可能でないもの全て (1 つ以上有限個) の集合を $\{S'_1, S'_2, \cdots\}$ とおき, そのうち任意の 1 つを S' と置く. S' は, $\rightarrow$ の左右に原子命題とトップレベルが AX, BEL, DESIRE, INTEND の論理式だけを持ち, しかも $\rightarrow$ の左同士や右同士に同じ論理式が複数回現れない.

ただ 1 つの要素 w_0 を持つ集合 W と, ただ 1 つの要素 t_0 を持つ集合 St を作る. また, ペア (w_0, t_0) を唯一のノードとする木 T を作り, このノードに第 1 ラベルとして sequent S, 第 2 ラベルとして sequent S' を貼る.

次いで, $w = w_0$, $t = t_0$ と置いて, S' とペア (w, t) に対し, 以下の '手順B' (図 2) を (再帰的に) 実行する.

手順B begin

S' の $\rightarrow$ の左右が原子命題ばかりであるなら, 何もしない. また, T の根 (w_0, t_0) からノード (w, t) までの経路のどこか (但し (w, t) は含まない) に, 第 2 ラベル S' を持つノードがあるならば, ノード (w, t) に「ループ」というマークを付け, 後は何もしない.

いずれでもなければ, S' に様相論理版の Wang のアルゴリズムを準用する. すなわち, S' に対し以下のいずれかを行って得られる sequent 全ての有限集合を $\{S''_1, \cdots, S''_k\}$ とする.

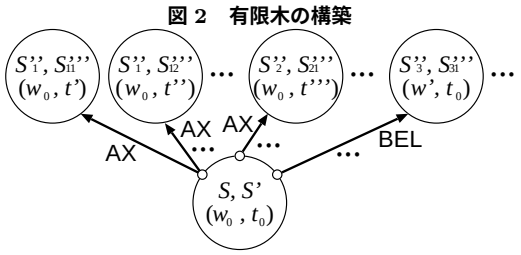
1. Weak で, $\rightarrow$ の左ではトップレベルが AX の論理式以外を全て消す. $\rightarrow$ の右では, トップレベルが AX の論理式があれば, そのうち 1 つを残して他を消し, そうでなければ全て消す.
   次いで AX–KD を適用する.
2. Weak で, $\rightarrow$ の左右それぞれについて, トップレベルが BEL の論理式全てを残し, 他を消す.
   続いて BEL–KD45 を適用する. その際, $\rightarrow$ の右が空なら Δ, Θ を空とおき, そうでなければ $\rightarrow$ の右のうち 1 つを Θ, それ以外を Δ と置く.
3. DESIRE, INTEND に対し, 1. と同様のことを行う.

このとき, $S''_1, \cdots, S''_k$ はいずれも証明可能でない. そこで, 各 S''_i $(1 \le i \le k)$ に対し, $S_0 = S''_i$ と置いて '手順A' を適用し, sequent $S'''_{i1}, S'''_{i2}, \cdots$ を得る. $S'''_{i1}, S'''_{i2}, \cdots$ はそれぞれ, $\rightarrow$ の左右に原子命題とトップレベルが AX, BEL, DESIRE, INTEND の論理式だけを持ち, $\rightarrow$ の左同士や右同士に同じ論理式が複数現れない, 証明可能でない sequent である.

ここで, 各 S'''_{ij} に対し, 以下を行う.

1. $S''' = S'''_{ij}$ と置く.
2. S' から S''_i を得る際に AX–KD, BEL–KD45, DESIRE–KD, INTEND–KD のどれを使ったかによって, 次のいずれかを行う.
   (a) AX–KD を使った場合
   St に新たな要素 t' を加え, T のノード

図 2　有限木の構築



(w,t) に新たな子 (w,t') を作って, それに第 1 ラベルとして sequent S_i'', 第 2 ラベルとして sequent S''' を貼り, 新たな枝に AX というタグを付ける
(直観的には, t から t' に AX による可視関係の枝を張る)

(b) BEL–KD45 を使った場合
W に新たな要素 w' を加え, T のノード (w,t) に新たな子 (w',t) を作って, それに第 1 ラベルとして sequent S_i'', 第 2 ラベルとして sequent S''' を貼り, 新たな枝に BEL というタグを付ける
(直観的には, w から BEL による可視関係で見える新たな可能世界を作る)

(c) DESIRE–KD, INTEND–KD を使った場合
DESIRE, INTEND に対し, 2(b) と同様のことを行う

3. 次いで, 今新設したノードを改めて (w,t) と置き, S''' を改めて S' と置いて, '手順B' を再帰的に呼び出す.

手順B end

どの段階においても, S' は, S に含まれる論理式の部分論理式 (その中に $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ がある場合は $\mathsf{AX}\,\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ を含み, また $\mathsf{E}(\phi\ \mathsf{U}\ \psi)$ がある場合は $\mathsf{EX}\,\mathsf{E}(\phi\ \mathsf{U}\ \psi)$, $\mathsf{AX}\neg\mathsf{E}(\phi\ \mathsf{U}\ \psi)$, $\neg\mathsf{E}(\phi\ \mathsf{U}\ \psi)$ を含む) のみからなり, しかも $\rightarrow$ の左同士や右同士に同じ論理式はない. 従って S' の可能性は有限通りしかない. また, 1 つの S' に対する S_i'' の可能性も有限個しかない. 従って '手順B' は有限の手間で止まる.

このとき, T は W の要素と St の要素のペアをノードとする有限木で, 各ノードには sequent のラベルが 2 つ貼られており, 各枝には AX, BEL, DESIRE, INTEND のいずれかのタグが付いている. また, 一部の葉には「ループ」というマークがつく.

6.1.2　枝刈り

T に対し, 以下の作業を可能な限り繰り返す.

1. 「ループ」や「行き止まり」のマークが付いていないノードで, しかも次のいずれかの条件
   (a) 葉である
   (b) 子のノードに全て「行き止まり」マークが付いている
   を満たすものがあれば, それに「行き止まり」マークを付ける
2. 「行き止まり」マークの付くノード N があり, N と同じ親および同じ第 1 ラベルを持つノード N' $(\neq N)$ があれば, N' を削除 (N' への枝, および N' 以下の部分木も含む)

木の有限性から, この操作は必ず停止し, その結果, 同じ第 1 ラベルを持つ兄弟ノードは「行き止まり」マークの付くもののみ 1 個か, 付かないもののみ 0 個以上のいずれかになる. また, この操作の後も T は木である.

6.1.3　ループ枝の融合

以下の全ての条件

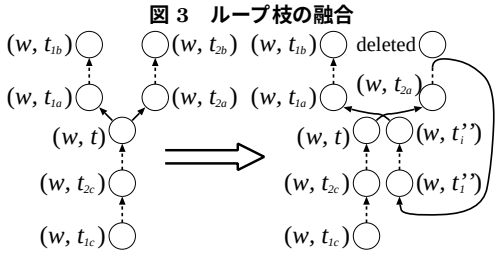
- (w,t_{1a}), (w,t_{2a}) は同じ親 (w,t) および同じ第 1 ラベルを持つノードで, $t_{1a}\neq t_{2a}$
- (w,t_{1b}), (w,t_{2b}) は「ループ」マークのある葉ノードで, (w,t_{1b}) は (w,t_{1a}) の, (w,t_{2b}) は (w,t_{2a}) の子孫
- (w,t_{1c}) は (w,t_{1b}) と, (w,t_{2c}) は (w,t_{2b}) と同じ第 2 ラベルを持つノードで, (w,t_{1c}) は (w,t_{2c}) の, (w,t_{2c}) は (w,t) の先祖

を満たすノード (w,t), (w,t_{1a}), (w,t_{2a}), (w,t_{1b}), (w,t_{2b}), (w,t_{1c}), (w,t_{2c}) が存在する限り, 以下の '手順C' (図 3) を繰り返す.

手順C begin

まず, (w,t_{2c}) から (w,t) までの経路のコピーを作る. すなわち, (w,t_{2c}) から (w,t) までの経路のノードが $(w,t'_1),\cdots,(w,t'_i)$ であるとする (ここで $t'_1=t_{2c}$, $t'_i=t$) と, St に新たな要素 $t''_1,\cdots,t''_i$ を増やし, $(w,t'_1),\cdots,(w,t'_i)$ とそれぞれ同じ第 1・2 ラベルを持つノード $(w,t''_1),\cdots,(w,t''_i)$ を作って (w,t''_1)

図 3　ループ枝の融合



から (w, t''_2) へ, $\cdots$, (w, t''_{i-1}) から (w, t''_i) へタグ AX を持つ枝を張る. また, ノード $(w, t''_1), \cdots, (w, t''_i)$ に「コピー」のマークを付ける.

次いで, (w, t''_1) の第 1 ラベルのみ, (w, t_{2b}) のそれと同じに変更する. 但し (w, t''_1) に「ループ」のマークは付けない.

最後に, (w, t_{2b}) の親から (w, t''_1) へ, および (w, t''_i) から (w, t_{1a}) へタグ AX を持つ枝を張り, (w, t_{2b}) の親から (w, t_{2b}) へ, および (w, t) から (w, t_{1a}) への枝と, ノード (w, t_{2b}) を削除する.

手順C end

直観的にはこの操作は, タグ AX の枝からなるループ全てを含むグラフの各連結成分を, それと同等な枝を全て持つ単一のループに置き換える.

'手順C' 1 回あたり, 条件を満たす箇所が 1 つ減るので, 繰り返しは必ず停止する. この操作の後も T は木であり, また, 1 つのノードから出る, タグ AX を持つ枝で, その先のノードの第 1 ラベルが同じものは高々1 つになる. 特に,「コピー」のマークを持つノードの子はただ 1 つである.

その後,「コピー」のマークのある各ノード N に対し, その祖先で N と同じ第 2 ラベルを持ち「コピー」マークのないノード N' (必ず存在する) を選び, N の「コピー」のマークを外し, N' 以下の部分木 (但し, N の子と同じ第 1 ラベルを持つ N' の子と, それへの枝, およびそれ以下の部分木を除く) を, 必要なら W や St の要素を増やしつつ, N 以下へコピーする. このとき, コピーした部分木においても, ノード (w_1, t_1) からその子ノード (w_2, t_2) への枝のタグが AX であれば $w_1 = w_2$, そうでなければ $t_1 = t_2$ となるようにする. また, コピー元の部分木の中にさらに「コピー」のマークを持つノードがあれば, 先に再帰的に同様のコピーを行う. この再帰のネストは, 高々'手順C' を行った回数なので, この操作は有限で停止する.

6.1.4　$(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure の構築

以下のような tuple $M = \langle W, \{St_w \mid w \in W\}, \{R_w \mid w \in W\}, L, \mathcal{B}, \mathcal{D}, \mathcal{I}\rangle$ を作る.

- 各 $w \in W$ に対し, $St_w = \{t \in St \mid T$ がノード (w, t) を持つ $\}$
- 各 $w \in W$ に対し, $R_w = \{(t, t') \mid T$ は (w, t) から (w, t') へのタグ AX の枝を持つ $\}$
- 各 $w \in W$, $t \in St_w$ に対し, $L(w, t)$ はノード (w, t) の第 2 ラベルである sequent の $\rightarrow$ の左に現れる原子命題全ての集合
- 各 $w, w' \in W$, $t \in St_w$ に対し, $\mathcal{B} = \{(w, t, w') \mid T$ は (w, t) から (w', t) へのタグ BEL の枝を持つ $\}$
- $\mathcal{D}, \mathcal{I}$ に対しては同様

次に, T の根からそれぞれの葉 L への経路のうち, 途中 (葉を除く) に葉と同じ第 2 ラベルを持つノード N があるもの全てについて, L の直前のノード N' から L への枝のタグが何であるかに応じて以下のいずれかを行う.

- タグが AX ならば, 補題 5.2 よりある w, t, t', t'' が存在して $N' = (w, t')$, $L = (w, t)$, $N = (w, t'')$ と書けるので, R_w から (t', t) を削除し (t', t'') を加え, St_w から state t を削除
- BEL ならば, 補題 5.2 よりある w, t, t', t'' が存在して $N' = (w', t)$, $L = (w, t)$, $N = (w'', t)$ と書けるので, $\mathcal{B}$ から (w', t, w) を削除し (w', t, w'') を加え, W から可能世界 w を削除
- DESIRE, INTEND であることは, 補題 5.2 からない

ここで, $t\, R_w\, t'$ を満たす t' の存在しないような全ての w と t に対し, (t, t) を R_w に加える. この場合, T のノード (w, t) の第 1・2 ラベルの $\rightarrow$ の左右には AX をトップレベルに持つ論理式はないことに注意. また, $(w, t, w') \in \mathcal{B}$ を満たす w' が存在しないような全ての w と t に対し, (w, t, w) を $\mathcal{B}$ に加える. この場合も BEL について同様の注意が成り立つ. $\mathcal{D}, \mathcal{I}$ についても同様の操作を行う.

