マルチメディア通信と分散処理　50－20
（１９９１．５．１７）

分散協調型問題解決における問題解決エージェント

長木正宏　村田努　柴田知博　井内稔　山崎晴明
山梨大学

分散協調型問題解決システムにおける知識表現モデルの異なる複数エージェントの試作について報告する。分散協調型問題解決を行うために問題解決エージェントに要求される事項について考察する。そして、今回試作したシステムにおけるプロダクション、意味ネットワーク、フレームの三つの問題解決エージェントの概要について報告する。また、今回の試作において検討した、エージェント間の通信情報の形式についても考察する。

AGENTS FOR PROBLEM SOLVING ON COOPERATIVE DISTRIBUTED PROBLEM SOLVING

Masahiro Choki　Tsutomu Murata　Tomohiro Shibata　Minoru Iuchi　Haruaki Yamazaki

Yamanashi University

4-3-11 Takeda, Kofu, Yamanashi 400, Japan

This paper describes the prototype of the cooperative distributed problem solving system which consists of several agents with their own knowledge representation models. Furthermore, the demands for the agents in order to achieve CDPS are discussed. This paper also discusses the outline of these three agents: Production, Semantic Network and Frame, and the cooperative messages among agents.

1．はじめに

問題解決を行う上で直面する問題の中には、情報や資源が、地理的、時間的、機能的に分散し、問題全体が、その解決の過程において相互に依存し合う複数の部分問題に分かれてしまうような複雑な問題も存在する。そのような問題の解決に対しては、複数の異なった知識表現モデルを持つ問題解決システムが有効である。[1]

今回我々は、知識表現モデルの異なる複数エージェントから構成される分散協調型問題解決システムの試作を行った。

本稿では、分散協調型問題解決システムにおける知識表現モデルの異なる複数エージェントの試作について報告する。

2．分散協調型問題解決システムの構成

今回試作したシステムは図1のような構成をとっている。

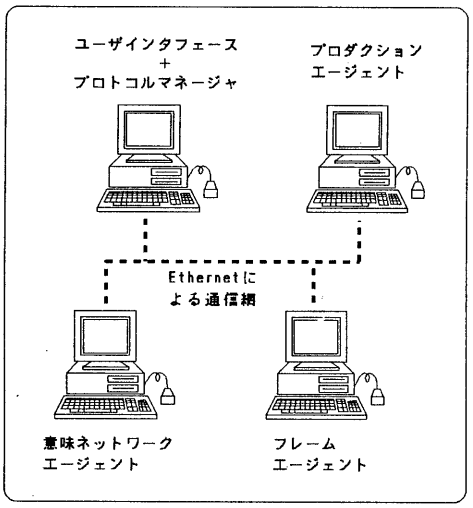

図1．分散協調型問題解決システムの構成図

問題解決エージェントとして、プロダクションエージェント、意味ネットワークエージェント、フレームエージェントの三つが存在し、問題解決におけるエージェント間の協調を支援するためのプロトコルマネージャが存在する。各エージェント及びプロトコルマネージャにはそれぞれ一台のＰＣが割り当てられ、Ｅｔｈｅｒｎｅｔを介して接続されており、互いに通信が可能である。また、プロトコルマネージャを搭載するＰＣには、システム全体の監視のためのユーザインタフェースも搭載する。

プロトコルマネージャは分散協調プロトコルによりエージェント間の協調的な問題解決の支援を行う。分散的な問題解決を行う立場から考えるとプロトコルマネージャは分散化し各エージェント内で動作すべきであるが、今回の試作ではエージェントを搭載するハードウェアの制約があるので、集中型のプロトコルマネージャを設けている。

3．問題解決エージェントが満たすべき機能

今回の試作では、エージェント間で協調して問題解決を行うために問題解決エージェントが満たすべき機能として、以下の事項を取り上げた。

(1) エージェント間で通信を行う機能を持つ：

エージェントは問題解決の過程において、他のエージェントと情報を交換するために通信を行う必要がある。したがって、システム内のエージェントは通信を行う機能を持つ必要がある。

