プログラミング 18－7
(1998. 3. 23)

BDIアーキテクチャを用いた分散人工知能戦略選択機構

山崎 賢治†, 楢崎 修二††, 牛島 和夫†
† 九州大学大学院 システム情報科学研究科, †† 九州工業大学 工学部
〒812-8581 九州大学大学院 システム情報科学研究科 情報工学専攻
Tel : 092-642-3872, E-mail : yamaken@csce.kyushu-u.ac.jp

分散人工知能 (DAI) とは，分散環境に存在する知的処理主体の協調的，合理的な振舞いに関する研究分野であり，今後の分散システムの発展に寄与するものとして期待を集めている．しかし各DAI研究がある特定の問題領域に特化した形で発展してきたために，各研究の成果であるDAIアルゴリズムは往々にして問題依存となり，その実際的な応用を困難にしている．この現状を打開するために，本稿では再利用可能な複数のDAIプログラムを同一の意思決定機構の下で実行・評価できるような枠組を提案する．その意思決定機構の設計に際しては，人間の思考過程をモデル化したBDIアーキテクチャの理論を用いた．

キーワード：分散人工知能, BDIアーキテクチャ, メタオブジェクトプロトコル (MOP), マルチエージェントシステム (MAS), メタレベル計算

A BDI decision making architecture for selecting DAI strategies

Kenji Yamasaki†, Shuji Narazaki††, Kazuo Ushijima†
† Graduate school of information science and electrical engineering , Kyushu University
††Faculty of engineering, Kyushu Institute of Technology
Dept. of Comp. Sci. and Comm. Eng., Kyushu University, 812-8581 Japan
Tel : + 81-92-642-3872, E-mail : yamaken@csce.kyushu-u.ac.jp

Distributed Artificial Intelligence (DAI), an important research field for future advances of distributed systems, is concerned with the collaborative and rational behavior of distributed intelligent systems. Since each research has been developed using a particular problem as its testbed, DAI algorithms tend to be "problem-specific", and difficult to be put into practical use. To improve this tendency, this paper proposes a framework in which a variety of reusable DAI programs can be executed and tested. The framework embodies a decision making architecture that we developed basing on the pragmatic theory of BDI architecture, which has been modelled on human beings' deliberative processes.

Key Words : Distributed Artificial Intelligence (DAI), BDI architecture, MetaObject Protocol (MOP), MultiAgent Systems (MAS), Meta-level computation

1 はじめに

分散人工知能 (DAI : Distributed Artificial Intelligence) とは，分散環境に存在する知的処理主体である“エージェント”の協調的，合理的な振舞いに関する研究分野であり，今後の分散システムの発展に寄与するものとして近年多くの期待を集めている．エージェントは物理的，または機能的に分散しているため，各エージェントは周囲の状況を把握し，周囲のエージェントと通信しあい，分散環境の中でどう合理的に振舞うべきかを自律的に決定する能力を持つ．DAIの研究はエージェントの集団としての問題解決能力を分析することにより発展してきた．

一般的に各エージェントの振舞いに関する研究はある特定の問題領域に特化した形で発展してきた．このため，その研究成果であるDAIアルゴリズムおよびそのプログラムは往々にして問題依存となり，その実際的な応用を困難にしている．この現状を打開するために必要なのは以下に示すようなDAIプログラミング手法の改善である．

- 各DAIプログラムを再利用できる部分と出来ない部分とに分離記述し，再利用できる部分に関してはそのプログラムが想定している根本的状況を分析する．
- 再利用可能なDAIプログラムをライブラリ化し，統一的な枠組の下で状況に応じて選択し，実行する．

これらのプログラミング手法を容易にするためにはプログラミング環境を改善することが必要である．そこで本稿では，再利用可能な複数のDAIプログラムを同一の意思決定機構の下で実行・評価できるような枠組を提案する．

この枠組はDAIプログラムが再利用できる部分と出来ない部分とに分離記述できるという考えに基づいている [9]．我々は再利用できるDAIプログラムを“DAI戦略”，または単に“戦略”と呼んでいる．第2節ではその戦略の分離記述について説明する．次にDAI戦略を汎用化するために第3節ではDAI戦略およびそれらを発火する契機となる状況変化の分析をそれぞれ我々の観点から行なう．

