ICOT Technical Memorandum: TM-1317

TM-1317

異種分散協調問題解決系Heliosにおける メッセージ通信の実現

和住　誠一郎、柳川　信晃（IMS）、
木島　秀治（IMS）

October, 1994

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Institute for New Generation Computer Technology

異種分散協調問題解決系 Helios におけるメッセージ通信の実現
Message Passing Mechanism in Helios

和住 誠一郎　　柳川 信晃†　　木島 秀治†
Seiichiro WAZUMI　Nobuaki YANAGAWA　Hidcharu KIJIMA
(財) 新世代コンピュータ技術開発機構
Institutc for New Generation Computer Technology
†(株) 情報数理研究所
Information and Mathematical Science Laboratory, Inc.

概要

異種分散協調問題解決系 Helios は、データベース、制約解消系、応用プログラムなど様々な問題解決器を組み合わせて使うことを支援するシステムである。このような問題解決系を構築するためには、個々の問題解決器が協調するために情報の交換を行う仕組みが必要になる。我々は、異種分散協調問題解決系を実現するため、問題解決器同士が通信するための機構を検討した。さらに、通信におけるボトルネックを防ぐための実装方法を検討し、その確認のため Helios 第1版を実装した。本稿では Helios における通信のメカニズムを中心に述べる。

1　はじめに

異種分散協調問題解決系 Helios は、データベース、制約解消系、応用プログラムなど様々な問題解決器を組み合わせて利用することを支援するシステムである。Helios では、分散協調を意識して作成した問題解決器の他に、既存の独立した問題解決器を利用することもできる [1][2]。このような問題解決系を実現するためには、問題解決器が互いに協調できるように、相互に情報を交換するための仕組みが必要である。

Helios では問題解決器間の通信をメッセージ通信を利用する。Helios では、組み合わせた問題解決器が協調するためのメッセージ通信を実現するため、「皮」と「環境」といった2つの仕組みを提供する。皮は、個々の問題解決器の内部表現と共通表現の間の変換を行い、同時に問題解決器が他の問題解決器に公開する処理をメソッドとして管理する。Helios では、問題解決器を皮でおおったものをエージェントと呼び、その問題解決器を実と呼ぶ。環境はエージェント間の通信を実現するための共通空間であり、通信に必要な情報や共通表現を管理する。

メッセージは他エージェントにメソッドの実行を依頼したり、依頼されたメソッドの実行結果を送ったりすることに用いられる。そのようなメッセージを他エージェントに送るためには、マシン情報やソケット番号といった相手のアドレスが必要になる。しかし、エージェントはどのような状況で問題解決系に組み込まれるかわからないので、他エージェントのアドレスをあらかじめ知っておくことはできない。そこでエージェントは、例えばエージェントの解くことができる問題の種類といった論理的な情報で相手を指定する。各エージェントは自分自身の論理的な情報を自己モデルとして管理する。環境は各エージェントの自己モデルを収集し、アドレスと論理的な情報の対応を管理する。環境は論理的な情報からアドレスを調べ、そのエージェントにメッセージを送る。

2 Heliosの概要

2.1 皮とエージェント

一般に、問題解決器は各々が異なる表現形式を持つ。これらの問題解決器がお互いに情報を交換するためには、問題解決器の間で共通に理解できる表現形式（共通表現）と、問題解決器から他の問題解決器へ情報を伝達してくれる共通空間が必要である。そこで、Heliosでは問題解決器の内部表現と共通表現を変換するための**皮**と呼ぶ機構で問題解決器をおおう。問題解決器を皮でおおったものを**単純エージェント**と呼び、その問題解決器を**実**と呼ぶ。同時に、皮は実が他エージェントに提供できる処理をエージェントの**メソッド**として管理し、エージェントの機能など論理的な特徴を**自己モデル**として管理する。

エージェントはメソッドの実行を他エージェントに依頼したり、メソッドの実行結果を渡したりすることをメッセージの交換で行う。メソッドの実行を依頼するために使うメッセージを**問合せメッセージ**と呼び、依頼されたメソッドの実行結果を渡すために使うメッセージを**回答メッセージ**と呼ぶ。