(2) 共通表現に関する変換を行う機能を持つ：

エージェントはそれぞれが異なった知識表現を使用しているため、他のエージェントが送信した情報の内容をそのまま理解できるとは限らない。したがって、エージェントの持つ局所的な表現形式とシステム内の共通表現形式とを相互に変換するための機能を持つ必要がある。

(3) 問題解決全体を複数のステージに分割する：

問題解決全体はあらかじめ複数の問題解決ステージに分割されていなければならない。また、各問題解決ステージごとに目標となる解がどういった

ものであるかを定めておき、エージェントのアプリケーションとして記述しておく必要がある。エージェントは各問題解決ステージにおいて、そのステージで目標として定められている解を求めることに努めることになる。

(4) エージェントが送受信する情報の形式：

通信情報の形式としては、問題解決全体を通して求められるべき値とそれらを求めるのに必要な要素の全てをリストにしたものを使用することになった。使用される値には実際の値の他に、そのステージで求めるべき値であることを示す特殊な値、そのステージでは推論に無関係な値であることを示す特殊な値などがある。また、通信情報は文字列として処理することにした。

(5) 四種類のメッセージを使用する：

他のエージェントへのメッセージは、以下の四種類の何れかの形態をとることにした。

「提案」：エージェントの推論結果及び中間結果で、その結果に対する評価を付加する。

「質問」：推論の続行が不可能と判断した場合に他のエージェントに援助を求めるときのメッセージで、どの部分が分からないために続行不能なのかを送信する。評価を付加する必要はない。

「応答」：「質問」に対する回答であり、質問の問題に対するそのエージェントなりの評価を送信するときのメッセージである。

「再評価」：その時のステージの目標の解であると思われる結果を得た時に、その結果の妥当性を他のエージェントに問うための形態で、その結果にそれに対する評価を付加して送信する。

(6) 定期的にプロトコルマネージャに対して解決すべき問題についての問い合わせを行う：

これは、エージェントが単独で推論できる限り推論し続けてしまえば、他のエージェントの影響を受けることが少なくなり、協調的システムの実現からは離れてしまう。したがって、エージェントは定期的にプロトコルマネージャに対して他のエージェントの送信が届いていないかの問い合わせを行うようにする。

(7) 推論の結果や中間結果は積極的に報告する：

エージェントが導いた結果や中間結果についての情報は他のエージェントにとっても有益な情報となり得るものである。したがって、エージェントは導いた結果や中間結果を他のエージェントに積極的に知らせるようにする。

(8) 推論の続行が不可能と判断した場合には他のエージェントに推論の続行を依頼する：

エージェントが推論を行った結果、知識の不足などの理由から目標の結果を得るに至らない場合が生じることもある。そのような場合には、推論の中間結果を他のエージェントに知らせ、推論の続行を促すことにする。

(9) 他のエージェントの推論結果や中間結果に対し固有の価値観で評価が可能である：

問題解決の過程においては、システム内には様々な局面が存在するが、最終的には一つの局面を解として選択しなければならない。したがって、エージェントは他のエージェントが導いた結果や中間結果に対して、そのエージェントなりの評価をつけることが可能である必要がある。

4．問題解決エージェントの概要

システム内の問題解決エージェントは、それぞれ推論を行う部分と、通信を行う部分及びエージェントの局所表現とシステムの共通表現との間で相互変換を行う部分から成っている。