また我々は，状況変化に応じてDAI戦略を柔軟に選択するための意思決定機構を枠組に導入した．この機構の設計には人間の思考過程をモデル化したBDIアーキテクチャの理論を用いた．BDIアーキテクチャとはエージェントを信念 (**B**elief)・欲求 (**D**esire)・意図 (**I**ntention) という三段階の心的状況を持つ処理主体として捉える機構である [5]．第4節ではそのBDIアーキテクチャについて説明し，それを基に設計した我々独自の意思決定機構を解説する．

第5節では枠組の実装方法について説明し，その利用例を第6節で紹介する．最後に第7節では枠組の特長および関連研究を示し，第8節では本稿をまとめ，残された課題について検討する．

2 DAIプログラムの分離記述

DAIプログラムはエージェントの知的動作を記述したプログラムである．各エージェントは基本的に以下のような能力を持つ．

- 環境に影響を及ぼす動作によって間接的に，または通信によって直接的に他のエージェントと相互作用できる．
- 様々な状況に応じて各自で動作を決定する能力を持つ．

各エージェントは集団の一員としてこれら二つの代表的な能力を駆使し，問題を解決する．

このようなエージェントの振舞いを記述したDAIプログラムは二つのレベルに分割することができる．一方はローカルで単純なエージェントの処理を記述した“問題レベル”で，もう一方は状況に応じた柔軟な処理を記述した“戦略レベル”である．

問題レベルプログラムは状況を考慮に入れない問題固有の処理を記述しており，一般に問題への依存度が高い．一方，戦略レベルプログラム (または単に“戦略”と呼ぶ) は様々な“状況”に基づいた処理であるので一般的に問題への依存度は低い．

各エージェントは物理的，または機能的に分散しているため，“分散環境”特有の制約の下で作業しなければならない．戦略がそれらの制約に対する処理を記述したプログラムである．戦略が対処する制約には，通信コスト，情報の局所化，単独では作業できない局面といったものがある．

図1に示すのがそれら二つのレベルの分離を説明する例である．この例はDAI研究で用いる問題解決法の一つであり，分散探索と呼ぶ [1]．この種の問題解決法では，グラフの最短経路の探索といった一つの問題を複数のエージェントで分担して解決する．各エージェントは共有メモリから取り出した部分探索空間をローカルに探索し，最適解を求める．しかし問題解決の効率を考慮すると，エージェントがローカルに発見した解を，周囲の状況を考慮した効率的な方法で他エージェントに知らせることによって探索空間を削減することができ，全体としての問題解決効率を上げることができる．

図1に示すように，この問題解決を記述したプログラムは二つのレベルに分けて考えることができる．一方がローカルな探索アルゴリズムを記述した探索プログラムで，もう一方が通信コストや探索結果の質，他者の問題解決効率といった状況を考慮して効率的な通信方法を記述した通信管理プログラムである．前者が問題レベルのプログラムに対応し，後者が戦略に対応する．通信管理プログラムは特に通信戦略と呼ぶ．前者が問題依存であるのに対し，通信戦略は状況に基づいて設計すべきプログラムである．

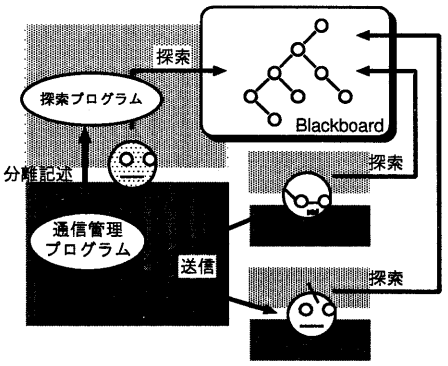

図 1: 分離記述例：分散探索

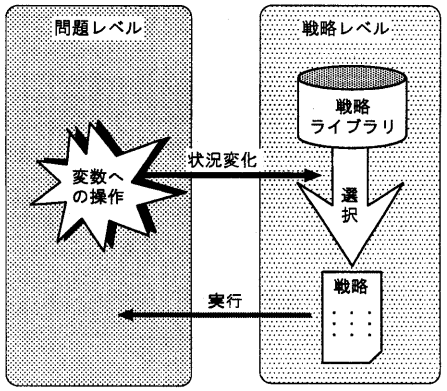

図 2: 問題レベルと戦略レベル

通信戦略のように問題領域に依存しないプログラムはライブラリとして再利用すべきであり，またDAI研究者はそのように設計すべきであるというのが本研究の基本的立脚点である．DAIプログラムを二つのレベルに分離して記述するという枠組は以前我々の研究で提案した[9]．しかし戦略をライブラリとして利用するための枠組としては不十分であった．その理由は以下に示す通りである．