メッセージをあるエージェントに送るためには、送信先のアドレスが必要になる。そのようにエージェントの所在を示す情報を**エージェント識別子**（**識別子**）と呼ぶ。しかし、エージェントはどのような状況でHeliosに組み込まれるかわからないため、他エージェントの識別子をあらかじめ知っておくことはできない。そのため、エージェントはエージェントの特徴を示す**論理名**や、ある機能を持つエージェントの総称である**機能名**などで相手を指定する。そのような指定では、複数のエージェントが送信先として該当することもある。そこで、問合せ元のエージェントは、指定に該当するエージェントから送信先を選択する方法や、複数のエージェントからの回答メッセージを受け取る方法も指定する。これを**処理内容**と呼ぶ。論理名や、機能名、処理内容などでメッセージを送る相手を指定した情報を**宛名**と呼ぶ。

実の内部表現と共通表現の間の対応や、他エージェントに公開するメソッド、自己モデルなどは、皮記述言語CAPL (CAPsule Language)で記述する。

2.2 環境

環境は、エージェントからエージェントへのメッセージの伝達を実現するための共通空間であり、自分の中にいるエージェントにメッセージを送る機能を持つ。また、自分の中にいるエージェントに対して、通信のための共通表現の提供も行う。

環境はその中にいるエージェントも含めて1つの問題解決系とみなすことができるので、皮でおおって新たなエージェントとすることができる。そのようなエージェントを**複合エージェント**と呼ぶ。このように、Heliosではエージェントの階層的な組み合わせで問題解決系を構築することができる(図1)。環境に含まれるエージェントを、その環境の**直下のエージェント**と呼ぶ。直下のエージェントが複合エージェントならば、そのエージェント内の環境を**直下の環境**と呼ぶ。また、複合エージェントを含んでいる環境を、複合エージェント内の環境に対する**直上の環境**と呼ぶ。図1では、エージェント1とエージェント2が環境0の直下のエージェントであり、環境2は環境0の直下の環境である。また、環境0は環境2の直上の環境である。Heliosでは、ユーザを最上位の環境に対する直上の環境の位置におくため、ユーザからは構築された問題解決系が1つの複合エージェントのように見える。

環境は直下のエージェントの自己モデルから機能名やメソッドを収集し、同時に各エージェントに論理名を与える。環境は、各エージェントの論理名、機能名、メソッドを**エージェントサーバ**で管理する。エージェントサーバはそれらの情報を3つのディレクトリに格納する(表1)。

環境はエージェントサーバで得られた識別子にメッセージを送る。直下のエージェントに送信先がなかった場合、環境の外のエージェント（外部エージェント）には対応できるものがあるかもしれない。そこで、環境は直上の環境に問合せメッセージを渡す。最上位の環境の場合はユーザに渡す。図1では、環境2は環境0に問合せメッセージ渡し、環境0はユーザに渡す。

共通表現やエージェントサーバなどは環境記述言語ENVL (ENVironment Language)で記述する。

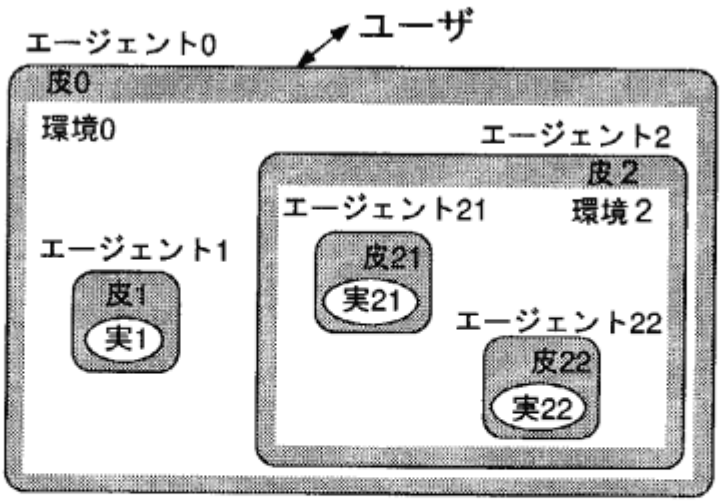

図 1: Helios のモデル

表 1: エージェントサーバが持つディレクトリ

ディレクトリ	情報
エージェント・ディレクトリ	論理名と識別子の対応
ファンクション・ディレクトリ	機能名と論理名のリストの対応
メソッド・ディレクトリ	[機能名, メソッド] の組と [論理名, メソッド] の組のリストの対応

3 Helios における通信

3.1 Helios におけるメッセージ

エージェントには、自分が解く問題の一部を他エージェントに依頼しなければならないものもある。そのようなエージェントは、他エージェントに依頼すべき問題の結果が必要になると、問合せメッセージで他エージェントに処理の実行を依頼する。