最後に, 3.1 節の条件 (**B-*4***) や (**B-*5***) を満たさない w, w', w'' がある間, (w, t, w'') を $\mathcal{B}$ に加える (世界や

state は有限個なので有限の手間で済む). そのような (w,t,w'') を 1 つ $\mathcal{B}$ に加える前に, 性質

> $(w,t,w') \in \mathcal{B}$ ならば, T のノード (w,t) の第 2 ラベルの $\rightarrow$ の左に現れる全ての $\mathsf{BEL}(\phi)$ に対し, ノード (w',t) の第 1 ラベルの $\rightarrow$ の左には $\mathsf{BEL}(\phi)$ と ϕ が現れる

が満たされているならば, 1 つ加えた後にもこの性質が満たされるので, この操作が全て済んだ後も, この性質は満たされている.

これにより M は $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure になる.

6.2 完全性の証明

定理 6.1 体系 $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$ は完全である.

証明 sequent S が証明可能でないとする. 6.1 節のアルゴリズムで作られた $(\mathrm{B}^{KD45}\mathrm{D}^{KD}\mathrm{I}^{KD})_{CTL}$-structure M は以下の性質を持つ.

> $S_w \ni t$ とする. '手順B' で T のノード (w,t) を生成した段階での S_i'' から S_{ij}''' に至る過程 (但し $w = w_0$, $t = t_0$ の場合は S から S' に至る過程) のどの sequent に対しても, その $\rightarrow$ の左に現れる論理式は (M,w,t) で真であり, 右に現れる論理式は (M,w,t) で真でない.

これを示すには, 論理式の構造に関する帰納法を, M の全ての state に対して同時並行で適用すればよい. 特に, $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$, $\mathsf{E}(\phi\ \mathsf{U}\ \psi)$ に関しては次のように示せる.

- $\rightarrow$ の左の $\mathsf{A}(\phi\ \mathsf{U}\ \psi)\cdots$ もしこれが偽なら, T の作り方から, (M,w,t) で始まるある path で, ずっと ϕ が真, ψ が偽となる. このとき, T のあるノード N とその子孫の葉ノード N' で, N から N' までの経路のノードの第 1 ラベルの $\rightarrow$ の左に常に $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ があり, しかもその経路の枝が全てタグ AX を持つものがある. すると, N の第 1 ラベルには, 終局性論理式 $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ を持つ導出のループを含む証明が存在することになり, 矛盾する.
- $\rightarrow$ の左の $\mathsf{E}(\phi\ \mathsf{U}\ \psi)\cdots$ 同様.
- $\rightarrow$ の右の $\mathsf{A}(\phi\ \mathsf{U}\ \psi)\cdots$ これに対しどこかで AU 右が適用されている. その次の sequent が AU 右の前提の左だとすると, 帰納法の仮定から ϕ も ψ も (M,w,t) で偽なので $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ も偽. 一方, 右だとすると, 同じ理由で ψ は (M,w,t) で偽. また, $t\,R_w\,t'$ を満たすある (w,t') があって, それに対応するノードの生成中に, sequent の $\rightarrow$ の右に $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ が現れている. これに同様の議論を適用して, (M,w,t') で ϕ も ψ も偽か, あるいは ψ が偽でかつ $t'\,R_w\,t''$ を満たすある (w,t'') があって, 同じ議論が繰り返せる. すなわち, t で始まる w 上のある path があって, ずっと ψ が偽か, あるいはあるところで ϕ が偽でかつそこまでずっと ψ が偽である. 従って (M,w,t) で $\mathsf{A}(\phi\ \mathsf{U}\ \psi)$ は偽.
- $\rightarrow$ の右の $\mathsf{E}(\phi\ \mathsf{U}\ \psi)\cdots$ 同様.

従って, 特に $(M,w_0,t_0) \not\models S$ である. よって, S は恒真でない. □

7 判定アルゴリズム

6 章で示したアルゴリズムを, 証明可能な sequent に対しても適用すれば, sequent の証明可能性の判定アルゴリズムが得られる.

与えられた sequent S に Wang のアルゴリズムを準用し, $\neg/\vee/\mathsf{AU}/\mathsf{EU}$ 左·右を可能な限り適用する. その後, もし各 sequent の $\rightarrow$ の左同士か右同士に同じ論理式が複数回現れていれば, Weak でそれを 1 つに減らす.

この結果得られる sequent のうち, 様相オペレータを持たないか, S からの過程で既に現れている sequent と同じものについては何もせず, それ以外の各 sequent S' に対し, 6.1.1 節の '手順B' と同様に, 様相論理版の Wang のアルゴリズムを準用する. 得られた各 sequent S_i'' に対し, S と同様に Wang のアルゴリズムを準用することを繰り返す.

こうして, 定義 4.4 の条件を満たす推論木が作れれば S は証明可能である. また, できなければ, 有限の手間でこの操作は行き詰まり, 証明可能でないとわかる.

現在, このアルゴリズムの Prolog によるインプリメントが動作している (`ftp://ftp.ics.nara-wu.ac.jp/pub/nide/research/bdi-1.00.tgz`).

8　今後の課題

複数のエージェントに関する BDI [5] をもつ logic への拡張も, エージェントの quantifier を入れなければ直ちに行えるだろう. また, mutual belief [5] のように, BDI オペレータの無限ネストに相当するオペレータを持つ体系に関しても, 本稿の AU/EU に対する取り扱いと同様の手法によって, 演繹体系を作れると考えられる. これらを導入できれば, マルチエージェントシステムの BDI logic を用いた仕様記述や検証に有効であろう. さらに, $\mathsf{DESIRE}(\phi) \supset \mathsf{BEL}(\mathsf{DESIRE}(\phi))$ など, 合理的エージェントに要求される心的状態の整合性公理 [2] [3] の導入も検討を要する. それらは今後の課題である.

エージェントの quantifier に関する取り扱いや, また, 現時点ではまだ, ベースが命題論理の CTL に限定されているので, これを CTL* や述語論理に拡張できるかどうかも今後の検討課題である.

参考文献

[1] Emerson, E. A. and Srinivasan, J.: Branching Time Temporal Logic, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*(de Bakker, J., de Roever, W., and Rozenberg, G.(eds.)), Springer-Verlag, 1989, pp. 123–172.

[2] Rao, A. S. and Georgeff, M. P.: Modeling Rational Agents within a BDI-Architecture, *Proc. of International Conference on Principles of Knowlegde Representation and Reasoning*, 1991, pp. 473–484.

[3] Rao, A. S. and Georgeff, M. P.: Decision Procedures for BDI Logics, *Journal of Logic and Computation*, Vol. 8,No. 3(1998), pp. 292–343.

[4] Singh, M. P., Rao, A. S., and Georgeff, M. P.: Formal Methods in DAI: Logic-Based Representation and Reasoning, *Multiagent Systems*, The MIT Press, 1999, pp. 331–376.

[5] Wooldridge, M.: *Reasoning about Rational Agents*, The MIT Press, 2000.

[6] 高田司郎, 五十嵐新女, 新出尚之, 榎本美香, 間瀬健二, 中津良平: マルチエージェント環境において意図的に言語行為を遂行する合理的エージェントの基本設計, **電気情報通信学会論文誌**, Vol. J84-D-I,No. 8(2001), pp. 1191–1201.

[7] 萩谷昌己: **ソフトウェア科学のための論理学**, 岩波講座 ソフトウェア科学 11, 岩波書店, 1994.

マルチエージェントによるチーム航行支援

安藤 英幸　広野 康平　大和 裕幸

1　はじめに

ビジネスや製造のプロセスモデルを整理し、情報や業務の引継ぎの自動化を目指したワークフロー管理システムが注目されている。

ワークフロー管理システムの中心となるのは、プロセスモデルとそれを実行するワークフローエンジンである。一般にプロセスモデルは、コミュニケーション、作業の対象、ユーザーのアクティビティといった視点から有向ネットワークにより表現される。またワークフローエンジンは、プロセスのテンプレートからのインスタンスの作成、タスクの割り当て、実行とモニタリング、例外処理を行う。

これらの枠組みは、定型的にプロセスモデルを記述できる場合には十分である。しかし、とりまく環境の変化に対応してプロセスの更新が必要であったり、プロセスを外部環境との関連で表現するのが適している場合には、プロセスモデルの表現の工夫と、持続的にプラニング、スケジューリング、実行のモニタリングを行うワークフローエンジンが必要となってくる。

本研究では、船舶のブリッジ業務に動的なワークフロー管理システムを実装する事を目的としている。ブリッジの業務は、周囲の船舶や自船の位置といった状況に応じて、実行する業務を適応させる必要がある。このために必要となるプロセス表現の整理とワークフローエンジンの実装を行う。

2　背景

船上業務の種類は、例えば見張り、船位確認といったレベルの業務をユニットとして約 600〜700 ユニット程度であると言われる [1]。それぞれの業務は、Hutchins も述べているように構造の良くわかった問題で、あらかじめ作業手順を定型化する事ができる [2]。しかし、それら定型的な業務も、自船位置と航路の関係、他船の位置と速度、ワッチレベル (watch level) と呼ばれる危険度の指標、他の作業、作業の記録といった多様なコンテクストに応じて、行うべき業務、優先順位、処理に要する時間、繰り返し業務に関する時間間隔等が変更する。

これらの船上業務についての詳細は SMS(Safety Management System) マニュアルという印刷物の形で各運航会社毎にまとめられ、船上で利用されている。しかし、作業の状況への依存性が高い事からマニュアルが複雑化したり、マニュアルの見直しに必要な実作業の履歴の入手が現状では困難であることから、ワークフロー管理システムに SMS マニュアルの情報を移植し、作業の見逃し、対応の遅れの防止、さらには、作業履歴の分析による基のプロセスモデルの改善が期待される。

Team Navigation Support with Multi-Agents System
Hideyuki Ando, 東京大学 新領域創成科学研究科, Graduate School of Frontier Sciences, University of Tokyo
Kouhei Hirono, (株) 日本海洋科学, Japan Marine Science Inc.
Hiroyuki Yamato, 東京大学 新領域創成科学研究科, Graduate School of Frontier Sciences, University of Tokyo

3　エージェントとアーキテクチャ

3.1　エージェントの構成

ブリッジ業務の動的なワークフロー管理システムは、プロセス、外的状況、作業者に関する情報をネットワーク上で交換し、それらに基づいて適切な業務のプラニング、スケジューリング、実行モニタリングを持続的に行う必要がある。これらの実装にマルチエージェントを採用することとし、その実装のベースに、Jin らがワークフローによる協調設計支援のために開発した ActiveProcess 環境を利用する [3]。ActiveProcess の各エージェントは、ネームサーバーを介して KQML メッセージを交換する。コンテンツ言語としては、順序なしリストを表現できる独自のフォーマットを利用している。本研究では、航海士の支援を行うユーザーエージェント、船長につきチーム全体の業務プロセスを管理するマネージャーエージェント、船舶運航に関する情報をもつセンサーデータサーバーのラッパーとして働くリソースエージェントを配置した (図 1)。

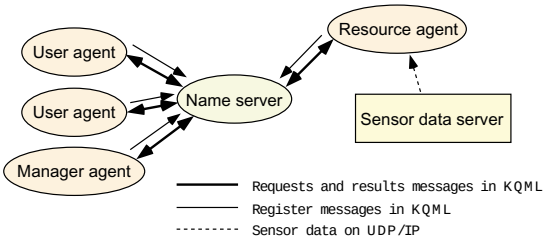


図 1　Multi agents configuration

3.2　多層型アーキテクチャ

ActiveProcess では、自律的な振る舞い、推論、エージェント間の協調といった知的な機構に関しては、ドメインに合わせて開発を行う必要がある。本システムでは、エージェントに求められるタスクの多様性から、即応、熟考、協調の 3 つの機能を独立したレイヤーとして実現し、協調ロボットの実装を良く整理した InterRAP アーキテクチャの概念を基本的に採用することとした [4]。ただし、熟考、協調層の推論部分へのエキスパートシステム JESS(Java Expert System Shell) [5] の利用、各レイヤーのコンポーネントの構成の変更といった改良を加えた。

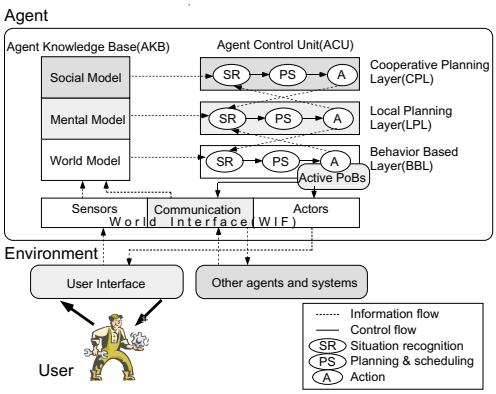


図 2　Agent architecture

エージェントの内部構成を図 2 に表す。全体は World Interface(WIF)、Agent Knowledge Base(AKB)、Agent Control Unit(ACU) によって構成される。WIF はエージェントと外部環境とのインターフェースで、ユーザーインターフェース、他のエージェントとの通信、センサーデータの取得に利用する。エージェントの持つ周囲の環境に関する知識、作業者の行うワークフロー、他エージェントに関する知識は World model, Mental model, Social model の 3 層に分けて AKB に置かれる。また、AKB はそれらの知識の登録、取り出しのためのインターフェースを持つ。ACU では AKB の知識を基に即応 (BBL)、熟考 (LPL)、協調 (CPL) のそれぞれの層が独立して状況の認識、プラニング、アクションを行う。ACU 内のレイヤー間の通信には KQML を使用する。即応層では、Pattern of Behavior(PoB) と呼ばれるエージェントの振る舞いのパターンを記述したモジュールの Active/Inactive の切り替えを行う。PoB には反射的な対応を行うための reactive PoB と他の層からの委託により Active な状態になる procedure PoB がある。

