ソフトウェア基礎論　37－20
プログラミング言語　27－20
（1990. 12. 14）

AOPL: エージェント指向プログラミング言語*

赤埴 淳一
NTT 情報通信処理研究所

あらまし　互いに通信しながら協調動作するようにエージェントをプログラムするためのプログラミング言語 AOPL を提案する．AOPL の特徴は以下の 3 点にある．1) エージェントは自分自身の責務 (Commitment) を果たすために，状況に応じて行動する，というエージェントの動作モデルに基づいている．すなわち，エージェントのプログラムは，ある状況において責務を果たすために実行すべき動作を示すものである．2) プログラムは，知識や責務と同様に，エージェントの心的概念として表現され，AOPL は心的概念に対応する様相性をもった時相論理として定義される．3) 発話行為理論に基づいた通信動作のプリミティブを用意している．本稿ではさらに，AOPL のインタープリタと，多エージェント系のシミュレータについて述べる．

AOPL: An Agent-Oriented Programming Language

Jun-ichi Akahani
NTT Communications and Information Processing Laboratories
1-2356 Take, #609C, Yokosuka, Kanagawa 238-03, Japan
akahani@nttkb.ntt.jp

Abstract　This paper presents a programming language, called AOPL, in which we can program agents so that they coordinate with communication. AOPL has the following three main features. 1) The language adopts a version of agent rationality: an agent takes actions adaptively to fulfill its commitments. In other words, a program for an agent specifies actions that the agent should take in a certain situation to fulfill its commitments. 2) Program is regarded as a mentalistic notion such as knowledge and commitment, so AOPL is defined as a temporal logic with modalities corresponding to intentional notions. 3) Communicative action primitives based on speech acts are provided.

We also describe an interpreter for the language and a simulator for multi-agent systems.

*本研究の主要な部分は，著者がスタンフォード大学滞在中に行なわれたものである．

1 はじめに

本稿では, 互いに通信しながら協調動作するようにエージェントをプログラムするためのプログラミング言語を提案する. 本プログラミング言語 AOPL は, 次の3つのアイデアに基づいている.

1. エージェントの動作モデルとして, エージェントは自分自身の責務 (Commitment) を果たすために, 状況に応じて行動する, というモデルを採用する.
2. プログラムを, 知識や責務と同様に, エージェントの心的概念として表現する.
3. 発話行為理論に基づいた通信動作のプリミティブを用意する.

上記のアイデアに基づいたプログラミング言語を実現するためには, 知識や決意といったエージェントの心的状態を表現するための言語が必要となる. われわれは, このような言語としてエージェント記述言語 ADL を用いる. エージェント記述言語は知識や信念, 責務, 能力など心的概念に対応する様相性をもった時相論理である. われわれがこの言語を採用するのは, この言語が明快な意味論をもっているからだけではなく, エージェントの心的状態を世界に関する事実と同様に記述可能な統一的な枠組を与えているからである. 後者により, 通知, 要求, 約束などの通信動作をエージェントの心的状態上の動作として定義することができる. また, エージェント記述言語は時相論理に基づいているので, 動作の遅延を取り扱うことが可能である. しかし, エージェント記述言語は動作を記述する枠組を与えていない. したがって本稿では, 動作を, 前件と後件に対応する2つの論理式が付随した論理式として導入する.

上述のように, 我々はあるエージェントに対するプログラムを, そのエージェントがある状況において自分自身の責務を果たすために実行すべき動作を示した心的概念であると考えている. この考えに基づいて, 我々はプログラムを動作記述で拡張されたエージェント記述言語の論理式として定義する. すなわち, プログラムは, エージェントがある責務をもっていて, さらにある条件が成立したときに, ある動作を選択することを表した論理式で定義される. 結果として, プログラミング言語はエージェント記述言語を動作とプログラムで拡張したものになる.