殻は表2から成るメッセージを生成する。エージェントは問合せメッセージの宛名を論理名や機能名などで指定し、依頼する処理をメソッドで指定する。環境は宛名に該当する直下のエージェントを送信先として選択し、問合せメッセージを送る。問合せメッセージを受け取ったエージェントはメソッドに対応する実の処理を実行し、結果を回答メッセージで送る。回答メッセージの送信先は問合せメッセージを送信したエージェント（問合せ元）の識別子で指定する。

論理名や機能名などで問合せメッセージの宛名を指定することは、Helios における通信の特徴である。宛名の指定と環境による該当エージェントの選択については 3.2で説明する。

3.2 宛名

問合せメッセージの宛名は以下の構文で記述する。

< 宛名 > ::= < 論理名 >
　　| < 機能名 > < 処理内容 >

表 2: メッセージを構成する情報

メッセージの構成要素	問合せメッセージ	回答メッセージ
メッセージタイプ	ask	reply
メッセージ識別子	メッセージの識別子	
宛名	論理名, 機能名など	エージェント識別子
送信元	エージェント識別子	エージェント識別子
トランザクション識別子	エージェント間のトランザクションを区別するための識別子	
ステータス		エラー情報
メソッド	依頼するメソッド	依頼されたメソッド
データ	メソッドへの入力データ	メソッドの実行結果

```
        | < 機能名 > < メソッド > < 処理内容 >
< 処理内容 > ::= first | bag_of | sequential
        | < 制約 > | < 選択基準 >
```

エージェントサーバは宛名に応じて以下に示すように各ディレクトリを使いわけ、該当するエージェント（該当エージェント）を求める。

1. **< 論理名 >**
   エージェント・ディレクトリから < 論理名 > に対応する識別子を得る。
2. **< 機能名 > < 処理内容 >**
   ファンクション・ディレクトリから < 機能名 > に対応する論理名のリストを得、リスト中の論理名に対応する識別子をエージェント・ディレクトリから求める。
3. **< 機能名 > < メソッド > < 処理内容 >**
   メソッド・ディレクトリから [< 機能名 >,< メソッド >] の組に対応する [論理名, メソッド] の組のリストを得る。そして、エージェント・ディレクトリからリスト中の論理名に対応する識別子を求める。メッセージを送信する時、問合せメッセージのメソッドをメソッド・ディレクトリから得たメソッドに置き換える。

2. と 3. の処理では、複数の該当エージェントが得られることがある。そこで環境は < 処理内容 > で指定された方法で、該当エージェントから送信先を選択し、問合せ元に回答メッセージを渡す。各々の < 処理内容 > に対する環境の処理を以下に示す。

first: 該当エージェントのリストの順に問合せメッセージの送信と回答メッセージの受け取りを逐次に行う。そしてメソッドの実行に成功した回答メッセージを問合せ元に送る。全ての該当エージェントがメソッドの実行に失敗した場合、問合せメッセージを直上の環境に送る。

bag_of: 全ての該当エージェントに問合せメッセージを送信する。そして、メソッドの実行に成功したエージェントからの結果だけをリストにまとめ、問合せ元に送る。環境は回答メッセージについてタイムアウトの検出を行う。すなわち、一定時間内に全てのエージェントからの回答メッセージが揃わなかった場合、それまでに受け取った回答メッセージの結果だけをまとめ問合せ元に送る。回答メッ

セージのステータスにはタイムアウトを検出したことを記す。環境はタイムアウト検出後に受け取った回答メッセージは破棄する。

sequential: 全ての該当エージェントに問合せメッセージを送信する。そして、メソッドの実行に成功したエージェントの回答メッセージを個別に問合せ元に送る。

＜制約＞: 全ての該当エージェントに問合せメッセージを送信する。そして、メソッドの実行に成功したエージェントの結果のうち、問合せ元が指定した＜制約＞を満たすものだけをリストにまとめ、回答メッセージで問合せ元に送る。各結果が＜制約＞を満たすか否かの検査は、＜制約＞を扱うことができるエージェントに環境が依頼する。

＜選択基準＞: 全ての該当エージェントに問合せメッセージを送信する。そして、メソッドの実行に成功したエージェントの結果から＜選択基準＞にしたがい問合せ元に返す結果を１つだけ選び、それを問合せ元に送る。複数の結果から＜選択基準＞にしたがい１つの結果を選ぶ作業は、＜選択基準＞を扱うことができるエージェントに環境が依頼する。＜選択基準＞は、例えばある式の値を最大にするものを選べ、というふうに与えられる。