エージェントは全体として、AKB の知識をベースに即応、熟考、協調の各層がプラニング、スケジュー

リングを行い、PoB の実行リスト (Active PoBs) の切り替えを判断している事になる。

3.3 実装

システムは WindowsNT/2K 上で開発を行った。エージェントの開発には Java 言語を利用したが、他のアプリケーションとの連携を容易にするため Visual J++ を用いて COM(Component Object Model) の DLL(Dynamic Link Library) として実装した。また、ユーザーインターフェースの実装は Visual Basic を用いた。

4 プロセスマネージメント

4.1 プロセスモデル

Wilkins らは、現実問題へのプラニング技術の適応を考える場合、ドメイン知識の利用が不可欠である事を指摘している [6]。ドメイン知識を利用したプラニングの代表として HTN(Hierarchical Task Network) がある。HTN ではツリー構造によってプロセスを記述し、ゴールに合わせて上位概念を選択し、下位のツリーを展開し、最終的にプリミティブな操作で構成されるプランを得る。本研究でも、記述が可能であることから、階層構造というドメイン知識を用いたプロセス記述を採用し、特に Jin らが ActiveProcess で利用している Workflow, Activity, Work Element, Operation の 4 階層によるプロセスモデルに従う事とした。プロセスモデルのツリーの各階層の役割を、表 1 に示す。

また、動的な環境の変化に対応して適切なプラニングを行うために、全体のツリーのうちどのサブツリーを駆動するかという知識をエキスパートシステム JESS のルールという形で記述する。(図 3)。

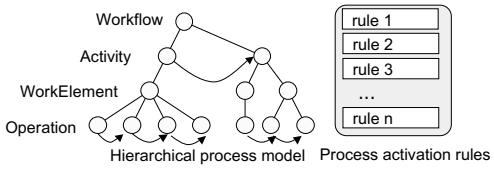


図 3　Process model

Workflow	チーム全体の作業プロセス
Activity	1 人の作業者に割り当てる作業の最小単位
Work Element	Operation の集合
Operation	作業の最小単位

表 1　Role of each layer

4.2 プラニング、スケジューリング

ユーザーの作業のプラニング、スケジューリングは、図 4 の手順で行われる。外部環境の変化にマッチするルールが発火し、実行するべきサブツリーが選択される。選択されたサブツリーは、順序の制約を満たすように展開され、一連のオペレーションにより構成されるプランとなる。これに、それぞれの作業時間を適応してスケジュールが構成される。この作業は継続的に繰り返される。また、現時点で、サブツリーの選択のために用意したルールの例を以下に示す。

- 周囲に船が接近してきた場合に起動
- ワッチレベルに応じた起動
- デフォルトで起動

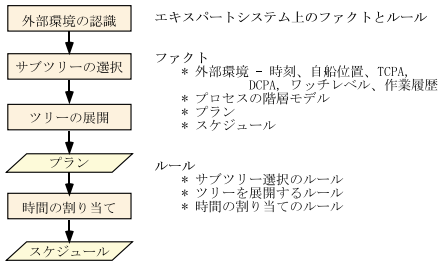


図 4　Planning and scheduling

4.3 実行とモニタリング

熟考層でスケジューリングが行われた作業は、ユーザーインターフェースを通して、ユーザーに対して提示され、その終了までをモニタリングされる。各作業のモニタリングは、図 5 に示す状態遷移規則に基づいて行われる。エージェントとユーザーインターフェー

スの間で、図 5 の矢印の上に示されたフラグを交換し、状態を遷移する。作業のモニタリングは、即応層のレベルの作業モニタリング用 PoB が担当する。開始時には熟考層から Active にするメッセージが即応層に送られ、ユーザーインターフェースを介してユーザーに提示される。作業終了時には、ユーザーからの終了の通知を受けて、熟考層が業務の終了を認識し、先のモニタリング用 PoB を Inactive にするように即応層にメッセージを送る。

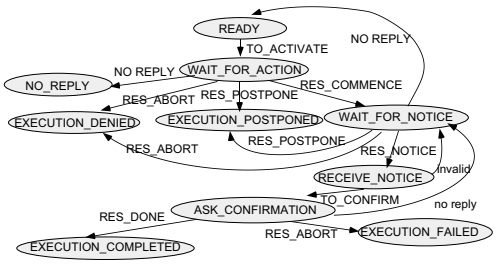


図 5　State transition for operation monitoring

5　検証

開発したエージェントの動作を確認するため、船舶のデスクトップ型の操船シミュレータ（仕様を表 2 に示す）に接続した。自船、他船に関する情報は、レーダー装置を代替するセンサーデータサーバからネットワーク上のリソースエージェントに対して送られる。シミュレーションの海域は、伊豆大島東方で、自船は洲崎沖変針点向け航行中とした。シミュレーションは、ワッチレベル 1 (当直航海士 1 名体制) として開始した。当直航海士は船長から「見張り」、「船位測定」、「避航操船」の 3 つの Activity を与えられている。船長は船橋にいない状態であるが、「ワッチレベルの切り替え」の Activity が常時割り当てられている。この状態で、以下の点についてシステムの動作を確認した。

1. プロセスツリーの展開
2. 作業のモニタリング
3. 動的な環境でのサブツリーの選択

自船データ	1 秒間隔で生成
他船データ	3 秒間隔で生成（一般的なレーダーの周期が 3 秒であるため)。
自船の運動計算	舵角とエンジンテレグラフ値の入力より、応答遅れを考慮して船体運動計算する。
他船の運動計算	予め設定されたコースをトレースさせて運動を再現。変針動作は一定の角速度で回頭させている。
表示	電子海図上に自船位置と他船位置、それぞれの速度ベクトルを重ねて表示。ECDIS と同等

表 2　Specification of the simulator

5.1　プロセスツリーの展開

航海士エージェントは、「見張り」Activity 以下の階層構造を展開し、目視による監視、レーダーによる監視といった Operation を順次促し、次に「船位測定」Activity 以下の階層構造を展開し、下位の Operation の実行を提示し、当直航海士の作業遂行支援を行う事を確認した。

5.2　作業のモニタリング

航海士エージェントは、状態遷移規則に基づき、モニタリング用の PoB を通して、ユーザーインターフェースとの間でメッセージの送受信を行い、選択された Operation が最終的に、正常終了、異常終了、延期、応答なしのいずれかの状態になるまでモニタリングを行った。また、応答が許容時間内にない場合には、入力の催促を行った。

5.3　動的な環境でのサブツリーの選択

「ワッチレベル切り替え」Activity が起動される条件を、この検証では“ TCPA （最接近までの時間）がプラスで、DCPA（最接近時の距離）が 1 マイル以下となる船舶が、同時に、10 マイル以内に 2 隻以上現れたとき ”とした。船長エージェントは、リソースエージェントから TCPA, DCPA 情報を定期的に

取得し、この条件を満たす状態に変化した直後に、船長に対してワッチレベルの切り替え Operation を提示し、船長はワッチレベルを変更した。 航海士エージェントはワッチレベルが変更された事を受け、「避航操船」Activity の担当を外し、代わって船長エージェントが「避航操船」Activity を担当するツリーに加えた。

6　まとめ

船上のブリッジ作業支援というドメインにおける動的ワークフロー管理システムの設計を行うにあたり、プロセスモデルとして、階層構造で表現されるプロセスモデルと、駆動するべきサブツリーを選択する規則を記述したエキスパートシステムのルールを併用した。また、それらを実行するためのワークフローエンジンのアーキテクチャとして、マルチエージェントを採用し、特に、多様なタスクを並行してこなすために、InterRAP の多層アーキテクチャを基に、エキスパートシステムを推論エンジンとして使用するなどの拡張を行った。

これらにより実装した動的ワークフロー管理システムが、環境を認識し、条件に適応するサブツリーを選択し、それを展開し、作業者のスケジュールを作成し、その実行をモニタリングする事を、船舶の操船シミュレーターを利用した実験において確認した。

実用に向けては、まだ非常に多くの課題がある。現時点では、プロセスモデルの準備とルールの数が十分ではなく、この整備が必須となる。また、エージェントの認識できる情報は大きく制限され、例えば気象や海象といった情報を扱うことができない。これらを含めて、どのように人間とエージェントが協調的に作業を行うかを考えていく必要がある。

ワークフローエンジンの機能としては、作業スケジューリングの作成に関して、iterative repair の導入等による、より無理のないスケジュールの更新手法の検討、作業者間の負荷分散、ユーザーによるスケジュール変更を許すためのインターフェースが必要と考えている。

7　謝辞

本研究の遂行にあたり、（株）日本海洋科学の東京大学名誉教授小山健夫先生から多数のご助言を頂きました。また、東京大学助教授の白山晋先生、増田宏先生、海上安全技術研究所の伊藤博子氏には研究の遂行にご協力いただきました。システム開発にあたり、南カリフォルニア大学の Prof. Yan Jin からは ActiveProcess のソースコードの提供と多数のご助言を頂きました。ここに感謝いたします。なお、本研究は平成 12 年度文部省科学研究費補助金による基礎基盤 (A)(2)-1230817 の補助を受けました。

参考文献

[1] 日本造船協会第 238 研究部会. 新しいフリートサポートシステムの研究、平成 10 年度報告書. 社団法人　日本造船研究協会, 1999.

[2] E. Hutchins. The technology of team navigation, 1990.

[3] Yan Jin, Li Zhao, and Arun Raghunath. Activeprocess: A process-driven and agent-based approach to supporting collaborative engineering. In *Proc. of ASME Design Engineering Technical Conference, Las Vegas, Nevada*, 1999.

[4] Jörg P. Müller. *The design of intelligent agents: a layered approach*, Vol. 1177. Springer-Verlag Inc., New York, NY, USA, 1996.

[5] Jess. *http://herzberg.ca.sandia.gov/jess/*.

[6] D. Wilkins and M. desJardins. A call for knowledge-based planning. *AI Magazine*, Vol. 22, pp. 99–115, 2001.

Wizard of Oz法を使ったマルチキャラクタエージェントインターフェースの評価

辻本秀樹　北村泰彦　辰巳昭治

Character-agent interface is a user-friendly interface on the Internet. Most of the systems use a single agent alone, but we here consider the effect of multiple character-agents. We have developed an information retrieval system called Venus & Mars based on multiple character-agents interface. In this paper, we observed and compared interactions between users and agents in two versions of Venus & Mars; one version employs a single character only and another employs multiple characters, by using Wizard of Oz method.

1　はじめに

今日ではパソコンやインターネットが社会に浸透し、今までコンピュータに触れることの少なかった初心者ユーザが利用する機会も増えている。そのような初心者ユーザを含め、ユーザに優しいインターフェースとしてキャラクタエージェントインターフェースがある。またユーザ層が広がることで情報源も広がり、分散した情報の統合が有効となる。ここで、キャラクタエージェントインターフェースのほとんどがシングルエージェントによるものであることから、それらのシステムを組み合わせてマルチキャラクタエージェントインターフェースのシステムとして動作させるということが考えられる。このマルチキャラクタエージェントインターフェースの概観を図 1 に示す。

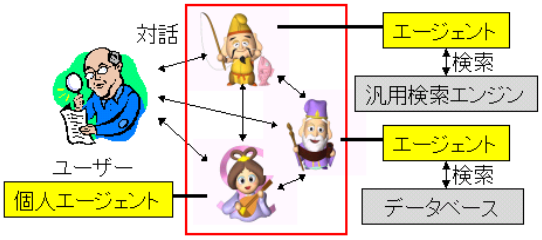


図 1　マルチキャラクタエージェントインターフェース

このようなマルチキャラクタエージェントインターフェースを実装し、それぞれのエージェントが持っている情報を組み合わせて検索を行うことができる、Venus & Mars[1] という情報検索システムがある。本研究ではマルチキャラクタエージェントインターフェースが分散した情報を統合するシステムに有効であるかを調べるため、Venus & Mars のインターフェースをシングルエージェントとマルチエージェントで動作させ、それぞれのユーザとエージェントのインタラクションを Wizard of Oz 法 [3][4][5] を用いて観測し、マルチキャラクタエージェントインターフェースを評価した。これはシステムのインターフェース設計に生かせると考えられる。

2　Venus & Mars：協調型マルチエージェント検索システム

Venus & Mars は情報検索に不慣れなユーザでも、自然言語でより的確なレシピの検索を可能とすることを目標としたもので、Microsoft Agent を利用したマルチキャラクタエージェントインターフェースが実装されて

Hideki Tsujimoto, Yasuhiko Kitamura, Shoji Tatsumi, 大阪市立大学大学院　工学研究科　情報工学専攻, Department of Information and Communication Engineering, Graduate School of Engineering, Osaka City University.

