データベースシステム 99−21
（１９９４．７．２１）

マルチエージェントによるマルチデータベースの拡張

横田 一正
(財) 新世代コンピュータ技術開発機構 (ICOT)
〒108 東京都港区三田 1-4-28 三田国際ビル 21F
kyokota@icot.or.jp

大規模知識情報処理の基盤システムとして研究開発している異種分散問題解決系 *Helios* は、データベースをエージェント (問題解決系) のひとつとすることにより、異種データベースを協調させる枠組を与えている。この点から、*Helios* はマルチデータベースの拡張と考えることができる。この特徴は、知識情報処理システムの応用を目的とした柔軟性にある。本稿では、*Helios* の基本モデルと機能を概説し、実装モデルを検討した後、それらをマルチデータベースの観点からデータベース機能を整理する。さらに *Helios* の応用として科学データベース・システムが有望であることを議論する。

Features of a Multi-Agent Based Multidatabase System

Kazumasa Yokota
Institute for New Generation Computer Technology (ICOT)
Mita-Kokusai Bldg. 21F, 1-4-28, Mita, Minato-ku, Tokyo 108, Japan

Helios, a heterogeneous, distributed, cooperative problem solving system which is being developed as an infrastructure for very large knowledge information processing applications, provides a cooperative framework of heterogeneous databases by taking a database as an agent (a problem solver). In the sense, *Helios* can be considered as an extension of multidatabase systems, for *Helios* has a capability of making heterogeneous databases working cooperatively. In this paper, we overview its basic model and important features, show its implementation model, and arrange them from a multidatabase point of view. Further, we discuss effectiveness of *Helios* for various scientific database systems.

1 はじめに

Quixote [12] は、知識情報処理の応用を対象に研究開発してきた演繹オブジェクト指向データベース言語 / システムである。その有効性を検証するために、これまでその上に自然言語処理、遺伝子情報処理、法的推論などの応用システムのプロトタイプを開発してきた [11]。しかし、さらに大規模な応用システムを構築するために、法的推論での経験を踏まえて文献 [14] では必要な拡張機能を次のように整理した。

1) 不完全情報 & 部分情報の管理
2) 異種・大規模データベース / 知識ベースの管理
3) データや知識の貯蔵庫から思考実験の場の構築
4) データベース機能の解体・再生 (他機能との統合)

これらは法律データベース (社会科学) に限らず、ゲノムデータベース (自然科学) を含む科学データベースの共通の特性 (データの膨大さ、構造の複雑性、データの曖昧性、データの不完全性、種類の多様性、個人データの重要性) から来る、共通の要件と考えている (5 節参照)。

それら要件を反映させるために現在次の 2 つの研究開発を進めている。

- *Quixote* の機能強化 ((非) 等式制約、外部呼出、…)
- 異種分散問題解決系 *Helios* [15] の研究開発

Quixote は、1) については、論理的オブジェクト識別子、モジュール階層、否定 (オブジェクトの非存在、非等式制約) によって、3) については、仮説生成 (アブダクション) と仮説推論によってサポートしているが、さらに外部機能の呼出しによって 4) の一部に応えようとしている。

Helios は上記の要件 2), 4) に対応したものであり、データベースを含む種々の問題解決系の包括的な協調システムの枠組の実現を目指している。*Quixote* は *Helios* での重要な構成要素のひとつと位置づけられており、さらに *Helios* によって ICOT で研究開発したさまざまな言語や制約解消系との協調が期待されている。*Helios* の異種問題解決系という枠組は、データベースをひとつのエージェント (問題解決器) としてとらえることにより異種データベース間の協調を可能にしており、マルチデータベース [1, 5, 9] の拡張としてもとらえられる。本稿では、その立場で、まず *Helios* の基本モデルと機能を紹介し、それらをデータベースの観点から議論する。

本稿の構成は次のようになっている。2 節では異種分散協調問題解決系としての Hellios へ基本モデルを述べた後、2 種類の言語とそれに基づく機能を概説する。3 節では *Helios* の実現方式について述べ、4 節ではマルチデータベースとしての機能を整理し関連研究との比較を行なう。5 節では *Helios* の応用として考えている科学データベースへとの関連について簡単に触れる。

2 異種問題解決系としての *Helios*

2.1 基本モデル

Quixote は、知識処理向きのデータベースと自然言語処理 (状況理論) の知識表現言語などの実現を動機に研究開発した言語 / システムであるが、*Helios* はさらに異種制約解消系の取込み、法的推論や遺伝子情報処理などの科学データベースへの対処などの応用からの強い動機を基に研究開発を行なっている。データベースに焦点を限定し、1節での要件を具体化すると次の 3 つに分けることができる。