4 Helios 第１版の実装

4.1 エージェントプロセス

Helios では、全てのメッセージは環境を経由して送られる。そのため、図1のような構成をナイーブに実装すると、環境にメッセージが集中するため、環境の処理がボトルネックとなる。そこで、単純エージェントを別々のプロセスで実現し、各々のプロセスにその単純エージェントを含む環境を重複させて持たせた(図2)。これにより、問合せメッセージの送信先の選択は問合せ元のエージェントと同じプロセス上にある環境でできるようになる。このようなプロセスを**エージェントプロセス (プロセス)** と呼ぶ。

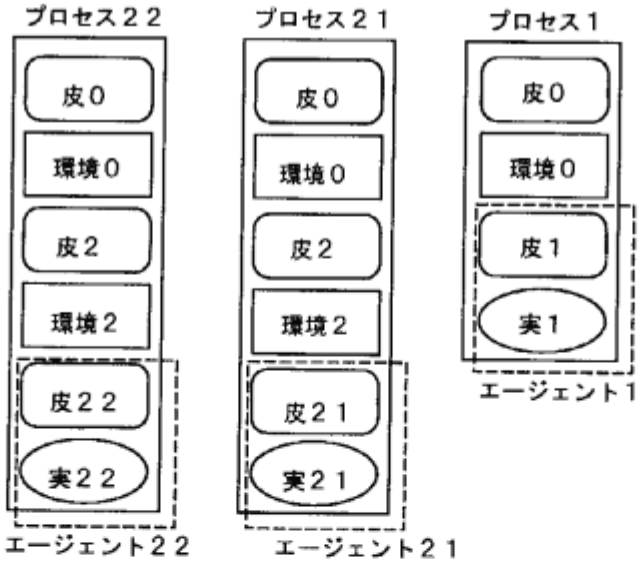

図 2: エージェントプロセス

4.2 エージェントプロセス間の通信

エージェントは問合せメッセージを送信する時、同じプロセス内の直上の環境にメッセージを渡す。環境が直上の環境にメッセージを渡す場合も、同じプロセス内の環境に渡す。環境が直下のエージェントにメッセージを送る場合、環境は送信先のいるプロセスへプロセス間通信でメッセージを送る。送信先が複合エージェントの場合、環境は複数のプロセスに重複しているため、送信先を含むプロセスは複数存在す

る。そのため、複合エージェントに対しては、送信先としてメッセージを受け取る**代表プロセス**を決めておき、環境は代表プロセスに問合せメッセージ送る。

直上の環境にメッセージを渡す必要がある場合のメッセージの流れの例と、複合エージェントにメッセージを送る場合のメッセージの流れの例を図3で示す。

直上の環境にメッセージを渡す必要がある例として、エージェント21からエージェント1に送られるメッセージをとりあげる。エージェント21が送信したメッセージは、プロセス21内の環境2を経由してプロセス21の環境0に渡される。そして、環境0からエージェント1へのメッセージの送信は、プロセス21とプロセス1の通信で行なわれる。

複合エージェントにメッセージを送る例として、エージェント1からエージェント22に送られるメッセージをとりあげる。ここでは、エージェント2の代表プロセスをプロセス21とする。エージェント1が送信するメッセージはプロセス1の環境0に渡される。環境0が送信先をエージェント2に決めると、プロセス1はエージェント2の代表プロセスであるプロセス21にメッセージを送る。プロセス21の環境2が送信先をエージェント22に決めると、プロセス21からプロセス22にメッセージが送られる。