図 2　Venus & Mars の動作画面

いる。エージェントには、レシピまたは健康に関する知識を持ちその領域での情報検索や助言を行う専門エージェント（献立検索エージェント、健康物知りエージェント）と、利用者のプロファイルを情報検索に反映させる個人エージェントが存在する。その動作画面を図 2 に示す。

ユーザは話しかけたいエージェントをクリックし、自然言語のテキストで発話することができる。ユーザや他のエージェントの発話を取得したエージェントはそれぞれ独立に処理を行い、音声による発話と身ぶりを交えて応答する。このときユーザやエージェントの発話は全てのエージェントに伝わっている。

Venus & Mars で登場するエージェントの役割とその行動は以下のようになっている。

- 献立検索エージェント「コン」: レシピを検索するエージェントで、食材、料理名、味付けに関するキーワードを知識として持っている。ユーザまたは他のエージェントの発話に自分の知識と一致するキーワードが有れば、そのキーワードと検索隠し味 [2] と呼ばれる追加キーワードを用いて検索する。また、検索結果が多い場合には絞り込み検索を行うためのキーワードを尋ねる。
- 健康物知りエージェント「チョー」: 健康に関する情報を提供するエージェントで、食材、病気や症状に関するキーワードと、食材が持つ効能、病気や症状に対して効果のある食材を知識として持つ。ユーザまたは他のエージェントの発話に自分の知識と一致するキーワードが有ればそのキーワードについて発話する。
- 個人エージェント「ペッコ」: ユーザの入力を監視するエージェントで、絞り込み検索でユーザが入力した食材や味付けに関するキーワードをユーザの好みとして記憶しておく。献立検索エージェントが絞り込み検索を行うためのキーワードを尋ねたときにそのキーワードを発話する。

これらのエージェントは検索時に 2 つの点で協調する。1 つはコンが絞り込み検索を行うときにペッコがユーザの好みを発話することで、自動的にユーザの好みによる絞り込み検索を行えるように支援する点、もう 1 つは健康に関するキーワードが入力されときにチョーがそれに対して食材を発話することで、健康に関する知識の無いコンがレシピ検索を行えるようになる点である。

3　評価

本研究ではマルチキャラクタエージェントインターフェースの評価のため、Venus & Mars において、そのインターフェースがシングルエージェントとマルチエージェントの場合でのユーザとエージェントのインタラクションを観測し比較した。

3.1　Wizard of Oz

当初行った Venus & Mars をそのまま用いた実験では、それぞれのエージェントの持つ知識が少なく、入力に対する応答や協調が上手くできない場合が多かったため、インタラクションのほとんどがレシピ検索のキーワード入力という結果となった。このような状態を避けるためには、エージェントはより多くの知識を持ち、自然言語での入力を正しく解釈して応答する必要がある。しかし、現在の技術ではそのような機能を実装するのは難しいため、Wizard of Oz 法を用いて評価を行うことにした。

Wizard of Oz 法では wizard と呼ばれる人間のオペレータがシステムの一部分または全体をシミュレーションし、見かけ上はコンピュータによるシステムとして動作させ、被験者の行動を観測する。ここではオペレータがエージェントを操作するように Venus & Mars を改造した。オペレータには被験者とオペレータの発話内容

が見えており、それが返事できる内容のものであれば応答し、そうでなければ理解できなかったことを被験者に分からせるように発話する。オペレータが返事できる内容は、レシピ、健康、エージェントの個人的なこと、被験者の好み、挨拶などの一般的な事柄とした。

3.2 評価方法

評価は次の 2 種類の実験により得られた結果を比較することで行った。

- マルチエージェント

  前述のエージェント 3 体が登場する。最初に各エージェントは自分の役割を明らかにするためにそれぞれ自己紹介を行う。特にコンとチョーは、「献立探しの名人」、「食は健康の基。いつでも健康でいられるように、体にいいものを教えてしんぜようぞ。」という台詞で役割を示している。ここでは 3 人のオペレータがそれぞれの担当するエージェントを 2 章で述べたエージェントの役割と行動に従って操作する。
- シングルエージェント

  前述のエージェントの能力を全て持っているエージェント 1 体（見た目はコンのものである）が登場する。献立検索と健康についての知識があることを示すため、このエージェントの自己紹介では上記の台詞を両方とも用いている。また、この場合もマルチエージェントの場合と同様に 3 人のオペレータがエージェントを操作するが、レシピについての話題はコンのオペレータ、健康についての話題はチョーのオペレータ、それ以外の話題はペッコのオペレータが操作するようにした。

被験者には自然言語で会話のできるキャラクタを利用したレシピ検索システムであると説明し、どちらか片方のタイプを割り当てて実際に使ってもらう。被験者からエージェントへの発話回数が 20 回を越えるのを目安として 1 回の実験とした。また、オペレータがエージェントを操作していることは実験終了後に伝えている。

3.3 実験結果と考察

3.2 節で記した方法で、大阪市立大学の学生 10 人に実験を行った。実験では被験者のパソコン歴、インターネット歴、レシピへの関心、健康への関心をアンケート調査し、その傾向や性別が偏らないように振り分けた。マルチエージェントの被験者の 1 人を除き、全員 20 回以上の発話を行った。各被験者のはじめの 20 回の発話を整理した結果を表 1〜3 に示す。

表 1　被験者の発話の種類

	マルチ	シングル
質問	34(36.2%)	49(49.0%)
検索依頼	43(45.7%)	29(29.0%)
その他	17(18.1%)	22(22.0%)
合計	94(100.0%)	100(100.0%)

表 2　被験者の発話内容

	マルチ	シングル
エージェント	24(25.5%)	32(32.0%)
レシピ	40(42.6%)	33(33.0%)
健康	13(13.8%)	3(3.0%)
好み	3(3.2%)	3(3.0%)
その他	14(14.9%)	29(29.0%)
合計	94(100.0%)	100(100.0%)

表 1 は被験者の発話を種類によって「質問」、「検索依頼」、「その他」に分類し、表 2 は内容によって「エージェントについての個人的な話題」、「レシピについての話題」、「健康についての話題」、「被験者の好みについての話題」、「その他」に分類したものである。表の数値はそれぞれの分類について、被験者の発話回数とその割合を示している。表 3 はマルチエージェントの実験において、被験者が表 2 で分類した内容をどのエージェントを対象として発話したかを示している。表の数値は発話内容とその対象となるエージェントについて、被験者の発話回数とその割合を示している。

表 1 からマルチエージェントとシングルエージェントのどちらにおいても検索依頼以外の発話が多く発生しており、そのほとんどがエージェントについての質問であったことから、エージェントは被験者の気をひく存在であったと考えられる。また表 2 から、マルチエージェントではレシピについてや健康についての検索に関係す

表 3　マルチエージェントでの被験者の発話内容とその対象

	エージェント	レシピ	健康	好み	その他
ペッコ	8(33.3%)	9(22.5%)	3(23.1%)	0(0.0%)	11(78.6%)
コン	7(29.2%)	29(72.5%)	2(15.4%)	3(100.0%)	3(21.4%)
チョー	9(37.5%)	2(5.0%)	8(61.5%)	0(0.0%)	0(0.0%)
合計	24(100.0%)	40(100.0%)	13(100.0%)	3(100.0%)	14(100.0%)

る発話が多いのに対し、シングルエージェントではエージェントについてやその他の検索に関係しない発話が多い。このことよりシングルエージェントでは被験者を飽きさせやすかったと考えられ、マルチエージェントの方が検索に関係した発話を発生させやすいと言える。

また表 3 から、マルチエージェントにおいて被験者は、レシピについては献立検索エージェントに、健康については健康物知りエージェントに、その他の話題は個人エージェントに発話することが多く、エージェントを使い分けていたと考えられる。これらのことから、マルチエージェントにおいて被験者は各エージェントの役割を認識し、それによってマルチエージェントの方が検索に関係した話題を発話が多く発生したと考えられる。また、どちらの場合においても被験者の好みについての話題は少なかったが、マルチエージェントでは個人エージェントにその他の話題を発話することが多かった。これは個人エージェントは司会者のような役割として認識され、ユーザの好みを覚えておくという役割についてはあまり重要視されなかったか、上手く認識されなかったと考えられる。この点は、個人エージェントが被験者の好みを覚えていることをよりアピールすれば変化が見られるかもしれない。

4　まとめ

本研究では、情報検索システムにマルチキャラクタエージェントインターフェースを実装した Venus & Mars において、シングルエージェントとマルチエージェントの場合のユーザとエージェントのインタラクションを Wizard of Oz 法を用いて観測し比較した。その結果、マルチエージェントの方が検索に関係する内容の発話が発生しやすく、ユーザを飽きさせない効果があると考えられた。さらに、マルチエージェントにおいて被験者は各エージェントの役割を認識しており、エージェントを使い分けていることが分かった。これらの結果は、Venus & Mars のような分散した情報を統合して提供するシステムにとってマルチキャラクタエージェントインターフェースが有効であることを表している。しかし、エージェントの役割が意図したものと違って認識される可能性があることも示していた。このことから、エージェントを設計する際にはその役割をはっきりさせる必要があると言える。

現在は実験によって得られたシングルエージェントとマルチエージェントの場合での被験者のインタラクションの違いがどのような理由から起きたのか、また被験者がエージェントの役割を認識するまでにどのようなインタラクションを行っているかという視点から結果の解析を続けているところである。

参考文献

[1] Yasuhiko Kitamura, et al.: Interactice Integration of Information Agents of the Web, Matthias Klusch, Franco Zambonelli (Eds.), Cooperative Information Agents V, Lecture Notes in Artificial Intelligence 2182, Berlin et al.: Springer-Verlag, pp.1-13,(2001).

[2] Satoshi Oyama, et al: Keyword Spices: A New Method for Building Domain-Specific Web Search Engines, Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), pp.1457-1463,(2001).

[3] Yeonsoo Yang, Masayuki Okamoto, and Toru Ishida: Applying Wizard of Oz Method to Learning Interface Agent, Proceeding of Workshop on Software Agent and its Applications(SAA2000), pp.223-230,(2000).

[4] Maulsby, D., Greenberg, S. and Mander, R., Prototyping an Intelligent Agent through Wizard of Oz, Proc. of the conference on Human factors in computing systems, pp.277-284,(1993)

[5] Dahlback, N., Jonsson, A. and Ahrenberg, L., Wizard of Oz studies - Why and How, Proc. of the international workshop on Intelligent user interfaces, pp.193-200,(1993)

エージェント基盤のための通信プロトコル：Agent Platform Protocol

高橋 健一　鍾 国強　雨宮 聡史　峯 恒憲　雨宮 真人

近年，ネットワークで接続された複数の機器が協調や交渉を行い仕事を達成するシステムを作る枠組みとして様々なエージェントが提案されている．これらのエージェントシステムでは，エージェントの活動を促進する環境（AP）が個々に準備されているが，それら異なる AP 上のエージェントシステムを相互運用する手段については提供されていなかった．そこで，本稿では，AP 間のプロトコルとして Agent Platform Protocol(APP) を提案する．APP を用いることで，1）サーバを介さない P2P な AP 間通信と 2）物理的環境を意識する必要しないエージェントコミュニケーションを実現できる．

1　はじめに

インターネットの普及とともに，ネットワークで接続された複数の機器によって実現された多くの情報システムやサービスが提供されてきた．これらのシステムでは，分散された機器が互いの情報を交換することで協調や交渉を行い問題を解決する必要が出てきており，このようなシステムを作る枠組みとして様々なエージェントが提案されている．

エージェントシステムでは，エージェントの活動を促進するための環境（Agent Platform，以降 AP と記す）（例えば，KODAMA の ACZ [1] [2] や FRM の Field [3]，Hive の cell [4] など）が準備されている．エージェントは，AP 上で，仕事の達成に必要な情報を得るためのコミュニケーションや仕事の達成に必要な資源を得るための AP 間の移動を行う．これらの AP は，元々各エージェントシステム固有のものとして開発されているため，そのような異なる AP を持つエージェントシステムを相互運用することは難しい．エージェントの標準化促進団体 FIPA [5] では，Agent Communication Language（以降，FIPA ACL）や，Agent Platform などの仕様が定められているが，その FIPA ACL では，メッセージの送信側と受信側エージェントの物理的なアドレスを指定する，受信側エージェントの物理的位置を明確に意識した仕様となっている．しかし，これは，次のような弊害を引き起こす．

- モバイルエージェントとのコミュニケーションが困難．
  エージェントが移動すれば，エージェントが存在するアドレスも変わる．すなわち，受信側エージェントの物理アドレスを特定することは難しい．
- エージェントに，予め決定された連携を行わせることが難しい．
  例えば，2種類のエージェント，Ping と Pong が存在し，Ping は，適当なタイミングで Pong にメッセージを送り，Pong の存在を確認する仕事を持つとする．このとき，2種類のエージェント

Communication Protocol for Agent Platform : Agent Platform Protocol.

Kenichi TAKAHASHI, Zhong GUOQIANG, Satoshi AMAMIYA, 九州大学 大学院 システム情報科学府, Graduate School of Information Science and Electrical Kyushu University.

Tsunenori MINE, Makoto AMAMIYA, 九州大学 大学院 システム情報科学研究院, Graduate School of Information Science and Electrical Engineering, Kyushu University.