外部呼出機能

問合せ処理中に算術演算、代数制約、情報検索、外部データベースなどの外部機能を自由に利用できる。この外部機能には応用モジュールも含まれており、並列データベースでの並列効果の向上 [11]、検索機能の強化 (数式データベースの AC マッチング、ゲノムデータベースのホモロジー検索) などにも有効である。

異種 / 大規模データベース

異種データベースの結合による大規模データベースの構築のための枠組が必要である。平坦な構造ではなく、ボトムアップな階層的な構築が望ましい。たとえば配列情報や構造情報など既存のデータベースの積み上げが必要なゲノムデータベースはこの典型である。

データベース間の協調

ひとつの問合せを処理するために複数のデータベースの処理結果をマージさせたり、互いの処理を制御しあったりする枠組が必要となる。法律データベースでのルールの選択的適用やスケジュールデータベースでの制約充足問題がある。これはひとつの複合問題を制約論理型言語のスキームで解くという分散問題解決 [13] を包摂した枠組となる。

これらを実現するために次のような基本モデルを設定した (図 1参照)。

- データベース、制約解消系、応用プログラムなどの各機能間に主従関係はつけない。つまり主従関係は応用によって変化するので、それらをすべて**問題解決器**として

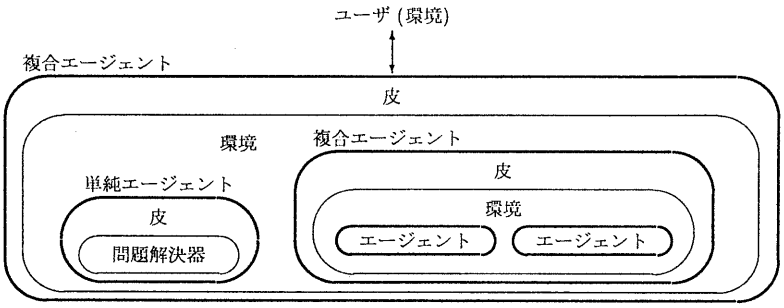

図 1: *Helios* の基本モデル

扱う。粒度に関しては何の制限もなく、問題解決器間の効率上の問題が設計上の考慮の対象となるだけである。したがって、大きなデータベース管理システムも小さな制約解消系も対等な問題解決器にすることができる。

- 問題解決器はカプセル化し、メッセージによって必要なメソッドを起動し処理を行なわせることにする。問題解決器の集まりを新たな問題解決器として定義する必要があるので、カプセル化した問題解決器はオブジェクトではなく**エージェント**とする。カプセル化するためのモジュールを**皮**という。

- エージェントが共存する場を**環境**といい、ここを流れるメッセージはすべて強い型づけがされている。そのための共通型システムは環境毎に定義可能で、各問題解決器の固有の型システムと共通型システムとの変換は皮が行なう。データベース問合せを含むメッセージは**制約**として一般化する。

- 環境とそこに共存する複数のエージェントをカプセル化することにより、新しくひとつのエージェントとして定義することができる。これを**複合**エージェントといい、内部構造をもたないものを**単純**エージェントという。

- エージェントには、外部から送られた問題を解決することと内部で解決できない問題を外部に問い合わせることの２つの機能がある。前者の機能しか持たないものを**受動**エージェント、両者の機能を持っているものを**能動**エージェントと呼ぶ。ほとんどのプログラムは受動エージェントになりうるが、能動エージェントになるためには問題解決器自体の機能拡張が必要となる。

- 環境は各エージェントの情報を収集し、エージェント間のメッセージの配信について責任を持つ。つまり能動エージェントが外部に投げたメッセージ、あるいは外部環境から皮を通して送られたメッセージは、すべて環境に送られ、環境はそれを解決する可能性のあるエージェントに送信する。もしエージェントを発見できなければ、その環境の皮を通して外部環境に問題を送る。

- **ユーザ**は前項の意味でエージェントのもっとも外の環境として位置づけられるが、さらに内部のエージェント(問題解決器) をユーザとすることもできる。つまりユーザによるラピッド・プロトタイピングやグループウエアの実現の枠組にもなりうる。

- エージェント間のメッセージのやりとりには様々なパタンがある。これら協調や交渉の方略については、またエージェントの行動範囲を制限したり情報を共有するための大域制約についても環境が責任を持つ。

当初 *Quixote* の拡張としてこの基本モデルの実現を計画していたが、型の扱い、協調方略、複合エージェントの構成など未解決の問題が多いので、*Quixote* の拡張としてではなく新たに *Helios* として上記の基本モデルを構築することにした。

2.2 *Helios* の言語と機能

Helios で使用される**メッセージ・プロトコル**は環境から独立したもので、次の５つ組になっている。

(メッセージID、トランザクションID、
送付元エージェントID、送付先エージェントID、
メッセージ)