- 戦略を汎用化，つまりライブラリ化するための指針を示していない．
- 以前の枠組には状況に応じて適切な戦略を選択する機構が備わっていない．

まず一つめの問題点を解消するためには戦略の基本的性質，および問題レベルプログラムとの関係を分析する必要がある．次節 (3節) ではこの点に関しての分析を我々の観点から行なう．二つめの問題点を解消するために我々は戦略選択機構を枠組に導入した．この点に関しては第4節で解説する．

3 戦略およびイベントの分析

図2に示すのが前述の問題レベルと戦略レベルとの関係を示したものである．ライブラリ化した戦略は問題レベルプログラムで発生する状況変化に伴って起動すべきプログラムである．状況変化はエージェントが問題レベルで保持する変数の変化または変数への操作に対応する．しかし変数への操作全てが戦略の必要性を示唆するわけではない．ある特定の性質を持つ変数のみに限定できるはずである．そこで本稿では戦略の必要性を示唆する変数をとくに“キー変数”と呼び，キー変数への操作の結果起こる状況変化を“イベント”と呼ぶ．

戦略の汎用化の指針を示すにはDAI戦略の性質，およびその根源であるキー変数の性質それぞれを分析する必要がある．本節ではそれぞれに関する分析を行なう．

3.1 DAI戦略の分類

一般的にDAIに関する研究は大きく二つの領域に分けることができる．一つが分散問題解決 (DPS : Distributed Problem Solving) で，もう一つがマルチエージェントシステム (MAS : MultiAgent Systems) である．

DPSは一つの問題を複数のエージェントに分割し，各エージェントがどのように振舞うべきかに関する研究領域であり，一般的に各エージェントの能力は均一である．一方MASはエージェントが単独では対処できない問題をいかに複数で相互作用しあって処理するかに関する研究領域であり，エージェントの能力および役割は一般的に異なる．

DAI戦略をDPSとMASとの違いにしたがって分類したのが図3である．

DPSには主に物理的に分散した環境でいかに情報を共有または獲得するかに関する戦略が属する．一方，MASには主に機能的または能力的に分散したエージェント間での相互作用に関する戦略が属する．図のように戦略を分類してみると，DPS戦略はMAS戦略の基礎となるものである．言い替えれば，DPS戦略もMAS戦略も処理の複雑さという点から見ると分けて捉える必要はない．従って，我々が提案する枠組の中で，分類の枠を越えて戦略をライブラリとして用いることができる．本研究では手始めとして図3に示す情報管理戦略を取り扱い，それに属する戦略についてその根源であるイベントについて分析した．情報管理戦略は以下に示す三種類の戦略からなる．