が別々にインストールされ，予めそれぞれのアドレスを特定できない場合，Ping は，Pong の物理的なアドレスを知ることができないため，送信先としては，”Pong”とだけしか指定できない．しかし，これでは，Pong が存在するアドレスが指定されていないため，Ping から Pong にメッセージを伝達するには，Pong の物理的位置を検索する機構が必要となる．

これらの問題を解決するためには，まず AP 間に共通のインタフェースを提供するためのプロトコルを定義しなければならない．そして，そのプロトコルでは，次のことを想定しなければならない．

- AP にエージェントの物理的アドレスを管理させ，エージェントに物理的な情報を意識させないようにする．
- 複数の種類の AP が存在しても，各 AP が一意にエージェントを識別できなければならない．
- AP によって，エージェントの管理方式が異なる．
- 要求した機能が，要求先 AP では実現されていない可能性がある．
- 一般に公開されていなければならない．

そこで我々は，AP 間でエージェントのコミュニケーションをサポートするためのプロトコルとして Agent Platform Protocol（以下，APP）を提案する．APP では，１３の method と header が定義されたリクエスト／リスポンス指向のプロトコルであり，AP はそれらに従ったメッセージを用いて他の AP と協調を行う．APP は，次の機能を提供する．

- サーバを介さない Peer to Peer（以降，P2P）な AP 間ネットワーク．
- エージェントメッセージの送受信機能．
- エージェントメッセージの伝達先の検索機能．
- エージェントの移動．

AP は，APP に従ってメッセージの送信先エージェントを検索し，メッセージの伝達を行うため，エージェントは，互いの物理的なアドレスを意識せず，コミュニケーションを行うことができる．エージェント開発者は，AP に APP を実装することで，他で開発された異種の AP との通信が行えるようになり，様々な場所で開発された様々な種類のエージェントを，互いに利用できるようになる．

以下，２節では APP について述べ，３節で APP を適用した AP の例として ACZ を紹介する．そして，４節で APP を適用した ACZ を用いて，APP の機能についての検証実験を行った結果について報告する．

2　Agent Platform Protocol（APP）

APP は，エージェントの活動をサポートする AP 間の通信プロトコルであり，図 1 のように位置づけられる．APP は，低レベルの通信プロトコルとして TCP [8] を持つ．TCP では，信頼性のあるコネクション指向型の通信を提供しており，データの欠損や重複，届けられる順序の違いなどを発生しない．

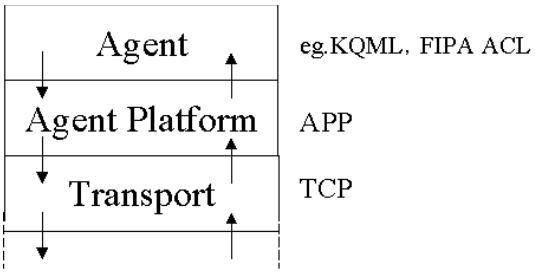


図 1　APP の位置づけ

APP の主な特徴として次の点が上げられる．

- サーバを介さない P2P な AP 間ネットワークの構築．

P2P なネットワークでは，各ネットワークの構成要素が置き換え可能なため，耐故障性が高く，一つの AP の故障が AP のネットワーク（以降，AP で構成されるネットワークを AP ネットワークと呼ぶ）に与える影響は少ない．また，サーバ／クライアントモデルで構成された AP ネットワークのように，サーバの役割を担う AP に負荷が集中するという問題はなく，更に，サーバの役割を担うような特定の AP への接続は不要で，AP ネットワークの任意な一つの AP への接続で AP ネットワークに加わることができる．

- エージェントメッセージの送受信機能の提供．

エージェントはコミュニケーション手段として，メッセージ（以降，APP によって用いられるメッセージ

表 1　APP で準備された method 一覧

分類	method	説明
Connect	LINK	AP 間の認証及び通信経路の作成
	DeLINK	AP 間の通信経路切断
	GET_LINK	要求先の AP が持つ通信経路情報の要求
	ALIVE	要求先が活動状態であるか否かの確認
Deliver	DELIVER	エージェントメッセージの伝達
	LOCATE_AGENT	エージェントの検索依頼
	GET_AGENT_LIST	要求先の AP が持つエージェント情報の要求
	PUT_AGENT_LIST	保持しているエージェントの情報を他の AP に通知
Mobile	GET_AGENT	エージェントを要求先の AP からローカルな AP に移動させる
	PUT_AGENT	ローカル AP 上のエージェントを要求先の AP に移動させる
Reply	ACCEPT	返答．要求を受理したときに用いる
	SUCCESS	返答．要求が成功したときに用いる
	FAILURE	返答．要求が失敗したときに用いる

と区別するために，エージェントメッセージと呼ぶ）を用いた情報交換を行う．

- エージェントメッセージの伝達先の検索機能の提供．

AP は，エージェントメッセージの送信先エージェントのアドレスが分からない場合，エージェントメッセージの伝達先を探さなければならない．エージェントメッセージの伝達先検索のため，APP では，（１）他の AP に検索を任せる方法と（２）他の AP から情報を集め自分自身で検索を進める方法とを提供する．

- モバイルエージェントの移動をサポート．

エージェントに，仕事の達成に必要な資源を求め AP 間を移動する手段を提供する．

2.1　AP メッセージ

APP で用いられるメッセージを AP メッセージと呼ぶ．AP メッセージは，method，header，body から構成され，この順に記述される．ここで，method は，AP メッセージが表す要求，または，応答を表す．header は，method を解釈，処理するのに必要な情報である．body は，各 method が対象とする任意のデータである．

AP メッセージ = {method, header, body}

2.1.1　method

AP は，エージェントのコミュニケーションや移動をサポートする環境を提供する．このため，APP では，１３の method を準備している（表 1）．

Connect，Deliver，Mobile に分類される method は，AP に対して要求を出し，その応答として，AP は Reply に分類される method を用いる．

Connect　サーバを介さないで P2P な AP ネットワークを構成するための method 群．LINK によって，AP 間での認証を行い，お互いの存在を知り，相互に信頼性のある通信を可能とする．GET_LINK は，要求先の AP が接続している AP の情報を要求する．AP は起動時に，まず LINK method を使って任意の AP に接続し，AP ネットワークに加わる．次に GET_LINK で AP ネットワークの情報を求める．新たに得た AP ネットワークの情報を利用して，複数の LINK を行うことで，特定の AP への依存度を下げることができる．DeLINK は，要求元の AP が通信を保証できなくなることを伝える．DeLINK を受け取った AP は，接続可能な他の AP との接続が分断されないように GET_LINK や LINK method を使って，ネットワークが分断されないように対処した後，DeLINK への応答を行わな

ければならない．ALIVE は，AP が実際に AP メッセージを受け付けることが可能であるかの確認を行う．

Deliver エージェントメッセージを送受信するための method 群．DELIVER は，エージェントメッセージを伝達する method では，body 部にエージェントメッセージを指定し，aimAt header(表 2 参照) でエージェントメッセージの受信側エージェントを指定する．DELIVER を受信した AP は，aimAt header が示すエージェントに body 部のエージェントメッセージを伝達する．受信側エージェントが存在する AP が分からない場合の検索手段として，LOCATE_AGENT と GET_AGENT_LIST を準備している．LOCATE_AGENT は，他の AP に検索を任せる方法であり（図 2(a)），LOCATE_AGENT を受信した AP は，aimAt で指定されたエージェントが存在する AP を探し出し，見つかれば，body 部をそのアドレス（IP アドレス/Port 番号）とした SUCCESS を返す．GET_AGENT _LIST は，要求先の AP に，それが持つエージェントの情報を要求する．これは，様々な AP からエージェントの情報を集め，その情報を用いて目的のエージェントが存在する AP を自分で見つける場合に利用する（図 2(b)）．また，PUT_AGENT_LIST は，保持しているエージェントの情報を他の AP に通知する場合に利用する．

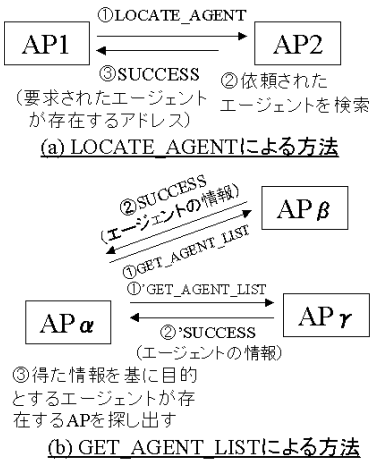


図 2　エージェントの検索方法

Mobile エージェントの移動を行うための method 群．POST_AGENT は，要求先の AP にエージェントを移動させ，GET_AGENT は，要求先の AP からエージェントを移動させる．

Reply Connect，Deliver，Mobile の各 method の応答として用いられる．ACCEPT は，各 method を受理したことを表し，FAILURE は，要求が失敗したことを，SUCCESS は，要求が成功したことを表す．

2.1.2　header

APP における header は，その役割によって，general-header，body-header，authentication-header に分けられる（表 2）．general-header は，すべての AP メッセージで必須であり，body-header は，body 部を持つ AP メッセージに必要とされる．authentication-header は，AP 間の認証が必要なときに用いられる．header は，

header = header-name ":" value

の形で表される．

2.1.3　body

body は，method で必要とされる任意のデータである．例えば，DELIVER の場合，body は，エージェントメッセージとなる．

3　APP の適用例

我々の研究室で研究・開発しているマルチエージェントシステム KODAMA（Kyushu university Open & Distributed Autonomous Multi-Agent）では，AP として ACZ （Agent Communication Zone）を利用する．本節では，まず，ACZ について述べ，次に ACZ への APP 適用例を示す．

3.1　ACZ

ACZ は，論理的なエージェント空間を物理的な構造にマッピングする機構であり，エージェント間のコミュニケーションを実現するために必要となる物理的通信機能を提供する．通常，一つの計算機上に一つの

表 2　header 一覧

	header-name	value	説明
general-header	message-id	string	メッセージつけられた固有の ID
	from	AP-Address (IP-Address/port)	メッセージを送信した AP のアドレス
	via	*AP-Address	メッセージの経路
	refer	*message-id	メッセージの動機となった message-id
	AP-Type	string	AP のタイプ
authentication -header	authentication-type	string	認証方法
	authentication-value	string	認証キー
	authentication-expire	number	authentication-value の有効期限
	absolute	0 or 1	リンクの重要性
body-header	body-size	number	body 部のサイズ
	body-type	string	body 部のデータ形式
	aimAt	string	body 部のデータの対象

* は，0 回以上の繰り返しを表す.

ACZ が存在し，分散環境上では複数の ACZ が存在する．これらの ACZ は，ネットワークを構成し，分散した計算機の間での通信を制御する．この管理のために ACZ は，ローカル層とリモート層より構成される（図 3）.

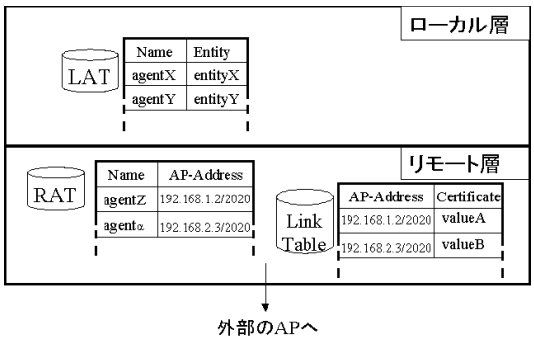


図 3　ACZ の構造

ローカル層は，ローカル ACZ 上のエージェントを管理するテーブル LAT（Local Agent address Table）を持ち，ローカル ACZ 上のエージェントへのエージェントメッセージの伝達を行う．ローカル ACZ 上のエージェントの生成または消滅時には，その名前が LAT に追加／削除される.

リモート層は，他の ACZ 上で活動するエージェントのアドレスを記したテーブル RAT（Remote Agent address Table）と，他の ACZ の認可情報を記したテーブル LT （Link Table）を持ち，他の ACZ 上で活動するエージェントへエージェントメッセージを伝達する際に利用される.

3.2　ACZ への APP の適用

ACZ のライフサイクルは，図 4 のように示される.