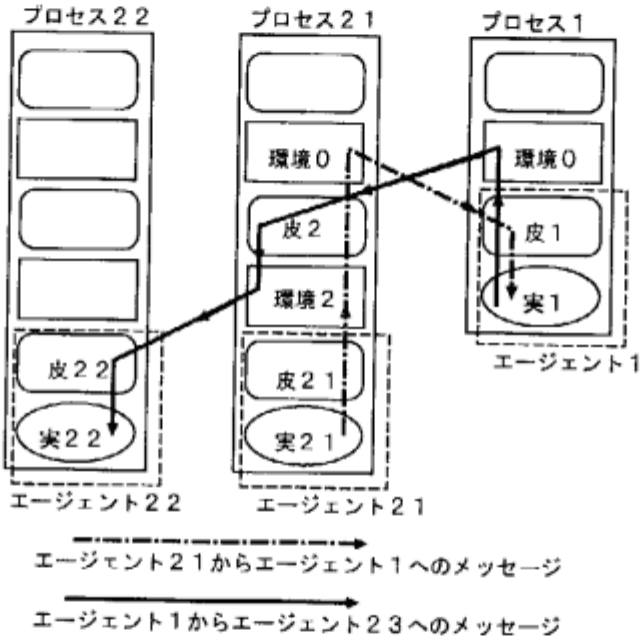

図 3: メッセージの流れ

4.3 第1版の構成

Helios第1版はエージェントプロセスによる通信を確認するための実験システムとして分散ネットワークトに実装した。

Helios第1版は**エージェントプロセス**の他に、**デーモン**、**ユーザインタフェース**から構成される。各々はUNIXのプロセスとして実装しており、通信はUNIXのソケットによるプロセス間通信で実現している(図4)。

ユーザインタフェースは、ユーザからの問合せをエージェントプロセスに渡したり、エージェントプロセスが生成した結果をユーザに提示したりする。

プロセス間の通信を実現するためには、単純エージェントのプロセスはどれか、複合エージェントの代表プロセスはどれか、通信経路を開くためのソケット番号は何かといった情報が必要になる。これらの情報は**デーモン**が管理する。デーモンは各マシン上に1つずつあり、各々のデーモンは自分がいるマシン上のプロセスの情報を管理する。プロセスは他のプロセスと通信する場合、自分がいるマシン上のデーモン

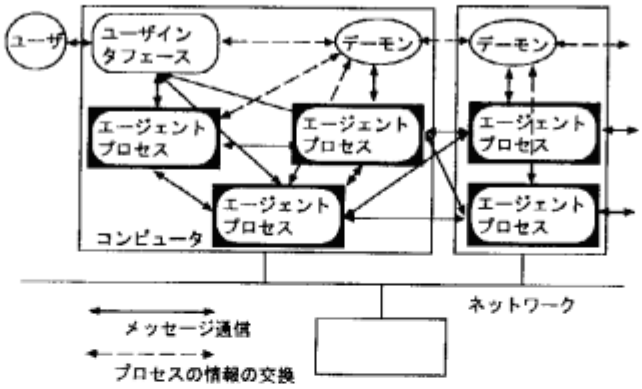

図 4: Helios 第 1 版の構成

に情報を要求し、それをもとに通信経路を開く。デーモンは、情報を要求されたプロセスが他のマシン上のものである場合、そのマシン上のデーモンに情報の提供を依頼する。

単純エージェントの実装には、実と皮を別のプロセスとするものと、実と皮を同じプロセスにするものが考えられる。例えば既存のデータベースを実として利用する場合は、皮と実は別のプロセスになる。また、ライブラリのように皮から呼び出される関数群として実が実装される場合、実と皮は同じプロセスになる。

第 1 版では、処理内容のうち first と bag_of だけを実装した。sequential、＜制約＞、＜選択基準＞に対する処理は bag_of に対する処理と大きく変わらないためである。また、環境の実装を単純にするため、環境は直上の環境から bag_of を受け取ると、bag_of を first に変更してから送信先の選択を行うようにした。

5 おわりに

我々は、異種の問題解決器の協調により問題解決を行う異種分散協調問題解決系 Helios について設計し、通信に関する機能を確認するため分散ネットワーク上の第 1 版を実装した。

現在、第 2 版において CAPL と、ENVL の実装を進めている。さらに、Helios の適用範囲を広げるため、*Quixote* や GDCC などの知識表現言語や Kappa などのデータベースを実として利用するための検討も行っている。また、Helios の応用として自然言語理解システムを作成している [4]。

エージェント間の協調は、Helios の重要なテーマであり、本稿で示した枠組上での実現について検討を行なっている。例えば、契約ネットプロトコル [3] のタスク提示や入札は、問合せメッセージの処理内容で制約や選択基準を指定することに対応する可能性がある。現在、本稿で示したエージェント間の通信にもとづくエージェント間の協調について検討を行っている。

参考文献

[1] 横田一正, 相場亮. マルチエージェントによる異種分散協調問題解決系の構想. In MACC'93, 1994.

[2] 横田一正. マルチエージェントによるマルチデータベースの拡張. 情報処理学会研究報告, Vol.94, No.62, pp.155-162, 1994.

[3] R.G.Smith. Reid G.Smith. The Contract Net Protocol:High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers, vol.29, no.12, pp.1104-1113, December,1980.

[4] 津田宏, 相場亮. 制約と分散協調に基づく自然言語理解システム. 日本ソフトウェア科学会第 11 回大会,1994.