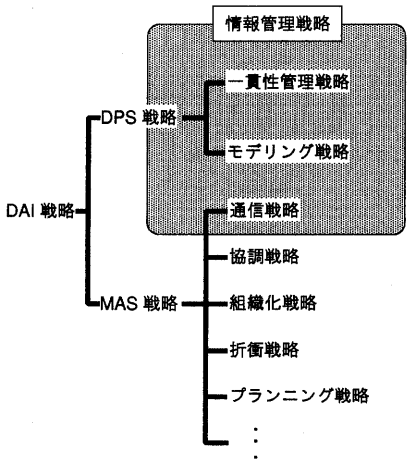

図 3: DAI 戦略の分類

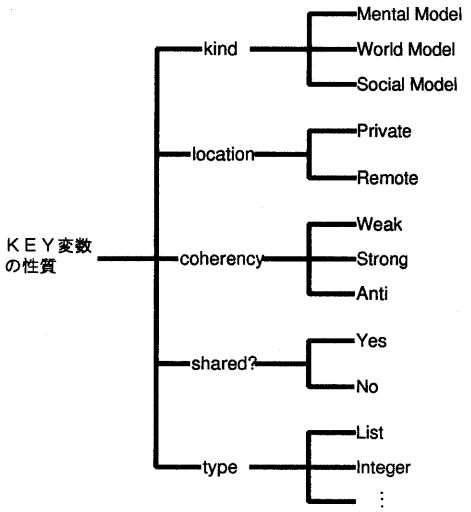

図 4: キー変数の性質の分類

一貫性管理戦略 どのような一貫性を保持しつつ他者と情報を共有すべきかに関する戦略.

モデリング戦略 他者の問題解決状況または動作環境といった不確実で動的に変化する情報をいかにモデル化するかに関する戦略.

通信戦略 通信コストや他者の状態といった周囲の状況を基にして通信頻度や通信対象を決定する戦略.

3.2 イベントの分析

前述のように, 戦略の必要性を示唆するイベントは問題レベルプログラム中の変数への操作に対応する. 図 4に示すように我々は情報管理戦略の必要性を示唆するようなキー変数にはどのような性質があるかを列挙し, それぞれについて分析してみた.

kind 変数の利用目的に関する性質. エージェント自身の内部情報を表現する Mental Model とエージェントが存在する環境情報を表現する World Model, そして他者に関する情報を表現する Social Model とに分類する.

location その変数の値が存在する物理的場所に関する性質. エージェントの手元に存在する場合 (private) と, 離れた場所に存在する場合 (remote) とに分類する.

coherency 変数の一貫性に関する性質. この性質の解釈は変数の利用目的によって異なる. もし変数の値が不確実である場合にはその値に要求する確実度を表し, もしその変数の値がエージェント間での共有情報となっている場合にはその共有時の一貫性を表す. 弱い一貫性 (weak) や強い一貫性 (strong) といった一貫性の程度によって分類する.

shared? その変数の値は他エージェントと共有すべきか否かに関する性質.

type 変数の型.

キー変数の各性質はその変数への操作が引き起こすイベントの性質および複雑さを特徴づける. 言い替えれば, そのイベントに対処するのに必要な戦略の性質および複雑さを特徴付けることになる. このようにキー変数の性質に基づいてイベントを分類し, そのイベントの性質に基づいて各戦略を設計することにより戦略を汎用化することができる.

4 戦略選択機構

前節では戦略およびイベントの性質の分析を行ない, 戦略の汎用化のための指針を示した. 戦略が汎用化できればライブラリとして利用することができる. しかし前述のように我々が以前提案した枠組にはその戦略ライブラリの中から状況に応じた, 即ちイベントに応じた適切な戦略を選択し実行する機構を備えていなかった. そこで我々は BDI アーキテクチャというエージェントの意思決定機構に着目し, それを基に戦略選択機構を枠組に実装した. 本節ではまずその BDI アーキテクチャについて説明し, 次にそれを基に設計した戦略選択機構について説明する.

4.1 BDI アーキテクチャ

合理的な振舞いができるエージェントに関する研究は近年多くの注目を集めている．そのようなエージェントの合理的振舞いを実現する機構の一つにBDIアーキテクチャがある．人間の知的思考過程をモデル化したBDIアーキテクチャは，エージェントを信念(Belief)・欲求(Desire)・意図(Intention)という三段階の心的状況を持つ処理主体として捉える機構である[6]．多くの研究がBDIアーキテクチャの形式化を行なってきた．しかしそれらの形式化は完全な形で実際のシステムへ応用するには複雑すぎる．そこで我々はその理論を基に，より実際的な観点から各要素の役割を説明する．図5に示すのがエージェントのBDIアーキテクチャに基づく意思決定過程である．この図に基づいて各要素間の関係も合わせて説明する．