本プログラミング言語は, 時間を陽に記述できる枠組を与えているので, 与えられた時間にエージェントが実行すべき動作を決定する, プログラミング言語のインタープリタを構築することができる. プログラミング言語 AOPL は, 時間記述を含む様相論理に基づいているので, そのインタープリタは, その論理の定理証明器になる. しかし, 一般的な定理証明は, インタープリタを実行不可能にしてしまう. そこで, 我々は, インタープリタを実行可能とするために, AOPL の論理式をあるクラスに制限する. 直観的には, 制限されたプログラミング言語は, ホーン節を拡張したものになっている.

本稿の構成は, 以下の通りである. 2章で, プログラミング言語に対する要求条件を挙げ, その条件を満足する言語の設計方針を述べる. 3章で, エージェント記述言語 ADL の概要について述べ, 4章で動作の概念を ADL に導入する. 5章で, プログラミング言語 AOPL を定義する. 6章で AOPL のインタープリタを構築し, インタープリタを実行可能にするための制限について述べる. 7章では, この言語の有用性を例を用いて説明する. 8章で, 今後の方向性について議論し, まとめを行なう.

2 プログラミング言語に対する要求条件

本章では, エージェント指向プログラミング言語に要求される条件を挙げ, その要求条件を満足する言語の設計方針について述べる.

要求条件 1. 責務駆動 (commitment driven) であること.

エージェントのプログラムは, そのエージェントの行動を決定するものである. プログラム言語を設計するためには, エージェントが実行すべき動作をどのようにして決定するかという理性的なエージェントのモデルが必要になる. エージェントのモデルとして, 刺激 - 反応モデル [Agre and Chapman, 1987] が提案されている. しかし, このモデルはメッセージ駆動であり, 時間が指定された動作を扱うことができない. 例えば, "午後3時に会議室へ行く" という動作を刺激 - 反応モデルで扱うためには, "午後3時" に外部 (例えば, クロック) からの刺激が必要になる.

そこで我々は, エージェントのモデルとして, エージェントは自分自身の責務を果たすために, 状況に応じて実行すべき動作を決定する, という責務駆動モデルを採用する. ここでいう責務とは, 持続性をもった目標

[Cohen and Levesque, 1990] を責務の対象エージェントを用いて拡張したものである．例えば，午後2時にエージェント a がエージェント b に対して，“午後3時に書類を会議室に配達する”ように要求したとする．このとき，エージェント b は，エージェント a に対して，“午後3時に書類を会議室に配達する”という目標を，達成するまでもち続けるという責務をもつ．さらに，エージェント b は，その責務を果たすために，状況に応じて動作を決定する．例えば，b が自分で3時に配達できると信じている場合には，“午後3時に会議室に行く”という動作を選択し，実現不可能な場合は，“他のエージェントに依頼する”という動作を選択したり，“エージェント a に実現不可能だと通知する”という動作を選択する．このモデルは，信念・欲求・意図 (BDI) モデル [Bratman *et al.*, 1988] に類似したものである．

要求条件 2. 明快な意味論をもつこと．

エージェントのプログラムを記述するためには，知識や信念といったエージェントの心的状態を表現する言語が必要になる．例えば，書類を会議室に配達するようにエージェントをプログラムするためには，会議室に関する知識 (例えば，会議室の位置) を記述しなければならない．プログラミング言語が明快な意味論をもつためには，エージェントの心的状態を表現する言語が明快な意味論をもつ必要がある．そこで，我々は，エージェントの心的状態を表現するための言語として，エージェント記述言語 ADL [Thomas *et al.*] を採用する．エージェント記述言語 ADL は知識や信念，責務，能力など心的概念に対応する様相性をもった時相論理であり，その意味論は Kripke 構造 [Kripke, 1971] を拡張した構造を用いて与えられる．

さらに我々は，プログラムを知識や責務と同様にエージェントの心的概念と考え，プログラムに対応する様相演算子を用いて拡張されたエージェント記述言語の論理式で，プログラムを表現する．すなわち，エージェントがあるプログラムにしたがっていることは，対応する論理式をそのエージェントがプログラムとしてもっているという心的状態で表現される．前節で述べたように，エージェントはその責務を果たすために状況に応じて実行すべき動作を決定する．したがって，エージェントがプログラムとしてもつ論理式は，責務を表わす論理式と状況を表わす論理式との連言から動作の選択を表わす論理式への含意となる．

