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.

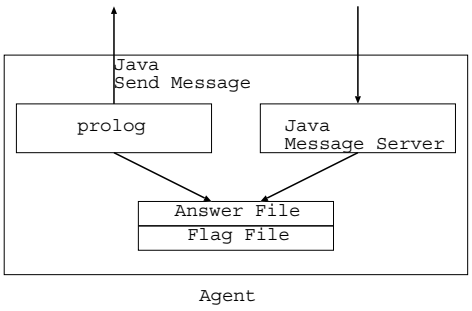

図 1　システム構成

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.

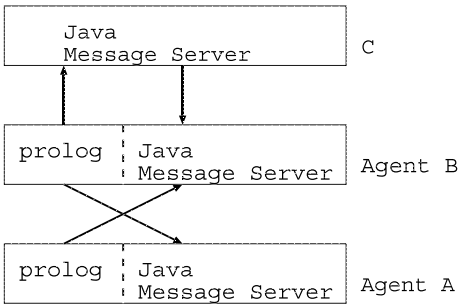

図 2　全体図

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)