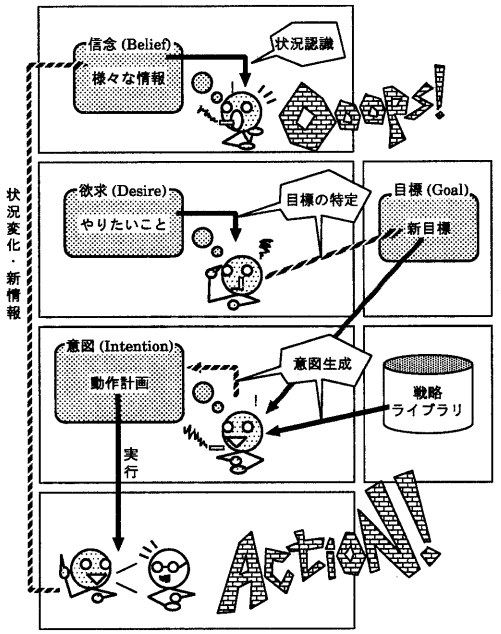

図 5: BDI アーキテクチャ

信念 (Belief) 信念はエージェントが持つ情報を表現する要素であり，エージェントが信じていることまたは知っていることがらに対応する．まずエージェントは状況変化を信念の変化として認識する．

欲求 (Desire) 欲求はエージェントが欲する可能性のあるタスクの達成を動機づける要素であり，エージェントが欲することがらに対応する．認識した状況に反応して，エージェントは信念を基にして状況を把握し，欲求の一つを目標 (Goal) として特定する．即ち，目標はその時点でエージェントが達成可能であると信じている欲求である．

意図 (intention) 意図はエージェントが現在実行している，または実行しようとしている動作計画を表す要素であり，エージェントが意図していることがらに対応する．選択した目標を基に，エージェントはそれを達成するための動作計画を意図し，実行する．動作の背後の意図では，動作遂行の時点で存在する“行為意図 (intention in action)”と，エージェントの将来の行為を規定する，“将来に向けられた意図 (future-directed intention)”とに区別できる[8]．前者はエージェントのすぐ未来の動作計画を規定するのに対し，後者は長期に渡る未来の意思決定に影響を及ぼす．

このようにBDIアーキテクチャを用いることによりエージェントの内部および外部の状況を的確に表現することができ，また目標や意図といったエージェントの現在の心的状態が未来の意思決定に影響するなどの複雑な状況をも表現することが可能となる．意図を決定する段階で図5のようにエージェントが戦略ライブラリを参照すれば，強力な戦略決定機構として枠組に導入することができる．

4.2 メタ BDI 意思決定機構

BDIアーキテクチャの実際的な理論に基づいて，我々は戦略選択のための機構を枠組に導入した．図5に示すのがその概観である．

戦略選択機構は問題レベルで起こる状況変化を常に把握し，それに応じて戦略選択を行なう必要があるため，我々はこの機構を問題レベルでのキー変数への操作に付随するメタレベル計算として実現した．その意味を反映して我々はこの機構を“メタBDI意思決定機構”と呼ぶ．

問題レベルのプログラムと戦略とを戦略汎用化のための指針 (前述) に従って分離記述した場合，各々が与えるキー変数および戦略の性質を枠組が意思決定機構の各要素へと割り当てる．BDI問題レベルのキー変数の性質を信念に，各戦略の発火条件を欲求に，そして各戦略を戦略ライブラリに格納する．これらの要素を基にしてメタBDI意思決定機構はイベントの性質に応じた適切な戦略を選択し実行する．

図6に示すように，メタBDI意思決定機構は，Event Monitorと Goal Selector，そして Means-End Reasonerという三つのモジュールが管理している．問題レベルにおいて，例えばあるキー変数に対して更新が起こったとすると，そのメタレベル計算としてメタBDI意思決定機構が始動する．するとまず Event Monitor がそのキー変