要求条件 3. プログラムの検証が容易であること．

エージェント指向プログラミング言語において，プログラムの仕様はエージェントのもちうる責務で表現される．したがって，プログラムの検証は，プログラムが示す動作によって責務が果たされるか否かを調べることで行うことができる．厳密には，エージェントが知らないうちに世界が変化することはないという仮説 (これを静世界仮説と呼ぶ) のもとで，ある責務を果たすためにプログラムが示す動作を実行したとき，その責務が満足されるかあるいは責務の対象エージェントによって取り消されるならば，そのプログラムは仕様にあっていると定義される．

要求条件 4. 通信動作のプリミティブをもつこと．

エージェントが通信しながら協調動作するようにプログラムするためには，全エージェントに共通な通信動作のプリミティブを用意する必要がある．我々は，発話行為理論 [Searle, 1969] にしたがって通信動作を分類し，エージェントの心的状態上の動作として定義する [Cohen and Perrault, 1979]．例えば，エージェント a がエージェント b に論理式 φ を時刻 t に通知するという動作は，時刻 t に a が φ を知っていることを前件にもち，「『時刻 t に a が φ を知っている』ということを時刻 $t+1$ に b が知っている」ということを時刻 $t+1$ に a が知っているということを後件にもつ動作として定義される．後述の知識の性質を用いて，この後件から，時刻 $t+1$ に b が φ を知っていることと，同じく時刻 $t+1$ に a が“$t+1$ に b が φ を知っている”ことを知っていることが導かれる．

3 エージェント記述言語

本章では，エージェント記述言語の概要を述べる．この言語の詳細は，[Thomas *et al.*] を参照されたい．我々は，まず知識と責務に対応する様相演算子をもつ時相論理の統語論と意味論を与え，次に，信念，選択，能力を表す様相演算子を，知識と責務を用いて定義する．

3.1 知識と責務に対応する様相性をもつ時相論理

本節では，あるエージェントがある時刻においてもっている知識および責務を表現するための言語を定義する．直観的

には, $[t, K(a,\varphi)]$ で時刻 t においてエージェント a が φ という知識をもっていることを示す. 例えば, $[10:00, K(Carol, [9:00, FullFlt(354)])]$ は, 時刻 9 : 00 に 354 便が満席 ($FullFlt$) だということをエージェント $Carol$ が時刻 10 : 00 に知っていることを表す. 同様に, $[t, CMT(a,b,\varphi)]$ で時刻 t においてエージェント a がエージェント b に対して φ を実現するという責務をもっていることを示す. 例えば, $[10:00, CMT(Carol, Pat, [10:15, HasRsrv(Pat, 354)])]$ は, 10:15 に, エージェント Pat が 354 便の予約をもつ ($HasRsrv$) ことを実現するという責務を, Carol が Pat に対して 10:00 においてもっていることを表す.

知識と責務に対応する様相演算子をもつ時相論理の統語論は, 次で定義される.

以下の集合を仮定する. TC: 時間定数の集合, TV: 時間変数の集合, AC: エージェント定数の集合, AV: エージェント変数の集合, P: (0 個以上の引数をもつ) 述語記号の集合, F: (0 個以上の引数をもつ) 関数記号の集合, V: 変数の集合. このとき, 項は次のように定義できる. 変数は項であり, $n \geq 0$ のとき, n- 引数の関数記号と n 個の項との組は項である. さらに, 整式は, 次で定義される.

1. $t_1 \in TC \cup TV$, $t_2 \in TC \cup TV$ に対して $t_1 < t_2$ は整式である.
2. $t \in TC \cup TV$, p を n 個の項を引数にとる n- 引数の関数記号とするとき, $[t,p]$ は整式である.
3. $t \in TC \cup TV$, $a \in AC \cup AV$, $b \in AC \cup AV$, φ を整式とするとき, $[t, K(a,\varphi)]$ と $[t, CMT(a,b,\varphi)]$ は整式である.
4. φ および ψ を整式とするとき, $\varphi \wedge \psi$ と $\neg\varphi$ は整式である.
5. $t \in TV$, $a \in AV$, $v \in V$, φ を整式とするとき, $\forall t.\varphi$, $\forall a.\varphi$, $\forall v.\varphi$ は整式である.