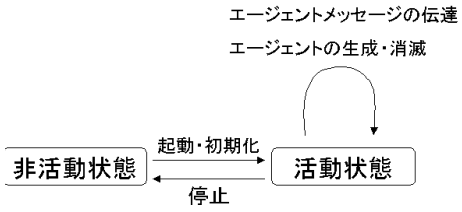


図 4　ACZ のライフサイクル

ACZ は生成されると直ぐに起動・初期化され，非活動状態から活動状態になる．活動状態中は，エージェントの生成や消滅，エージェントメッセージの伝達が行われる．そして，最後に停止され，活動状態から非活動状態に変化する．このように，ACZ の動作は，

起動・初期化，エージェントの生成・消滅，エージェントメッセージの伝達，停止の４つに分けられる．このうち，エージェントの生成・消滅はローカル層での動作であり，APP は関係しない．そこで，エージェントの生成・消滅以外の３つの ACZ の動作について APP がどのように適用されているかを示す．

3.2.1　ACZ の起動・初期化

KODAMA は，Java で開発されており，ACZ は次のコマンドで起動・初期化される．

java acz.ACZ (使用 port) (接続 IP:port)

図 5 のような状態において，

java acz.ACZ 2000 192.168.0.1:2020

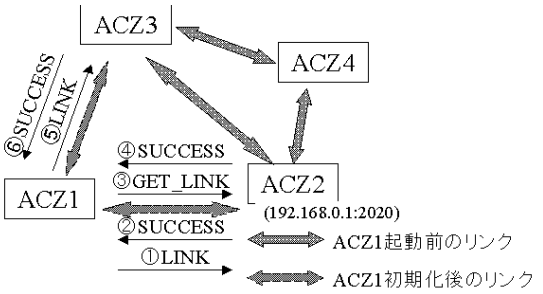


図 5　ACZ の初期動作

で ACZ1 を起動すれば，ACZ1 は，まず，LINK を使って，ACZ2 (192.168.0.1 /2020) との間で，相互通信のためのリンクを作る．これは，ACZ1 と ACZ2 が各自の LT に相互登録する（図 5 ①，②）ことによって実現する．次に，ACZ2 の持つ ACZ ネットワークの情報を集め（図 5 ③，④），集めた ACZ 中の任意の ACZ との間で相互の通信経路を作成する（図 5 ⑤，⑥）．図 5 では，ACZ3 とだけ通信経路を作成しているが，ACZ の設定[†1]によって，任意の数の通信経路を作成することができる．このことで，ACZ1 は特定の ACZ への依存度を減らすことができる．

3.2.2　ACZ の停止

図 6 のような状態において，ACZ1 が停止すれば，ACZ2 と ACZ3 の間での通信経路が無くなり，ネットワークが２つに分断される．このため，ACZ1 から DeLINK を受け取った ACZ2 は（図 6 ①），ACZ ネットワークの分断をさけるため，GET_LINK で停止する ACZ1 から ACZ ネットワークの情報を受け取り（図 6 ②，③），ACZ3 との間にリンクを構成する（図 6 ⑤）．

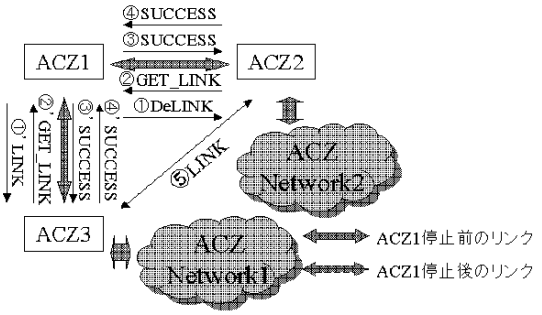


図 6　ACZ の停止動作

3.2.3　メッセージの伝達

エージェントメッセージの受信側エージェントが LAT のエージェントであれば，ローカル層でエージェントメッセージの伝達を行う．RAT に登録されているエージェントであれば，RAT からアドレスを求め，DELIVER を用い，エージェントメッセージを伝達する．LAT，RAT 双方に登録されていないエージェントであれば，そのエージェントの検索が行われる．

ACZ でのエージェントの検索は，KODAMA エージェントの論理的構造を参照し，LOCATE_AGENT を用いて行う．(図 7)．KODAMA では，エージェントは階層的に配置され，エージェントの名前が階層の場所を示す．例えば，”root・A”の名前を持つエージェントは，”root”の名前を持つエージェントを上位階層に持ち，”root・A・child”の名前を持つエージェントを下位階層に持つ．ACZ は，検索依頼を表す LOCATE_AGENT を用いて，次の方針でエージェントの検索を行う．

(a) 検索は RAT，LAT から refer header に含まれない ACZ で活動するエージェントで階層的に最も近いエージェントを見つけ，見つけたエージェントが目的のエージェントであれば，始めに LOCATE_ AGENT を出した ACZ に SUC-

†1　ユーザによって，ACZ 初期化時に作成される通信経路の枠組みが指定される．

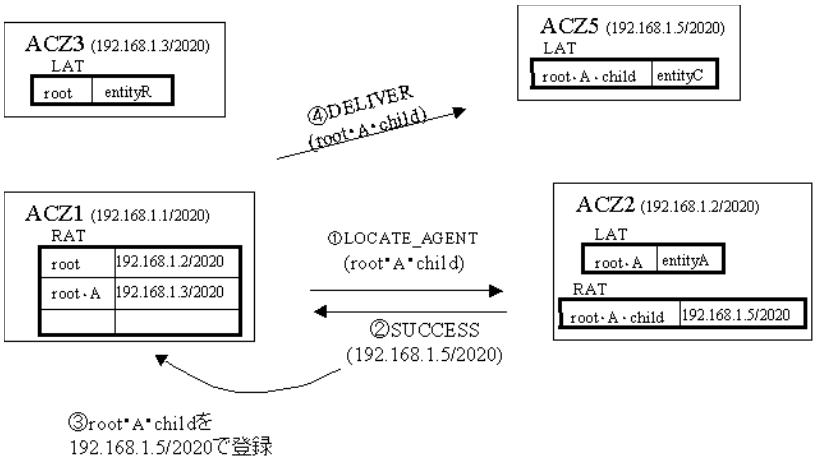


図 7　ACZ でのエージェント検索方法

CESS で応答し（図 7 ②），目的のエージェントでなければ，見つけたエージェントが活動する ACZ に LOCATE_ AGENT で検索を依頼する（図 7 ①）．LAT，RAT が共に要素を持たない場合は，最初に LT に登録された ACZ に LOCATE_ AGENT で検索依頼を出す．

(b) LOCATE_AGENT を受け取った AP は，via header を見て，すでに受け取っている LOCATE_ AGENT 要求であれば，始めに LOCATE_ AGENT を出した ACZ に FAILURE で応答する．まだ，受け取っていない LOCATE_ AGENT 要求であれば，(a) を行う．

(c) LOCATE_AGENT の応答として SUCCESS を受け取った ACZ は（図 7 ③），検索結果を RAT に登録する．応答として FAILURE を受け取れば，検索失敗である．

結果として，ACZ は，目的のエージェントを見つけた場合，DELIVER で（図 7 ④），エージェントメッセージを伝達する．

4　実験

APP の有効性を検証するために，APP を実装した ACZ で実験を行った．実験では，各計算機に一つの ACZ を準備し，合計 8 台の計算機を利用した．各 ACZ は，ランダムに停止・起動を繰り返すこととし，起動時には接続する ACZ を活動中の ACZ からランダムに選び，停止時には LAT，RAT，LT のテーブルを初期化した．また，各 ACZ には，１０のエージェントを準備し，それぞれに適当なエージェントとコミュニケーションを行わせることとした．この条件で 1 時間経過したあとにすべての ACZ を停止させた．このときの実験結果を表 3 に示す．

表 3　実験結果

総 AP メッセージ数	2831
総起動・停止回数	44
エージェントのコミュニケーション数	906

4.1　ネットワークに関する評価

実験では，ACZ の起動・停止が共に 44 回行われた．IP=192.168.35.76，port=2200 の ACZ 停止時に発生した AP メッセージを例 1 に示す．†2．

例 1 では，192.168.35.76 の停止によって AP ネットワークが分断されることを防ぐため，GET_LINK で AP ネットワークの情報を求め，192.168.35.64 との間に新しくリンクを作成した．192.168.35.73 との間には，すでにリンクを持っていたため，新しくリンクは構成しなかった．

結果，44 回の ACZ の起動・停止にも係わらず，AP

†2　すべての ACZ で 2200 番ポートを使用しているため，ポート番号は省略した．

1.DeLINK : From 192.168.35.76 to 192.168.35.65
2.ACCEPT : From 192.168.35.65 to 192.168.35.76 --(1の応答)
3.GET_LINK : From 192.168.35.65 to 192.168.35.76
4.SUCCESS : From 192.168.35.76 to 192.168.35.65,
[192.168.35.73, 192.168.35.64] --(3の応答)
5.SUCCESS : From 192.168.35.65 to 192.168.35.76 --(1の応答)
6.LINK : From 192.168.35.65 to 192.168.35.64
7.SUCCESS : From 192.168.35.64 to 192.168.35.65 --(6の応答)

例 1: ACZ 停止時の AP メッセージ

ネットワークから孤立した ACZ は，発生しなかった．すなわち，本稿で提案する APP を使用することで，サーバを介さない P2P な AP 間ネットワークを構成でき，AP ネットワーク接続の頑健性が確認できた．

4.2 エージェントのコミュニケーションに関する評価

実験では，906 回のエージェントのコミュニケーションが試みられた．このうち，903 回のコミュニケーションが正常に行われた．残り 3 回については，相手のエージェントが存在しないため，コミュニケーションを行わないものであった．この実験で，エージェントは，相手の物理的位置を意識することなく，コミュニケーションが正常に行えた．このように APP を使用することで，ACZ はエージェントの位置透過性を与えることができた．

5 おわりに

本稿では，エージェントの活動を促進する環境 AP を相互運用するためのプロトコルとして APP を提案した．APP は，サーバを介さない P2P な AP 間ネットワークで，エージェントが物理的環境を意識することなくコミュニケーションを行うための環境を与えることができる．

この検証のため，我々の研究室で開発している ACZ に，APP を実装した結果，APP を実装した ACZ は，動的な経路変更機能を持つ P2P な ACZ ネットワークを構成することができた．

今後の課題は，ACZ 以外の AP に APP を実装し，ACZ と相互運用が可能であるかを検証することと，APP で提供されるエージェント検索方法の効率を検証することである．

参考文献

[1] G. Zhong, K. Takahashi, T. Helmy, K. Takaki, T. Mine, S. Kusakabe and M. Amamiya, “KODAMA: As a Distributed Multi-agent System”, Proceedings of the 7th International Conference on Parallel and Distributed Systems: Workshops, IEEE Computer Society Press, pp.435–440, Iwate, Japan, July.2000.

[2] 高橋 健一, 高木 幸一郎, 鍾 国強, 雨宮 聡史, 峯 恒憲, 雨宮 真人,“分散マルチエージェントシステム KODAMA”, 「ソフトウェアエージェントとその応用」特集ワークショップ,SAA2000, 157–164, 2000

[3] T. Iwao, M. Takada, M. Amamiya, “Flexible Multi-Agent Collaboration using Pattern Directed Message Collaboration of Field Reactor Model”, Approaches to Intelligent Agents - Second Pacific Rim International Workshop on Multi-Agents, LNAI, vol.1733, Springer, pp.1–15, 1999.

[4] N. Minar, M. Gray, O. Roup, R. Krikorian and P. Maes, “Hive: Distributed Agents for Networking Things”, IEEE Concurrency, vol.8, no.2, 2000.

[5] FIPA : http://www.fipa.org

[6] T. Finin, Y. Labrou, and J. Mayfield, “KQML as an agent communication language”, In Software Agents, MIT Press, Cambridge, pp.291–316, 1997.

[7] E.D. Pietro, A.L. Corte, O. Tomarchio, A. Puliafito, “Extending the MASIF Location Service int the MAP Agent System”, IEEE Symposium on Computers and Communications, 2000.

[8] D. Comer, D. Stevens（村井 純, 楠本 博之（訳））, “TCP/IP によるネットワーク構築”, 共立出版, 2001.

投機的計算を使用したマルチエージェントシステムでの問題解決手法

山本 京司　佐藤 健

In this paper, we propose a method of problem solving in multi-agent systems when communication between agents is not guaranteed. To solve a problem under incomplete communication environments, a method using speculative computation is proposed by Satoh et al. However, this method applies only in a case that another agent's responses to the agent's question can't be revised in solving a problem. In this paper, we extend speculative computation proposed by Satoh et al. to a case that responses can be revised.

1　はじめに

現在のマルチエージェントの研究の多くでは, エージェントが他エージェントに質問を与えた場合, 質問を受けたエージェントから回答が来るまで, 質問をしたエージェントの処理は中断されるのが普通である. しかし, この "処理が中断される" ということは, 問題点となる. なぜなら, 実際には, 回答が来ない可能性という事も考えられるからである. 例えば, インターネットのように通信が必ずしも保証されていない環境では, 質問, 回答が途中で失われる事が有りうる. このような事態が生じた場合に, 処理が中断されるという事が大きな問題となるのは明らかである.

この様な問題を解決する方法として, 投機的計算を用いた手法が佐藤らによって提案されている[1] [2]. 佐藤らによって提案された手法は, あるエージェントから他エージェントに与えた質問の回答が翻らない場合において, 上記の問題を解決している. しかし, 各エージェントが投機的計算を行い, 他エージェントに対して質問, 回答を互いに送る様な場合には, 回答が何回も変化することがあり得る. この様な場合, 回答が翻らない場合に限られている佐藤らの手法では対応することが出来ない.

本論文は, 各エージェントが投機的計算を行なうようなマルチエージェントシステムによる問題解決の手法を提案することを目的としたものである. その実現の為, あるエージェントからの回答が翻るような場合にも投機的計算を適用する手法の提案, 及びここで提案した手法の計算機上への実装を行った.

本論文では, 説明のため以下のような例を用いる.