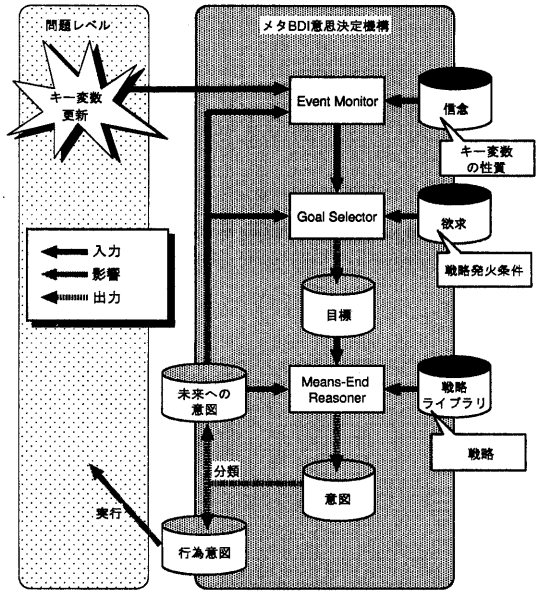

図 6: 戦略決定機構の概観

数の性質を保持する信念および目標や意図といった，現在のエージェントの内部状態を基にイベントの性質を解析し，Goal Selector へ渡す．次に Goal Selector はそのイベントの性質と一致する発火条件を持つ欲求を選択し，目標として採用する．最後に Means-End Reasoner がその目標達成のために最適な戦略を戦略ライブラリから選び，それを行為意図または未来へ向けられた意図に分別する．行為意図についてはすぐに実行し，周囲のエージェントや環境に影響を及ぼす．未来へ向けられた意図についてはそのまま格納し，その後の意思決定に影響を及ぼす．

5 枠組の実装

前述のように我々はメタ BDI 意思決定機構をメタレベル計算として実現した．この機構を備えた枠組の実装には Tiny-CLOS を用いた [3]．Tiny-CLOS は CLOS (Common Lisp Object System) をより簡素化した Scheme 上の実装で，十分強力な MOP(MetaObject Protocol) を持つ．MOP を使ってメタオブジェクトに手を加えることにより，言語のデフォルトの振舞いを利用者の意向に応じて変更または拡張することができる [4]．Tiny-CLOS の MOP はインスタンス表現や操作などを利用者に一部公開しているので，その部分に手を加えることによってメタレベル計算としてのメタ BDI 意思決定機構の実装を行なった．

一般的に MOP を用いた言語の振舞いの拡張は以下の手順で行なう．

(1) 新しいメタクラスを定義する．

(2) そのメタクラスに特定化したメソッドを新しく追加する．

具体的には，問題レベルのエージェントのクラスのメタクラスと，そのエージェントインスタンスが持つキー変数用のメタクラスを新しく定義し，そのメタクラスに属するエージェントが，キー変数のクラスに属する変数に対して参照または代入を行なった場合の言語の振舞いを拡張した．デフォルトの変数に対して参照または代入が起こった場合には単に値を返すのみまたは値を格納するのみである．新しく定義したメタクラスのインスタンスに対してはこれらデフォルトの処理に加えてメタ BDI 意思決定機構の起動を行なわせるように拡張した．

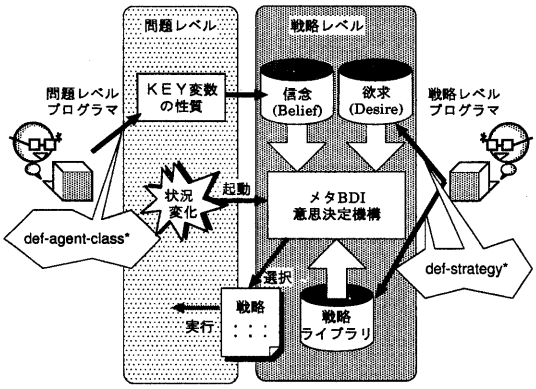

図 7: 枠組の概観

図 7に示すのがメタ BDI 意思決定機構を導入した枠組の概観である．枠組を利用するプログラマにはそれぞれ幾つかのマクロを提供している．

問題レベルのプログラムを与えるプログラマには，メタ BDI 意思決定機構を利用するエージェントクラス，およびそのキー変数の性質を定義するためのマクロ def-agent-class* を提供している．利用者はキー変数の性質の分類 (図 4) に基づいて設計したオプションを用いることによってキー変数の指定とその性質とを定義することができる．このマクロを用いて定義したキー変数の性質はメタ BDI 意思決定機構が信念として保持する．

一方，戦略を定義する戦略レベルのプログラマには，戦略とその発火条件とを定義するためのマクロ def-strategy* を提供している．利用者はその戦略が必要となるような状況，すなわち戦略の発火条件を，イベントの性質と，エージェントがその時点で持つ目標および意図の条件と