通常の論理の構築と同様に, $\vee$, $\supset$, $\exists$ を通常の通りに導入する. $[t,p] \wedge [t,q]$ を $[t, p \wedge q]$ と省略することがある. また, $[t, \neg p]$ を $\neg[t,p]$ と書くこともある. 同様の省略を $\vee$ と $\supset$ に対しても行なう. 閉整式を文と呼ぶ.

整式の意味論は, Kripke 構造 [Kripke, 1971] の拡張である K-CMT 構造を用いて定義される. まず, 時間構造を導入する. 時間構造は, 時点の集合 T と T 上の全順序 $\prec$ との組 $\langle T, \prec \rangle$ で定義される. 時間線の集合 L と時間定数と時間変数に対する解釈関数 $\tau : TC \cup TV \to T$ を仮定することで, 時間線 $l \in L$ と意味関数 $m : TC \cup TV \to 2^P$ を同一視することができる.

K-CMT 構造は, L を時間線の集合, r_K を時点 t とエージェント a から L 上の到達可能関係 $R_{k_{t,a}}$ への写像, r_{CMT} を時点 t とエージェントの組 (a,b) から L 上の到達可能関係 $R_{CMT_{t,a}}$ への写像としたとき, $\langle L, r_K, r_{CMT} \rangle$ で定義される. ある時間線に関するある整式の充足可能性は次のように定義される.

1. $l \models t_1 < t_2$ iff $\tau(t_1) \prec \tau(t_2)$.
2. $l \models [t,p]$, ただし $p \in P$, iff $p \in m(t)$.
3. $l \models \neg\varphi$ iff $l \not\models \varphi$.
4. $l \models \varphi \wedge \psi$ iff $l \models \varphi$ かつ $l \models \psi$.
5. $l \models \forall t.\varphi$, ただし $t \in TV$, iff 全ての $t' \in TC$ に対して $l \models \varphi(t/t')$, ここで $\varphi(t/t')$ は φ の t を t' で置き換えたものである.
6. $l \models \forall a.\varphi$, ただし $a \in AV$, iff 全ての $a' \in AC$ に対して $l \models \varphi(a/a')$.
7. $l \models [t, K(a,\varphi)]$ iff $(l,l') \in R_{K_{t,a}}$ を満たす全ての l' に対して $l' \models \varphi$.
8. $l \models [t, CMT(a,\varphi)]$ iff $(l,l') \in R_{CMT_{t,a}}$ を満たす全ての l' に対して $l' \models \varphi$.

各到達可能関係 $R_{K_{t,a}}$ が等値関係であり, 各到達可能関係 $R_{CMT_{t,a,b}}$ が推移律と連鎖律を満たすものとする. したがって, 様相演算子 K と CMT はそれぞれ S5 と KD4 の演算子となる. 対応する公理系を以下に示す.

$[t, K(a,\varphi)] \wedge [t, K(a, \varphi \supset \psi)] \supset [t, K(a,\psi)]$.

$[t, K(a,\varphi)] \supset \varphi$.

$[t, K(a,\varphi)] \supset [t, K(a, [t, K(a,\varphi)])]$.

$\neg[t, K(a,\varphi)] \supset [t, K(a, \neg[t, K(a,\varphi)])]$.

$[t, CMT(a,b,\varphi)] \wedge [t, CMT(a,b,\varphi \supset \psi)] \supset [t, CMT(a,b,\psi)]$.

$[t, CMT(a,b,\varphi)] \supset [t, CMT(a,b,[t, CMT(a,b,\varphi)])]$.

$\neg[t, CMT(a, \varphi \wedge \neg\varphi)]$.