推論を行う部分ではエージェントごとにそれぞれ異なった知識表現モデルを使用しているが、通信部と表現変換部の主な機能はおおむね同じである。

4.1. プロダクションエージェントの概要

(1) モジュール構成

プロダクションエージェントは図2のような構成をとっている。

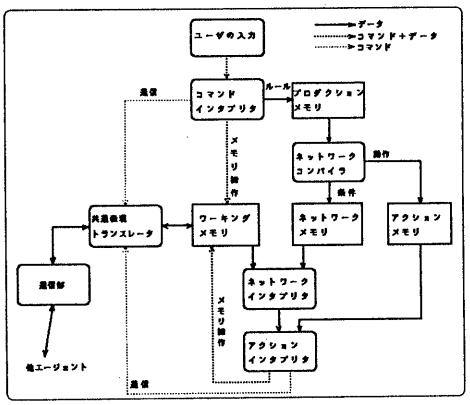

図2．プロダクションエージェントの構成図

a)コマンドインタプリタ

ユーザがプロダクションエージェントに対して初期設定及び各種の制御を行うために設けられたモジュールである。ユーザはこのモジュールにコマンドを与えることにより、ルールの定義、エージェントの状態の設定、入出力の制御、共通表現の定義などを行うことができる。アプリケーションプログラムはこれらのコマンドを用いて記述される。

b)ネットワークコンパイラ

プロダクションシステムが推論を行う際には、競合集合の作成及び競合解消の処理にかなりの時間がかかる。この処理時間を軽減するために、プロダクションエージェントでは競合集合の作成にＲＥＴＥのアルゴリズムを使用した。したがって、このモジュールではプロダクションルールを解釈し、ＲＥＴＥのアルゴリズムによりネットワークコードを生成する。

c)ネットワークインタプリタ

前述のネットワークコンパイラにより生成されたネットワークコードに従って、競合集合を作成し、競合解消を行うためのモジュールである。競合解消は、アプリケーションプログラムによって設定したルールの重要度をもとにして、ルールを選択することで行う。

d)アクションインタプリタ

成立したルールの行動部をアクションメモリから取り出し、解釈実行するためのモジュールである。成立するルールがない場合は、他のエージェントに中間解を送信する。

e)共通表現トランスレータ

プロダクションエージェントでは、情報の表現の変換の他に、他のエージェントに送信する推論結果に対する評価も行う。

f)通信部

エージェント間で推論結果の交換を行うための通信を行うモジュールである。

(2) プロダクションエージェントの推論動作

プロダクションエージェントはアプリケーションプログラムが起動されると、まず、問題を受け取るためにプロトコルマネージャに受信要求を送る。送られてきた問題は、ワーキングメモリに格納され、プロダクションエージェントの推論が始まる。

推論が始まってからの動作はおおむね一般のプロダクションシステムと同様であるが、認識行動サイクルが停止して推論の継続が不可能になった場合には、以下のような特別な動作を行う。

a)評価値を算出することができない場合：

ワーキングメモリの内容を共通表現形式に変換し、評価値は付加せずに「質問」であることを示す情報を付加して送信する。送信直後には他のエージェントからの情報を受信しようとする。

b)評価値を算出することができた場合：

ワーキングメモリの内容を共通表現形式に変換し、中間解の評価値と「提案」であることを示す

情報を付加して送信する。ただし、推論していた問題が「質問」だった場合、「応答」であることを示す情報を付加して送信する。

4.2.意味ネットワークエージェントの概要

(1) モジュール構成

意味ネットワークエージェントは図3のような構成をとっている。

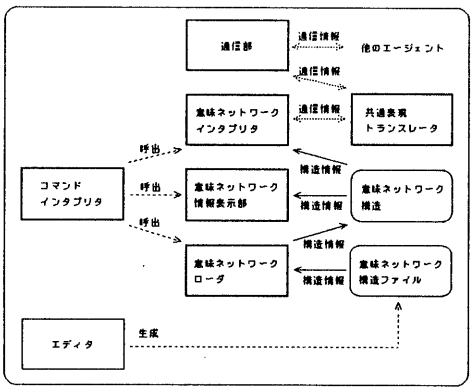

図3．意味ネットワークエージェントの構成図

a)コマンドインタプリタ

ユーザが意味ネットワークエージェントに対して処理の指示を行うためのモジュールであり、処理要求に応じて他のモジュールを呼び出す。

b)意味ネットワークローダ

ファイルに記述された意味ネットワークの構造を読み込むための処理を行うモジュールである。意味ネットワークの構造は二項関係の並びとしてファイルに記述されており、その記述を解釈しメモリ内に意味ネットワークのデータ構造を構成する。

c)意味ネットワーク情報表示部

メモリ内に構成されている意味ネットワークの各種の情報を要求に応じて表示するモジュールである。主に、存在するノードやアークの表示、ノードの周囲に存在するアークの表示などを行う。