を指定することにより定義できる．イベントの条件に関しては，キー変数の性質を表現するのに用いたオプションを用いて定義できる．このマクロを用いて定義した戦略はライブラリとして，また戦略発火条件は欲求としてメタBDI意思決定機構が保持する．

枠組で利用する戦略の種類が多くなるほどそれらの間の差別化を行なう必要が出てくる．従って新しい戦略発火条件を定義するための新しいオプションを追加することが必要となる．そのような必要性を持つ利用者のために，新しいオプションを追加するためのマクロadd-options*を提供している．このマクロを用いて定義したオプションは問題レベルでのキー変数の性質を定義する時に利用できるようになる．

なお，戦略の選択をより合理的に，また戦略発火条件をより柔軟に表現できるようにするために，欲求の表現および選択にはScheme上のPrologの実装であるSchelogを用いている[2]．メタBDI意思決定機構のモジュールEvent Monitorはオプションで指定されたキー変数の性質を基にしてイベントの性質をその時点での事実を表す論理式へと変換する．戦略発火条件はマクロdef-desire*が同様に論理式に変換してあり，Goal Selectorはそれら論理式を用いて目標を特定する．

次節では前述の分散探索の例を用いて各プログラムの定義例，およびそれを基にした枠組の処理の流れを示す．

6 枠組の利用例

問題レベルではまずマクロdef-agent-class*を用いてエージェントクラスを定義する．分散探索の例のように，エージェントがローカルに発見した現時点での最適解を表すキースロット`threshold`を持つ場合はエージェントクラス`<agent>`を以下のように定義できる．

```
(def-agent-class* <agent>
                  (list    ...
                           'threshold
                             ...          ))
```

次に`make`を用いてそのクラスのインスタンスとしてエージェントオブジェクトを作り，そこでキー変数クラス`<key-slot-class>`のインスタンスとしてキー変数を定義し，その性質をオプションを用いて指定する．このキー変数はエージェントの問題解決状況という内部状態を表し(kind=mental)，手元に存在する(location=private)．また，他エージェントとの弱い一貫性(coherency=weak)を保持して共有する(shared?=#t)ことによって全体としての処理効率を上げることができる．従って以下のように定義できる．

```
(make <agent>
      ...
      'threshold (make <key-slot-class>
                       ...
                       'kind        'mental
                       'location    'private
                       'coherency   'weak
                       'shared?     #t
                       ...                ))
```

一方，戦略レベルではマクロdef-strategy*を用いて戦略とその発火条件とを定義する．例えば，他者との弱い一貫性を保持し，ローカルな値を効率的に共有するための通信戦略`comm-st`は以下のように定義できる．

```
(def-strategy* comm-st
               (list  '('access     'set       )
                      '('location   'private   )
                      '('kind       'mental    )
                      '('coherency  'weak      )
                      '('shared?    #t         ))
               ( ...  目標に関する条件   ... )
               ( ...  意図に関する条件   ... )
               ( ...      戦略の定義     ... ))
```

第一引数は戦略名，第2～4引数はイベント，エージェントが持つ目標，そして意図，それぞれに関する戦略発火条件の定義で，第5引数がその戦略の定義である．マクロdef-strategy*は戦略名と第2～4引数を用いて欲求を，戦略名と第5引数を用いて戦略ライブラリを作成する．

以上で一通りの問題レベルおよび戦略レベルの定義は完了した．もし問題レベルプログラム実行中にエージェントが最適解候補を発見し，その値をキー変数`threshold`へ代入した場合のメタBDI意思決定機構の処理は以下のような流れになる．

(1) キー変数として定義した`threshold`の値の更新が起こるとその処理に付随するメタレベル計算としてまずEvent Monitorが始動する．

(2) Event Monitorは信念を参照し，`threshold`の性質を基にイベントの性質を論理式へと変換し，同様にエージェントが持つ現在の目標および意図の状況も論理式へ反映し，それらをGoal Selectorへ渡す．

(3) 複数存在する欲求の中で，現在の状況と発火条件とが合致するような欲求があればGoal Selectorがそれを目標として適用する．

(4) Means-End Reasonerは目標を達成するための処理即ち戦略(ここでは`comm-st`)をライブラリから選び出し，それが行為意図を表す戦略であるので行為意図として捉え，実行する．

