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

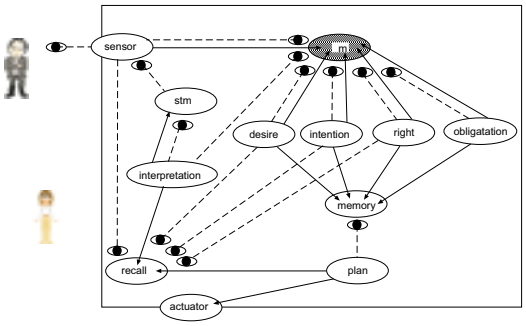

図 1　アーキテクチャ

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.