3.2 他の様相性の導入

本節では, 知識と責務を表す様相演算子を用いて, 信念, 選択, 能力を表す様相演算子 B, CH, CAN を定義する. まず, [Shoham and Moses, 1989] にしたがって, 信念を取り消し可能な知識として定義する.

$$[t, B(a, \varphi, \psi)] \stackrel{\text{def}}{\equiv} [t, K(a, \psi \supset \varphi)] \wedge ([t, K(a, \neg\psi)] \supset [t, K(a, \varphi)]).$$

ここで, $[t, B(a, \varphi, \psi)]$ は, 時刻 t にエージェント a が仮定 ψ のもとで論理式 φ を信じていることを表わす. この定義は, 仮定が論理式を含意することを知っていて, かつ, 仮定が成立しないことを知っていることが論理式を知っていることを含意することによって信念が定式化されることを示している. 知識が S5 の体系にしたがうとき, この定義による信念は KD45 の体系にしたがうことが示されている [Shoham and Moses, 1989].

次に, 選択を, 自分自信に対する責務と定義する. $[t, CH(a, \varphi)]$ は, 時刻 t にエージェント a が論理式 φ を選択することを表わし, 次で定義される.

$$[t, CH(a, \varphi)] \stackrel{\text{def}}{\equiv} [t, CMT(a, a, \varphi)].$$

さらに, 能力を, 選択と知識を用いて定義する. $[t, CAN(a, \varphi)]$ は, 時刻 t にエージェント a が論理式 φ を実現する能力をもつことを表わし, 次で定義される.

$$[t, CAN(a, \varphi)] \stackrel{\text{def}}{\equiv} [t, K(a, [t', CH(a, \varphi)] \supset \varphi)],$$

ここで, t' は文 φ の時間引数である. この定義は, ある論理式を実現する能力をもつことは, その論理式を選択すれば実現することを知っていると同値であることを示している.

4 エージェント記述言語に対する動作記述の導入

前述のように, われわれは, プログラムをエージェントによって実行されるべき動作を記述したものとしてとらえている. しかし, 前節で定義したエージェント記述言語は動作を記述する枠組を与えていない. そこで, 本章では, エージェント記述言語に動作記述を導入する.

4.1 動作の定義

我々は, 動作を人工知能の古典的な方法 — 前件と後件の組で表現する. 前件と後件はエージェント記述言語の文で表現される. 例えば, 時刻 t においてエージェント a が点 p から点 q に移動する動作 $[t, \mathbf{Move}(a, p, q)]$ は次のように定義できる.

$[t, \mathbf{Move}(a, p, q)]$

前件: $[t, At(a, p)]$

後件: $[t+1, At(a, q)]$

ここで, $At(a, p)$ はエージェント a が点 p にいることを表す述語であり, 1 は動作に要する時間を表す. この定義は, 時刻 t に点 p にいたエージェント a が, 動作 $[t, \mathbf{Move}(a, p, q)]$ によって時刻 $t+1$ に点 q に『移動』することを示している.

エージェントは, 動作に関する信念をもっているかも知れない. 例えば, 動作『移動』によって, エージェントの位置が変わるという信念は次のように表わすことができる.

$$[t, B(a, [t', At(a, p)] \wedge [t', \mathbf{Move}(a, p, q)] \supset [t'+1, At(a, q)])]$$

4.2 通信動作のプリミティブ

前節で定義したエージェント記述言語は, 知識や責務といったエージェントの心的状態を, 世界に関する事実と同様に記述可能な統一的な枠組を与えている. したがって, 通知, 要求, 約束などの通信動作をエージェントの心的状態上の動作として定義することができる. 例えば, 時刻 t においてエージェント a がエージェント b に φ ということを通知する動作 $[t, \mathbf{Inform}(a, b, \varphi)]$ は次のように定義できる.