- ある保険の契約を行う問題を考える. 本社エージェント a, 営業エージェント b, 契約者 c がいる.
- a は, 契約者 c がオプションをつけるかを営業エージェント b に聞く.
- b は, 契約者 c に, オプションをつけるかどうかということを聞く.
- a は, c がオプションをつけないならばプラン 1 を, オプションをつけるのであればプラン 2 を用意する.
- b は, c からの回答を聞き, a に回答する.

この例題で, 以下の様な場合を考えてみる.

通常のマルチエージェントシステムの場合, エージェント a,b はそれぞれエージェント b 及び契約者

Keiji YAMAMOTO, 北海道大学大学院工学研究科, Graduate School of Engineering, Hokkaido University
Ken SATOH, 国立情報学研究所, National Institution of Information

c から回答が来なければ, 処理を進めることはできない. しかし, ここでエージェント a が “多くの契約者はオプションを付けない” ということを知っていれば, b からの回答を待たずに先行的に準備を進めることができる.

同様にエージェント b が “契約者 c は, 多くの場合オプションを付ける” ということを知っていれば, 事前にエージェント a に対してそのこと報告をすることができ, a はその報告を受けて準備の修正を行なう. もしその後, c から “オプションをつけない” という回答を b が受け取れば, b は a に対して先程の報告を訂正し, さらに a は再度修正を行なう.

この場合, エージェント a と b がそれぞれ投機的計算を行なっている. もし回答が変化する場合に対応できなければ, エージェント b が先行的に出した回答をエージェント a に送ることは危険性が高い. なぜならば, エージェント b が出した回答は, 投機的計算の性質上, 保証されていないからである.

回答が変化する場合に対応するための手法の概略は, 以下のようになる.

1. 各エージェントは事前に質問に対するデフォルトの仮回答を用意する.
2. 各エージェントは, 用意した仮回答をもとに以降の処理を進める.
   - もしデフォルトの仮回答を使って計算を進められるならば, そのまま続ける.
   - またはデフォルトの仮回答とは矛盾する回答を使わなければ計算を進められなければ, その矛盾する回答の集合を覚えておき, 計算は中断し, 他の計算プロセスの処理を始める. (ここで, この集合を処理中断集合と呼ぶ.)
3. 計算中に回答が返されたならば,
   (a) 現在計算中および中断されている全ての計算プロセスをチェックし, その中に回答と矛盾する仮定を使用しているものがあれば, その仮定を処理中断集合に移す.
   (b) 処理が中断されている全ての計算プロセスの処理中断集合をチェックし, その中に回答と一致するものが含まれていれば, それを回答待ち集合から削除する.
4. 処理中断集合が空集合となっているものを選択して, 計算を続ける.

4 では, 処理中断集合が空集合の計算プロセスを選択している. これは, 処理中断集合が空集合であるものは, デフォルトの仮回答または実際の回答と矛盾しておらず, 処理を進めても成功する可能性が高いと思われるからである.

本論文では, 他エージェントからの回答を待っているリテラルを集合として保存している. これにより, 他のエージェントからの回答が変化する場合に投機的計算の導入を行うことができる.

2　投機的計算導入の実現

以下では, 他エージェントからの回答が変化する場合に投機的計算を導入するための準備を行う.

2.1　枠組

ここでは, 投機的計算の枠組の定義を述べる.

定義 1

Q を原子式としたとき,Q を正リテラル,$\sim Q$ を負リテラルとする. ここで,$\sim$ は「失敗による否定」を表す.Q をリテラルとしたときに, $\sim\sim Q = Q$ と定義する.

定義 2

n 個のエージェントからなるマルチエージェントにおける投機的計算の枠組は, 2 つ組 $\langle \mathcal{I}, \mathcal{S} \rangle$ である. ここで, $\mathcal{I}$ は, エージェントの識別子の列 $\langle a_1, ..., a_n \rangle$ であり, $\mathcal{S}$ は, エージェントでの個別投機的計算の枠組の列 $\langle S_{a_1}, ..., S_{a_n} \rangle$ を表し以下のように定義される.

各エージェントでの個別投機的計算の枠組 S は, $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$ の 4 つ組であり, それぞれは以下の通りとする.

- Σ : 定数の有限集合.Σ の要素をエージェント識別子と呼ぶ.
- $\mathcal{E}$: 外部述語と呼ばれる述語の集合.Q が外部述語を持つリテラルで,S がエージェント識別子のとき,$Q@S$ を質問リテラルと呼ぶ. $\sim(Q@S)$ を $(\sim Q)@S$ と定義する.
- Δ : 以下の条件を満たす質問リテラルの基礎式全ての外部述語 p を用いた基礎原子式 $p(t_1, ..., t_n)$ と全てのエージェント S に対して,Δ には

$p(t_1,...,t_n)@S$ または $(\sim p(t_1,...,t_n)@S)$ のいずれかが含まれているものとする. Δ をデフォルト回答集合と呼ぶ.

- $\mathcal{P}$: 一般論理プログラム

質問リテラル $Q@S$ は, 以下の二つの意味を持っている.

1. $\mathcal{P}$ のルール中の質問リテラル $Q@S$ は,Σ 中のエージェント S に対して行なう質問を表す.
2. Δ における質問リテラルは, 外部エージェント S への質問に対するデフォルトの仮回答を表す. もし,$p(t_1,...,t_n)@S \in \Delta_{S'}$ ならば, 質問 $p(t_1,...,t_n)$ に対する S のデフォルトの仮回答は yes であり,$(\sim p(t_1,..,t_n))@S \in \Delta_{S'}$ ならば, 質問 $\sim p(t_1,...,t_n)$ に対する S のデフォルトの仮回答は yes である.

先程の保険契約の例題をこの枠組を使用して表すと, 以下のようになる.

枠組: $\langle \langle a, b \rangle, \langle S_a, S_b \rangle \rangle$

S_a

- $\Sigma_a = \{b\}$
- $\mathcal{E}_a = \{op_c\}$
- $\Delta_a = \{\sim op_c@b\}$
- $\mathcal{P}_a$:

  $prepare(P) \leftarrow plan(P).$

  $plan(1) \leftarrow \sim op_c@b.$

  $plan(2) \leftarrow op_c@b.$

S_b

- $\Sigma_b = \{c\}$
- $\mathcal{E}_b = \{op\}$
- $\Delta_b = \{op@c\}$
- $\mathcal{P}_b$:

  $op_c \leftarrow op@c.$

Δ_a は,“契約者はオプションをつけない” という仮定に, Δ_b は “契約者 c はオプションをつける” という仮定に相当する.

2.2 準備

投機的計算の実行は, 以下の二つの過程からなる. 一つはプロセス簡約過程であり, もう一つは回答到着過程である.

プロセスは,(a) その計算を開始, または継続するかを判定する為の集合,(b) 計算の現在の状態,(c) そのプロセスで既につかわれたデフォルト,(d) 最初に与えられた質問の変数に対する解代入,(e) 回答を返す相手, の 5 つの状態からなる.

一つの質問に対する証明を行うために, 一つのプロセスリストが作成される. プロセスは, 場合分けのような選択肢がある場合に分岐し, 各プロセスは, おのおの別な分岐計算を表す. プロセス簡約過程では, 計算を開始, または継続すると判定されたプロセスを新たなプロセスに変換する. プロセス簡約過程とは, 各エージェントにおける通常のプログラムの処理に相当するものである.

一方, 回答到着過程は, 他エージェントからの回答が来たときの割り込み処理に相当するものである. 本論文と [1] [2] との差異は, 回答到着過程の手法の違いである.[1] [2] では, エージェントからの返答を待っている中断リテラルが 1 個だけであるのに対して, 本論文では, 中断リテラルを集合としてプロセス中に保存するという形にしている. そして, 中断リテラルの集合が空集合ならば, そのプロセスは, 他エージェントから送られた実際の回答, もしくはまだ回答が返されていないならば, 事前に準備した仮回答と矛盾していないということになる. よって, そのプロセスを実行可能プロセスとし, また, そうでないときはプロセスを実行中断プロセスとしている.

定義 3

プロセスは 5 つ組 $\langle SGS, GS, AD, ANS, ID \rangle$ とする. ここで,

- SGS:他エージェントからの回答を待っている GS 内の質問リテラルの集合. 処理中断集合と呼ぶ.
- GS:拡張リテラルの集合. ゴール列と呼ぶ.
- AD:質問リテラルの集合. 仮定デフォルトと呼ぶ.
- ANS:初期ゴールにおける変数に対する解代入, または他エージェントから送られた質問リテラル.

- ID:送信エージェント識別子.

SGS とは, 真偽値の値がデフォルトまたは実際に他エージェントから来た回答の値と異なる質問リテラルを格納する集合となる. この集合が空集合ならば, そのプロセスは現時点では仮定したデフォルト, または実際に来た回答とも矛盾していない. そのため, そのプロセスは成功する可能性が高く, 効果があると考えられるのでそのプロセスの計算を開始, または継続する. 空集合でなければ, そのプロセスは計算を行なっても失敗する可能性が高いので, 中断して他のプロセスに実行を移す.

ANS は, 初期ゴールに変数が含まれている場合はその変数に対する代入が格納され, 他エージェントから質問を受け, その質問の証明を行う場合は, 質問リテラルが格納される.

ID は, **質問を送ったエージェントの識別子**である. エージェント S からエージェント S' への質問は $Q@S'\ from\ S$ という形で送られ, この場合,$ID = S$ となる. また, エージェントに対する質問がユーザーから与えられた場合は,$ID = u$ とする.

定義 4
プロセスにおいて, $SGS = \emptyset$ となるプロセスを実行可能プロセス, $SGS \neq \emptyset$ となるプロセスを実行中断プロセスとする.

計算状態を, 以下のリスト及び集合で定義する.
定義 5

- プロセスリスト PS とは, 実行可能プロセス, 実行中断プロセスの集合である.
- 既質問集合 AAQ とは, 他エージェントに送られた質問リテラルの集合である.
- 既回答集合 RF とは, 実際に他エージェントから返答された質問リテラルの集合である.

プロセスリスト PS は, 初期ゴールが与えられる度に新たに作成される. また, 一つのエージェント内で, ある初期ゴール GS から作成されたプロセスリストの処理は, 他の初期ゴール GS' から作成されたプロセスリストの処理とは独立に行なわれる.

既質問集合 AAQ 及び既回答集合 RF は, 各エージェントに対して各々ひとつずつ作成される. あるエージェントにおいて, 複数のプロセスリストが作成された場合でも, 各々のプロセスリストから同じ AAQ を参照することにより, 質問の重複を避けることが可能となる.

ゴール G の証明について, 以下のように定義する.
定義 6

- ゴール G を証明する為のプロセスリスト中で, あるプロセスが $SGS = \emptyset$ かつ $GS = \emptyset$ となった場合, ゴール G の証明に成功したとみなす.
- ゴール G を証明するためのプロセスリスト中の全てのプロセスが中断された場合, すなわち全プロセスで $SGS \neq \emptyset$ となった場合, このゴール G の証明に失敗したとみなし,$\sim G$ とする.

証明の失敗の定義は, ゴール G を証明するための証明木において, 全ての枝に, 事前に用意した仮定デフォルトと異なるリテラルが現れるため, 失敗する可能性が高くなる. それ故, ゴール G の証明も失敗する可能性が高くなる. このため, ゴール G の証明を失敗したとみなし,$\sim G$ としている.

2.3 プロセス簡約過程

以下の簡約過程では, エージェント S において, 各ステップで新しく書き換わった PS_S,AAQ_S,RF_S をそれぞれ $NewPS_S$,$NewAAQ_S$,$NewRF_S$ と書くことにする. もし, 書き換えが書かれてていない場合には, 変化しないことを表す. エージェント S に複数の初期ゴールが与えられた場合は, 以下の処理を各々の初期ゴールに対して独立に行なう.

初期ステップ: あるエージェント S で, 初期ゴール列 GS に対して実行可能プロセス $\langle \emptyset, GS, \emptyset, \emptyset, ID \rangle$ を証明手続きに与え,$AAQ_S = RF_S = \emptyset$ とし, 以下の**ステップ 1**, **ステップ 2**, **ステップ 3** を繰り返す.

ステップ 1: $\langle \emptyset, \emptyset, AD, ANS, ID \rangle$ というプロセスが

プロセスリスト PS_S 中に出現した場合,$ID = S'$ ならば ANS をエージェント S' に返し, $ID = u$ ならばユーザーに対し ANS を出力する. また, プロセスリスト中の全てのプロセスが実行中断プロセスとなった場合は,ANS の否定を S' に返す, または u に出力する.

ステップ 2: さもなければ, プロセスリストから実行可能プロセス $\langle \emptyset, GS, AD, ANS, ID \rangle$ を選び, さらに, GS から拡張リテラルを選び, $PS' = PS - \{\langle \emptyset, GS, AD, ANS, ID \rangle\}$ および $GS' = GS - \{L\}$ とする.

ステップ 3: 上で選ばれたリテラル L に対して以下を行なう.