d)意味ネットワークインタプリタ

メモリ内に構成されている意味ネットワークを探索することにより、ユーザの要求に応じた推論を行う。推論の方法としては、活性化伝播の方法を使用している。

e)共通表現トランスレータ

意味ネットワークエージェントでは、情報の表現の変換を行う他に、他のエージェントに対して情報を送信する場合に、定められている通信フォーマットに沿うように、通信内容を構成し直す。

f)通信部

エージェント間で推論結果の交換を行うための通信を行うモジュールである。

(2) 意味ネットワークエージェントの推論動作

意味ネットワークエージェントは協調型推論の開始命令を受けると、プロトコルマネージャに問題を要求する。問題を受け取ると、その問題に対する推論が始まる。

受け取った問題からその時のステージに応じて意味ネットワークの活性化の起点を選び出し、活性化伝播を行うことにより、推論を進める。

一つの活性化伝播が終了した時点での状態に応じて、以下のような動作を行う。

a)結果が得られなかった場合：

「質問」を示す情報を付加してその問題を送信し、すぐにプロトコルマネージャから問題を受け取ろうとする。

b)結果が得られた場合：

「提案」を示す情報を付加して得られた結果を送信する。ただし、推論した問題が「質問」だった場合には、「応答」を示す情報を付加して送信する。いずれの場合も、すぐにプロトコルマネージャから問題を受け取ろうとする。

4.3. フレームエージェントの概要

(1) モジュール構成

フレームエージェントは図4のような構成をとっている。

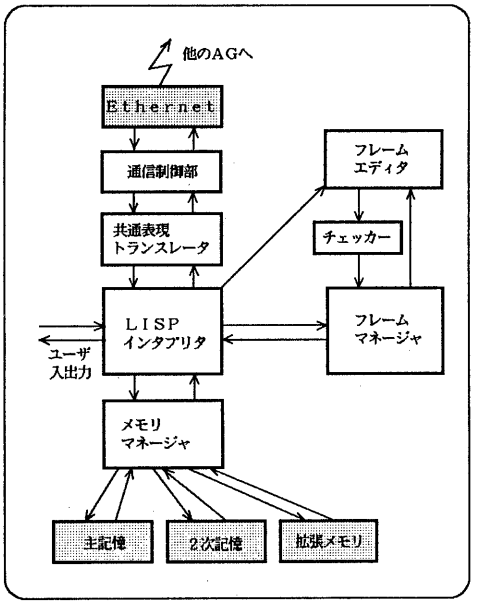

図4．フレームエージェントの構成図

a)ＬＩＳＰインタプリタ

フレームエージェントのコマンドインタプリタの役割を担うモジュールである。一般のＬＩＳＰインタプリタと比較すると組み込み関数の数は少ないが、フレームに関する関数、通信のための関数が組み込まれている。このＬＩＳＰインタプリタは付加手続きの処理も行う。したがって、アプリケーションプログラムと付加手続きを同一の言語で記述できるので、ユーザの負担を減らすことになる。

b)フレームマネージャ

フレーム名、スロット名の管理、及び付加手続き、性質継承の支援を行うモジュールである。フレーム名はともに一つのシンボルとしてシンボルテーブルに登録する。スロット名は各フレームがそれぞれ管理する。

c)メモリマネージャ

フレームエージェントが扱う記憶装置を管理するモジュールである。フレームエージェントが使用する記憶装置には主記憶、ＥＭＳ、補助記憶の三つがあり、大量のデータを扱う場合にデータをどの記憶装置上に格納するかという問題が生じる。このモジュールではそれを判断し、データに対して記憶装置の割り当てを行う。

d)フレームエディタ

フレーム、スロットの追加、更新、削除などは、ＬＩＳＰインタプリタに直接処理させることもできるが、一般ユーザ向けではない。したがって、このモジュールでは、容易に知識ベースの内容を編集できるような環境を提供する。

e)共通表現トランスレータ

前述のように、情報の表現の変換の処理を行うモジュールである。

f)通信制御部

エージェント間で推論結果の交換を行うための通信を行うモジュールである。

(2) フレームエージェントの推論動作

フレームエージェントの推論動作は、アプリケーションプログラムとして自由に記述できるが、今回の試作では、エージェントは推論を始める前に必ずプロトコルマネージャから問題を受け取ることになっている。