$[t, \mathbf{Inform}(a, b, \varphi)]$

前件: $[t, K(a, \varphi)]$

後件: $[t+1, K(b, [t, K(a, \varphi)])] \wedge [t+1, K(a, [t+1, K(b, \varphi)])]$

すなわち, 動作 $[t, \mathbf{Inform}(a, b, \varphi)]$ の前件は, 時刻 t にエージェント a が φ を知っていることである. また, 後件は, 時刻 t にエージェント a が φ を知っていることを, 時刻 $t+1$ にエージェント b が知っていて, かつ時刻 $t+1$ にエージェント b が φ を知っていることを, 時刻 $t+1$ にエージェント a が知っていることである.

同様に, 時刻 t においてエージェント a がエージェント b に φ を実現するように要求する動作 $[t, \mathbf{Request}(a, b, \varphi)]$ は次のように定義できる.

$[t, \mathbf{Request}(a, b, \varphi)]$

前件: $[t, B(a, [t, CAN(b, \varphi)])]$

後件: $[t+1, K(a, [t+1, CMT(b, a, \varphi)])]$

すなわち, 動作 $[t, \mathbf{Request}(a, b, \varphi)]$ の前件は, 時刻 t にエージェント b が φ を実現する能力があると時刻 t にエージェント a が信じていることである. また, 後件は, 時刻

$t+1$ にエージェント b がエージェント a に対して φ を実現する責務をもっていることを，時刻 $t+1$ にエージェント a が知っていることである．知識の性質を用いて，後件から，時刻 $t+1$ にエージェント b がエージェント a に対して φ を実現する責務をもっていることが導かれる．

また，上述の通信動作のプリミティブを用いて，新たな通信動作を定義することができる．例えば，質問という通信動作は，知識に対する要求と定義できる．具体的には，時刻 t においてエージェント a がエージェント b に φ を質問する動作 $[t, \mathbf{Query}(a,b,\varphi)]$ の定義は次の通りである．

$$[t, \mathbf{Query}(a,b,\varphi)] \equiv [t, \mathbf{Request}(a,b,[t+1, K(a,\varphi)])]$$

5 エージェント指向プログラミング言語

われわれは，エージェントがあるプログラムにしたがうことを，そのエージェントがある心的状態をもつことと捉えているので，プログラムに対応する様相演算子を導入する．プログラムに対応する様相演算子 $PROG$ はエージェントと文の2引数をとる．直観的には，$[t, PROG(a,\varphi)]$ はエージェント a が時刻 t に文 φ に従うことを表す．ここで，文 φ は，エージェントがある状況において自分自身の責務 (Commitment) を果たすために実行すべき動作を示すものである．したがって，プログラム文は次のように定義される．

プログラム文は，γ を責務，ψ を文，α を動作あるいはその連言としたとき，以下の文である．

$$[t, PROG(a, \gamma \wedge \psi \supset [t, CH(a,\alpha)])].$$

例えば，航空機の予約カウンタの受付エージェント $Carol$ がもつプログラムの例を，以下に示す．

$$\begin{aligned}
&[t, PROG(Carol, [t, CMT(Carol, a, [t_1, HasRsrv(a, flt)])] \\
&\quad \wedge [t, CAN(Carol, [t, HasRsrv(a, flt)])] \\
&\quad \supset [t, CH(Carol, [t, \mathbf{Book}(Carol, a, flt)] \\
&\quad \wedge [t, \mathbf{Inform}(Carol, a, [t_1, HasRsrv(a, flt)])] \\
&\quad \wedge [t, \mathbf{Revoke}(Carol, a, [t_1, HasRsrv(a, flt)])])])]
\end{aligned}$$

ここで，動作 $[t, \mathbf{Book}(Carol, a, flt)]$ は，時刻 t に受付エージェント $Carol$ が乗客エージェント a を航空便 flt に予約する動作である．このプログラムは，$Carol$ が a に対して，時刻 t_1 に a が航空便 flt に予約をもつことを実現するという責務をもっているとき，もし，$Carol$ がその責務の内容を実現する能力があるならば，次の3つの動作を選択することを表している: 1) a を航空便 flt に予約し，2) 予約がとれたことを a に通知し，3) 予約するという責務を棄却する．