- L が正リテラルならば,
  $NewPS = PS' \cup$
  $\{\langle \emptyset, (\{body(R)\} \cup GS')\theta, AD, ANS, ID\theta \rangle |$
  $R \in \mathcal{P}$ and $\exists$most general unifier(mgu) θ
  s.t. $head(R)\theta = L\theta\}$
- L が基礎負リテラルならば,
  $NewPS = PS' \cup \{\langle \emptyset, NewGS, AD, ANS, ID \rangle\}$
  ここで $NewGS =$
  $\{fail(body(R\theta)) | R \in \mathcal{P}$ and $\exists$mgu θ
  s.t. $head(R)\theta = L\theta\} \cup GS'$
- L が $fail(BS)$ という式ならば,
  - もし, $BS = \emptyset$ ならば $NewPS = PS'$.
  - もし, $BS \neq \emptyset$ ならば BS から B を選び,
    $BS' = BS - \{B\}$ とし,
    $*$ B が正リテラルならば,
    $NewPS = PS' \cup$
    $\{\langle \emptyset, NewGS \cup GS', AD, ANS, ID \rangle\}$
    ここで $NewGS =$
    $\{fail((\{body(R)\} \cup BS')\theta) |$
    $R \in \mathcal{P}$ and $\exists$mgu θ s.t. $head(R)\theta = B\theta\}$
    $*$ B が基礎負リテラルまたは,
    基礎質問リテラルならば,
    $NewPS = PS' \cup$
    $\{\langle \emptyset, \{\sim B\} \cup GS', AD, ANS, ID \rangle\} \cup$
    $\{\langle \emptyset, \{fail(BS')\} \cup GS', AD, ANS, ID \rangle\}$
- L が基礎質問リテラル $Q@S'$ のとき
  - $L \notin AAQ$ かつ $\sim L \notin AAQ$ ならば,
    エージェント S' に質問 $Q@S'$ $from$ S を送り,
    $NewAAQ_S = AAQ_S \cup \{L\}$ とする.
  - $L \in RF_S$ ならば,
    $NewPS =$
    $PS' \cup \{\langle \emptyset, GS', AD \cup \{L\}, ANS, ID \rangle\}$ とする.
  - $\sim L \in RF_S$ ならば,
    $NewPS =$
    $PS' \cup \{\langle \{L\}, GS', AD \cup \{L\}, ANS, ID \rangle\}$ とする.
  - さもなければ $L \in \Delta$ ならば,
    $NewPS =$
    $PS' \cup \{\langle \emptyset, GS', AD \cup \{L\}, ANS, ID \rangle\}$
    とする.
  - さもなければ $\sim L \in \Delta$ ならば,
    $NewPS = PS' \cup \{\langle \{L\}, GS', AD, ANS, ID \rangle\}$
    とする.

2.4 回答到着過程

ここでは, 回答が到着した場合についての処理手法を説明する. あるエージェント S' からエージェント S に対して回答 Q が返されたとき, S は以下の処理を行う. 以下の処理過程では, 新しく書き換わった SGS_S,AD_S,RF_S をそれぞれ $NewSGS_S$,$NewAD_S$,$NewRF_S$ とする. もし, 書き換えが書かれていなければ, 変化しないことを表す.

- プロセスの処理
  PS 中の全てのプロセスに対して以下を行う.
  - $Q@S' \in SGS_S$ ならば,
    $NewSGS_S = SGS_S - \{Q@S'\}$ かつ
    $NewAD_S = AD_S \cup \{Q@S'\}$
  - $\sim Q@S' \in AD_S$ ならば,
    $NewSGS_S = SGS_S \cup \{\sim Q@S'\}$ かつ
    $NewAD_S = AD_S - \{\sim Q@S'\}$

  上記の処理後, $SGS_S = \emptyset$ となっているプロセス (実行可能プロセス) を選択し計算を進める. そう

でないプロセス (実行中断プロセス) は計算を中断する.

- RF_S の処理
  - $Q@S' \notin RF_S$ かつ $\sim Q@S' \notin RF_S$ ならば, $NewRF_S = RF_S \cup \{Q@S'\}$
  - $Q@S' \in RF_S$ ならば, $NewRF_S = RF_S$
  - $\sim Q@S' \in RF_S$ ならば, $NewRF_S = RF_S \cup \{Q@S'\} - \{\sim Q@S'\}$

  この処理は, 各エージェントからの最新の回答だけを蓄えておくということを表す.

3　計算機上への実装

提案した手法の計算機上への実装を行なった. システムの構成は図 1 のようになる.

投機的計算部分に prolog を, 通信部分に Java を用いて実装を行なった. 初期状態では,Flag File に “false” と書き込み, Answer File には何も書き込まないでおく.

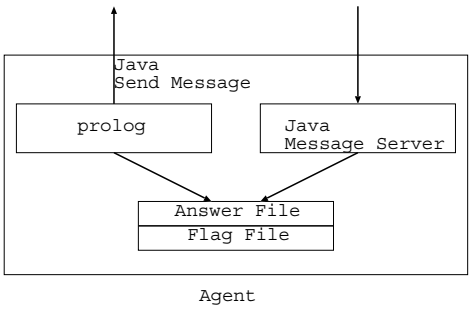


図 1　システム構成

prolog では, 投機的計算を進める過程において, 質問リテラルが出現した場合, またはゴールの証明に成功もしくは失敗した場合, 質問リテラルまたは回答を対象となるエージェントの Message Server に送信する.

一方,Java で作成した Message Server では, 他エージェントからのメッセージを受け取った場合,Flag File を “true” と書き換え, 実際のメッセージを Answer File に書き込む.

prolog では,1 ステップ毎に Flag File のチェックを行なう. この時,flag が “true” となっていれば, 他エージェントからのメッセージが来ているので, Answer File を読み込み, 投機的計算に他エージェントのメッセージを反映させる. その後,Flag File を “false” と書き換える.

保険契約の例題における全体の構成は, 図 2 の通りとなる. なお,c は投機的計算を行なうエージェントではなく,c からエージェント b への回答はユーザーが行なう.

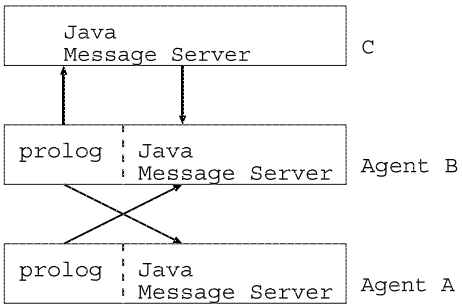


図 2　全体図

4　実行例

実行例として, 先に述べた保険契約の問題の実行履歴を示す.

以下で, エージェント a に $prepare(P)$ を与えた場合の実行履歴を示す.

最初に, エージェント a の履歴を示す.

1. $PS_a = \{\langle \emptyset, \{prepare(P)\}, \emptyset, \emptyset, \emptyset \rangle\}$, $AAQ = \emptyset$, $RF = \emptyset$
2. $PS_a = \{$ $\langle \emptyset, \{plan(1)\}, \emptyset, \{P = 1\}, \emptyset \rangle$, $\langle \emptyset, \{plan(2)\}, \emptyset, \{P = 2\}, \emptyset \rangle\}$
3. $PS_a = \{$ $\langle \emptyset, \{\sim op_c@b\}, \emptyset, \{P = 1\}, \emptyset \rangle$, $\langle \emptyset, \{plan(2)\}, \emptyset, \{P = 2\}, \emptyset \rangle\}$
4. $PS_a = \{\langle \emptyset, \emptyset, \{\sim op_c@b\}, \{P = 1\}, \emptyset \rangle$, $\langle \emptyset, \{plan(2)\}, \emptyset, \{P = 2\}, \emptyset \rangle\}$ $AAQ_a = \{\sim op_c@b\}$ エージェント a からエージェント b に $\sim op_c@b$ $from$ a という質問を送る

ここで, a は b から回答が来る前に, デフォルト $\sim op_c@b$ を使用して $P=1$ という解を得た.

つぎに, エージェント a から $\sim op_c@b\ from\ a$ という質問を受け取ったエージェント b の履歴を示す.

1. $PS_b = \{$
   $\langle \emptyset, \{\sim op_c\}, \emptyset, \{\sim op_c@b\}, a\rangle\}$,
   $AAQ_b = \emptyset, RF_b = \emptyset$
2. $PS_b = \{$
   $\langle \emptyset, \{fail(op@c\}, \emptyset, \{\sim op_c@b\}, a\rangle\}$
3. $PS_b = \{$
   $\langle \emptyset, \{\sim op@c\}, \emptyset, \{\sim op_c@b\}, a\rangle\}$
4. $PS_b = \{$
   $\langle \{\sim op@c\}, \emptyset, \emptyset, \{\sim op_c@b\}, a\rangle\}$
   $AAQ_b = \{\sim op@c\}$
   エージェント b から c に対して,$\sim op@c\ from\ b$ という質問を送る

   ここで, $\sim op@c$ は b のデフォルトと一致しないため, 処理中断集合に入り, この計算プロセスは中断される. また,$\sim op_c$ を証明するための計算プロセスが全て中断されたため, このゴールの証明に失敗したものとみなし, $\sim(\sim op_c)$, すなわち $op_c@b$ がエージェント a に対して回答される.
   続いて, このエージェント b からの回答を受け取ったエージェント a の履歴を示す.
   (a) $PS_a = \{\langle \{\sim op_c@b\}, \emptyset, \emptyset, \{P=1\}, \emptyset\rangle,$
   $\langle \emptyset, \{plan(2)\}, \emptyset, \{P=2\}, \emptyset\rangle\}$
   $RF_a = \{op_c@b\}$
   (b) $PS_a = \{$
   $\langle \emptyset, \{plan(2)\}, \emptyset, \{P=2\}, \emptyset\rangle\},$
   $\langle \{\sim op_c@b\}, \emptyset, \emptyset, \{P=1\}, \emptyset\rangle\}$
   (c) $PS_a = \{$
   $\langle \emptyset, \{op_c@b\}, \emptyset, \{P=2\}, \emptyset\rangle\},$
   $\langle \{\sim op_c@b\}, \emptyset, \emptyset, \{P=1\}, \emptyset\rangle\}$
   (d) $PS_a = \{$
   $\langle \emptyset, \emptyset, \{op_c@b\}, \{P=2\}, \emptyset\rangle\},$
   $\langle \{\sim op_c@b\}, \emptyset, \emptyset, \{P=1\}, \emptyset\rangle\}$
   履歴 (a) において, b から回答 $op_c@b$ を受け取ったことによって, それまで計算を行っていたプロセス $\langle \emptyset, \emptyset, \{\sim op_c@b\}, \{P=1\}, \emptyset\rangle$ 中の $\sim op_c@b$ が処理中断集合に移される. **このプロセスは処理中断集合が空集合でなくなるので, 実行中断プロセスとなり, 別のプロセスの計算が行なわれる.**

   a が出力する解は $P=2$ となる.

ここで,c からエージェント b に対して $\sim op@c$ という回答がきたとする. この時, エージェント b では以下のようになる.

1. $PS_b = \{$
   $\langle \emptyset, \emptyset, \{\sim op@c\}, \{\sim op_c@b\}, a\rangle\}$
   $RF_b = \{\sim op@c\}$
   先程まで SGS に入っていた $\sim op@c$ が AD に移された. これにより $SGS = \emptyset$ かつ $GS = \emptyset$ となるプロセスが出現し, $\sim op_c@b$ の証明が成功したとみなされ, エージェント a に回答される.
   エージェント a は, 先程とは翻った回答をエージェント b から受けることになる. この時のエージェント a は以下のようになる.
   (a) $PS_a = \{$
   $\langle \emptyset, \emptyset, \{\sim op_c@b\}, \{P=1\}, \emptyset\rangle\},$
   $\langle \{op_c@b\}, \emptyset, \emptyset, \{P=2\}, \emptyset\rangle\}$
   $RF_a = \{\sim op_c@b\}$

エージェント a から, 再び解 $P=1$ が得られる. この実行履歴で示した通り, エージェント a は翻った回答をエージェント b から受け取ることになるが, 最新の回答に基づいた解が得られる. また, この履歴では示していないが, エージェント b が c から翻った回答を受け取った場合でも, エージェント a の場合と同様にして最新の回答に基づいた解が得られる.

5 おわりに

本論文における成果は, 投機的計算の手法を拡張し, 他のエージェントからの回答が変化するという場合に対処することが出来る手法の提案を行なった事, またその手法を計算機上に実装し, 動作の確認を行なったことである. 今後の課題としては, 応用問題による効果の検証が挙げられる.

謝辞

本研究を進めるにあたり, 北海道大学の原口 誠先生, 大久保 好章先生, 神戸大学の井上 克己先生, 和歌山大学の坂間 千秋先生, 山梨大学の岩沼 宏治先生に貴重なご助言, ご指摘をいただきました. ここに謝意を示します.

参考文献

[1] Satoh,K., Inoue,K., Iwanuma,K., Sakama,C., "Speculative Computation by Abduction under Incomplete Communication Environments", *Proceedings of the Fourth International Conference on MultiAgent Systems*,to appear(2000)

[2] 佐藤, 井上, 岩沼, 坂間, 不完全通信環境下におけるアブダクションによる投機的計算, 人工知能学会知識ベース研究会資料, SIG-KBS-9904, pp.43 - 48 (2000)

[3] 山本, 佐藤, マルチエージェントシステムへの投機的計算導入手法, 第 15 回人工知能学会全国大会 (2001)