推論は未確定のスロット値を求めることにより行われる。推論が始まってからの動作はおおむね一般のフレームシステムと同様だが、以下のような場合には、特別な動作を行う。

a)推論結果が得られなかった、もしくは評価値が定められなかった場合：

推論した問題の内容に、「質問」を示す情報を付加して送信する。そして、すぐにプロトコルマネージャに問題の送信を要求する。

b)推論結果及び評価値が得られた場合：

推論した問題が「質問」だった場合には、「応答」を示す情報を付加して、「提案」だった場合には「提案」を示す情報を付加して、推論の結果を送信する。そして、すぐにプロトコルマネージャに問題の送信を要求する。

5．共通表現形式について

エージェント間で情報を交換する際の手段として、共通表現形式なるものを採用する事になった。

しかし、汎用の共通表現形式を定義するというのは、かなり大きな問題であり、今回の試作では、アプリケーションに依存した形式で情報を送信する際の書式を定めておき、それに従って送信を行うことにした。同時に、アプリケーションの問題解決を行うに当たって必要となる概念や事象などにシステム内で統一した呼称を与えることにしている。したがって、情報を送信する際の書式やシステム内の統一の呼称はアプリケーションの一部として記述され、エージェントに取り込まれることになっている。

今回の試作において検討したのは、問題解決の全てのステージで出てくる、値を求めるべき変数及びそれを求めるための推論に必要な情報の要素の全てに対して個別の記述場所を提供するような書式を設定するという方法である。この書式を使用する場合には、以下のような取り決めに従うことになる。

a)ある変数にある値が対応するということを記述するとき：

その変数の記述場所にその値を入れて送信する。

b)ある変数の値の確定を要求しているとき：

その変数の記述場所に値の確定要求を示す信号を入れる。

c)ある変数の値は無視して欲しいとき：

その変数の記述場所に値の無視を要求する信号を入れる。

この書式を用いているとき、確定要求の信号が全く存在しない情報は、そのステージでの中間解になっていることになる。したがって、確定要求信号を消去することが、即ち、中間解を求める動作になる。

プロダクションエージェントは、この書式をワーキングメモリの一パターンとして扱うようにアプリケーションを設計すれば、情報を容易に取り込むことができる。

意味ネットワークエージェントは、この書式を活性化の起点のリストとして扱うようなアプリケーションを設計すれば、情報を容易に推論へと結び付けることができる。

フレームエージェントは、この書式をＳ式として認識するように扱えば、ＬＩＳＰインタプリタで容易に情報を受け取ることができる。

このようにすれば三つのエージェントの何れにもかなり容易に情報を取り込めると考えた。

6．おわりに

今回試作したシステムでは、ハードウェアの制約上、プロトコルマネージャがエージェント間の全ての通信を集中管理しているが、プロトコルマネージャの機能に障害が発生した場合、エージェント間の通信は全て不可能になり、協調動作は行われなくなる。

したがって、今後の課題としては、プロトコルマネージャの処理も分散化して、その処理を各エージェントに吸収することが挙げられる。

参考文献

[1] E.H.Durfee, V.R.Lesser, D.D.Corkill: "Trends in Cooperative Distributed Problem Solving", IEEE transactions on knowledge and data engineering, VOL.1, NO.1, pp.63-83,

March(1989).
[2] Susan Leigh Star: "The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving", Distributed Artificial Intelligence, Chapter 2
[3] Charles L. Forgy: "Rete: A Fast Algorithm for the Many Pattern /Many Object Pattern Match Problem", Artificial Intelligence, Vol.19, (1982).
[4] 小林重信：「知識工学」，昭晃堂 (1986).
[5] 上野晴樹，石塚満：「知識の表現と利用」，オーム社 (1987).