6 インタープリタ

6.1 一般的なアルゴリズム

インタープリタの主要な手続きは，Select と Update の2つである．Select は，与えられたエージェントが与えられた時刻に実行すべき動作の集合を出力する手続きである．また，Update は，与えられたエージェントの与えられた時刻における心的状態を，外部からの入力を用いて更新する．インタープリタの基本的な動作は，次の通りである．

1. ある時刻において実行すべき動作を，手続き select を用いて決定する．
2. 動作の実行によって，エージェントの心的状態の一部，時間が更新される． 部分的な心的状態の変化を基に，エージェントの心的状態を，手続き update を用いて更新する．
3. 1 に戻る．

以下，手続き Select と Update の詳細について述べる．

手続き: $\mathbf{Select}(t, a, \Pi, \Delta, \Gamma)$

入力:

t: 時刻

a: エージェント

Π: エージェント a が時刻 t においてもっているプログラム文の集合

Δ: 文の集合

Γ: エージェント a が時刻 t においてもっている責務の集合

出力: 動作の集合

1. 各プログラム $[t, Prog(a,\gamma,\psi,\alpha)] \in \Pi$ に対して，$\Gamma \vdash \gamma$ かつ $\Delta \vdash \psi$ が成立するプログラムを集める．このプログラム文の集合を π とする．
2. もし，π が空集合なら空集合を返して，終了する．
3. そうでなければ，各プログラム文 $[t, Prog(a,\gamma,\psi,\alpha)] \in \pi$ に対して，動作 α を集め，この動作の集合を返して終了する．

エージェント記述言語に対する健全かつ完全な証明手続きを仮定して，上記手続きの健全性と完全性を証明することができる．

Update は，与えられたエージェントの与えられた時刻における心的状態を，外部からの入力を用いて更新する．

手続き: $\mathbf{Update}(t, a, \Delta, \Phi)$

入力:

t: 時点

a: エージェント

Δ: 心的状態を表す文の集合

Φ: 外部からの入力を表す文の集合

出力: 心的状態を表す文の集合

1. $\Delta' = \Delta \cup \Phi$ なる Δ' を構成する．
2. 各 $[t', \psi] \in \Delta'$ に対して，$[t, \psi] \cup \Delta'$ が無矛盾ならば，$[t, \psi]$ を Δ' に付け加える．
3. 集合 Δ' を返して，終了する．

6.2 プログラミング言語に対する制限

前節では，エージェント記述言語に対する健全かつ完全な証明手続きを仮定していたが，効率的な証明手続きはまだ知られていない．そこで，本節では，プログラミング言語の表現能力を制限することにより，効率的な証明を可能にする．

2.1 節で述べた整式の生成規則の 4 を次のように変更する．

4. $\varphi_1, \cdots, \varphi_n$ および ψ を整式とするとき，$\varphi_1 \vee \cdots \vee \varphi_n \vee \neg\psi$ は整式である．

直観的には，この制限は，整式をホーン節に制限することに対応するものである．これまでに述べた例は全てこの制限を満たしている．

7 インタープリタ・シミュレータの実現例

本章では，制限されたプログラミング言語に対するインタープリタとシミュレータの実現例について述べる．本インタープリタとシミュレータは Prolog を用いて実現されている．インタープリタは，セレクト，アップデート，ブルーバの 3 モジュールから構成される．最初の 2 モジュールは，前章で述べた 2 手続きに対応し，最後のモジュールは定理証明器である．また，シミュレータはクロックと，動作解釈部の 2 モジュールから構成される．シミュレータの基本的な動作は以下の通りである．

1. クロック部から得られた時刻を各エージェントのインタープリタに渡す．
2. 各エージェントのインタープリタから出力された動作を動作解釈部で解釈し，各動作の効果 (後件) を対応するエージェントのインタープリタに渡す．
3. 1 に戻る．