(5) 戦略の実行が無事終了すると，メタレベル計算はデフォルトの処理(新しい変数値の格納)を行ない，処理を問題レベルへと戻す．

7 枠組の特長・関連研究

本稿ではDAIプログラミングのための枠組を提案した．この枠組をその提案の意義という点から以下のように評価できる．

- 戦略の起動はメタレベル計算として行なうので問題レベルには明示的な戦略呼び出し命令が現れない．従ってDAIプログラムの分離記述がきれいにできる．
- 様々な種類の戦略を同一の機構の下で評価できる．
- BDIアーキテクチャの実際的な理論に基づいているため，エージェントの内部および外部の状況を柔軟に表現することができる．
- 戦略の基本的な性質に基づいた設計を提案しているので，DAI研究のさらなる実用化に寄与する．
- BDIアーキテクチャの一応用例として，その実用化にも寄与する．

関連研究としては[5]や[7]を挙げることができる．どちらもBDIアーキテクチャの形式化というよりもその実際的な理論に基づくエージェントアーキテクチャの研究に重点を置いている．本研究のメタBDI意思決定機構の設計に際しては両者の理論を参考にしている．

Haddadiらによる研究[5]では，エージェント間のタスクの依頼や請け負いといった協調のための通信プロトコルを，BDIアーキテクチャを用いて合理的に選択させる機構について研究している．この通信プロトコルを戦略の一つとみなし，その発火条件を特定すれば我々の枠組の中でライブラリとして評価することができる可能性がある．今後，枠組のさらなる分析において参照すべき重要な研究の一つである．

8 まとめと今後の課題

本稿ではDAI研究において研究成果の実用化を妨げる原因に着目し，それを解消するプログラミング手法として，問題依存度の高い問題レベルプログラムと再利用できる戦略プログラムとに分けて記述するという技法を提案した．さらに戦略とその発火条件とを分析してそのプログラミングのための指針を示し，戦略選択機構を備えた枠組を提案することにより，DAIプログラミングの環境を改善し，DAI研究の実用化へ寄与することを試みた．

本稿では手始めとして，DAI戦略の基礎となる情報管理戦略に限定して枠組の設計を行なった．我々の研究室では今後もより一般的なDAI戦略への利用が可能となるように研究を進める予定である．

図3に示すような，よりMAS戦略寄りの高レベルな戦略へ応用するためには本稿の分析方針に従って戦略の性質およびキー変数の性質のさらなる分析が必要となる．これらの結果を用いて枠組およびメタBDI意思決定機構をさらに拡張してゆくことが今後の課題である．

参考文献

[1] Alan H. Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, 1988.

[2] Dorai Sitaram. Programming in Schelog. *http://www.cs.rice.edu/CS/PLT/packages/schelog/*, 1997.

[3] Gregor Kiczales. Tiny-CLOS. *ftp: parcftp.xerox.com:pub/mops/**, 1992.

[4] Gregor Kiczales, Jim des Rivières and Daniel G. Bobrow. *The Art of the Metaobject Protocol.* The MIT Press, 1991.

[5] Afsaneh Haddadi. *Communication and Cooperation in Agent Systems A Pragmatic Theory.* Springer-Verlag, 1995.

[6] Afsaneh Haddadi and Kurt Sundermeyer. Belief-Desire-Intention Agent Architectures. In G.M.P.O'Hare and N.R.Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 5, pages 169–185. Johon Wiley & Sons, inc., 1996.

[7] Klaus Fischer, Jörg P. Müller and Markus Pischel. A Pragmatic BDI Architecture. In Klaus Fischer, Jörg P. Müller and Markus Pischel, editor, *Intelligent Agents II*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 203–218. Springer-Verlag, German, 1996. Proceedings of ATAL-95.

[8] Yasuhiro Katagiri. A distributed system model for actions of situated agents. In *Conference on Information-oriented Approaches to Logic, Language and Computation*, 1994.

[9] 山崎 賢治 楢崎 修二 牛島 和夫. メタレベル計算を用いた協調処理の実現. 情報処理学会研究報告 *PRG, 95(82)*, pages 145–152, 1995.