7.1 プログラム例

5 章で述べた航空機の予約システムに対する応用例のプログラムの一部を以下に示す．

```
can(T, carol, book(T, A, Flt)) :-
    k(T, carol, notFullFlt(T, Flt))).

prog(0, carol, cmt(T, carol, A, book(T1, A, Flt)),
      can(T, carol, book(T, A, Flt)),
      (book(T, carol, A, Flt),
       inform(T2, carol, A, hasRsrv(T2, A, Flt)),
       revoke(T, carol, A, book(T1, A, Flt)) ))
  :- delay(T, T2).

prog(0, carol, cmt(T, carol, A, book(T1, A, Flt)),
      k(T, carol, fullFlt(T, Flt)),
      (revoke(T, carol, A, book(T1, A, Flt)),
       inform(T, carol, A, fullFlt(T, Flt)),
       refuse(T2, carol, A, book(T1, A, Flt)) ))
  :- delay(T, T2).
```

8 おわりに

互いに通信しながら協調動作するようにエージェントをプログラムするためのプログラミング言語 AOPL を提案した．本稿では，まずエージェント指向プログラミング言語に対する要求条件を述べ，プログラミング言語 AOPL を，プログラムに対応する様相性と動作記述とで拡張されたエージェント記述言語として定義した．次に，与えられた時刻にあるエー

ジェントが実行すべき動作を出力する AOPL のインタープリタを構成し，インタープリタを効率的に実行するためのプログラミング言語に対する制限条件を示した．さらに，制限されたプログラミング言語のインタープリタと，インタープリタが出力した動作を解釈実行するシミュレータの実現方法について述べた．

今後の課題として，6.2 節で導入したプログラミング言語に対する制限の有効性を理論的に示すことが挙げられる．また，我々は，本プログラミング言語に基づくプラン認識理論の構築を検討中である．

謝辞

Stanford 大学において指導していただいた Yoav Shoham 助教授，ならびに，本研究の機会を与えていただいた神奈川大学の村上国男教授に感謝します．また，討論していただき，有益な助言をいただいた Kurt Konolige, Martha Pollack, Les Gasser の各氏に感謝します．

参考文献

[Agre and Chapman, 1987] P. E. Agre and D. Chapman, Pengi: An Implementation of a Theory of Activity, Proc. of IJCAI-87, pp. 268–272, 1987.

[Allen and Perrault, 1980] J. F. Allen and C. R. Perrault, Analyzing Intention in Utterances, Artificial Intelligence 15, pp. 143–178, 1980.

[Appelt, 1985] D. E. Appelt, Planning English Sentences, Cambridge University Press, 1985.

[Bratman *et al.*, 1988] M. E. Bratman, D. J. Israel, and M. E. Pollack, Plans and Resource-Bounded Practical Reasoning, Computational Intelligence 4, pp. 349–355, 1988.

[Cohen and Levesque, 1990] P. R. Cohen and H. J. Levesque, Intention is Choice with Commitment, Artificial Intelligence 42, pp. 213–261, 1990.

[Cohen and Perrault, 1979] P. R. Cohen and C. R. Perrault, Elements of a Plan-Based Theory of Speech Acts, Cognitive Science 3, pp. 177–212, 1979.

[Georgeff and Ingrand, 1989] M. P. Georgeff and F. F. Ingrand, Decision-Making in an Embedded Reasoning System, Proc. of IJCAI-89, pp. 972–978, 1989.

[Kripke, 1971] S. Kripke, Semantical Considerations on Modal Logic, in L. Linsky eds. Reference and Modality, pp. 63–72, Oxford University Press, 1971.

[Moore, 1980] R. C. Moore, Reasoning about Knowledge and Action, TR-191, SRI International, 1980.

[Searle, 1969] J. R. Searle, Speech Acts, Cambridge University Press, 1969.

[Shoham, 1989] Y. Shoham, Time for Action: On the Relationship between Time, Knowledge and Action, Proc. of IJCAI-89, pp. 954–959, 1989.

[Shoham and Moses, 1989] Y. Shoham and Y. Moses, Belief as Defeasible Knowledge, Proc. of IJCAI-89, pp. 1168–1173, 1989.

[Thomas *et al.*] B. Thomas, A. Schwartz, and Y. Shoham, Modalities in Agent-Oriented Programming, *forthcoming.*