これらは次のような内容を持っている。

- メッセージID は皮によって各メッセージに付加される一意的な内部識別子である。

- トランザクションID は更新を伴うメッセージの制御を行なうために内部的に付加されるトランザクションの内

表 1: CAPL と ENVL での定義項目

CAPL	ENVL
エージェント名	環境内のエージェント・リスト
輸入メソッド / 輸出メソッド	共通型システム
自己モデル	大域制約
ユーザモデル	協調 / 交渉方略
同時実行制御の方式	オントロジー
データ変換 (構文、型、変数環境、データ構造)	パラメータ付きエージェントのテンプレート

部識別子である。入れ子トランザクションはIDを階層構造としている。

- エージェントIDは、ユーザが与えるエージェント名とは別に内部的にエージェントを一意的に識別するものである。更新メッセージの場合、あるトランザクションを開始した送付元エージェントIDがそのトランザクションのユーザIDとなる。送付先エージェントIDには、エージェントIDの集合やエージェント名や(エージェントを指定しない)機能名を指定することもできる。
- メッセージにはメッセージ (制約) 自体の他に メッセージの種別や (全解探索などの) 処理モードの指定が含まれている。さらに環境でメッセージが複数の部分メッセージに分解されたときの (結果の同期情報を含む) 実行プラン情報もここで指定される。

環境依存の情報は2.1節で述べた基本モデルを実現するために2種類の言語を持っている。

皮言語 (**CAPL**; capsule language)
環境言語 (**ENVL**; environment language)

これら言語で指定する情報は表1の通りである。

2つの言語での指定項目は主として次のように関連して、エージェントの機能を実現している。

- CAPL のエージェント名が ENVL のエージェント・リストに書かれ、環境内のエージェントが特定される。ただし環境内に類似のエージェントが多い場合には、ENVL でパラメータ付エージェントとして指定することが可能で、それらは必要に応じて動的に生成される。
- 環境は各エージェントの自己モデルと輸入メソッドによって各エージェントの機能を知ることができ、この情報によってメッセージのエージェント間の配信する。具体的にはエージェント名、メソッド情報、機能名の3種類による配信が可能になる。ただし自己モデルは各エージェントの自己申告なので実行時の修正が必要となる。
- ENVL で指定された共通型定義を各エージェントの皮は取り込み、各問題解決器に固有の型と共通型の変換ルーチンをメソッド毎に生成する。

CAPL と ENVL によって定義される皮と環境機能を整理すると次にようになる。

皮の機能

* (外部) 環境 (あるいはユーザ) から送られたメッセージを輸入メソッド定義にしたがって解析・変換し、(内部) 環境あるいは内部問題解決器に送る。

* (内部) 環境あるいは内部問題解決器から送られてきたメッセージは輸出メソッド定義にしたがい解析・変換し、(外部) 環境に送る。外部環境がユーザであれば、ユーザモデルにしたがったメッセージを送付する。

* 実行プラン付きのメッセージであれば、必要なら他エージェントからの解の待合せ (同期) を行ない、解のマージを行なう。

* 逐次実行の機能しか持っていない問題解決系に複数のメッセージの処理を行なわせたいとき、この皮で施錠 / 解錠を行なったり、必要なら問題解決器自体の複製を行ない、同時実行を制御する。

環境の機能

* 送付されたメッセージの送付先によって、必要なエージェントIDを探す。機能名、メソッド名、エージェント名から探すことが可能。見つけることができなければこの環境の皮 (から外部の環境) に送る。

* 送付元と送付先の関係により、必要ならオントロジーに

よりメッセージを変換する。

* 処理モードにしたがい、メッセージのエージェントへの送付を制御する。全解探索ならば見つけたすべてのエージェントに送る。契約ネットの場合は必要なメッセージを対象エージェントに送付しエージェントを選択する。

* 返ってきた解の評価、選択、マージなどを行ない、また必要ならばオントロジーで変換し、メッセージの送付元に送る。一定時間を経過しても返ってこないメッセージはタイムアウトする。返ってきた解が必要な基準を満たしていなければ、あるいはタイムアウトされると、必要に応じて代替エージェントにメッセージを送る。

* 送付されたメッセージが部分メッセージに分割可能で、実行プランが作成できるならば、それにしたがったメッセージ群を作成する。

* 環境内に存在すべきエージェントを要求駆動で立ち上げ、探索情報などの必要情報を初期/再設定する。パラメータ付エージェントであればパラメータによって必要個数を立ち上げる。

3 *Helios* の分散実装モデル

前節に述べた *Helios* の論理モデルをそのまま実装するには処理効率上の問題がある。つまり

1) 環境への処理の集中の緩和
2) 階層的構成のプロセス環境への写像
3) 分散環境への写像

などの問題である。1) は環境の複製、2) は通信負荷を減らすためにプロセス数を減少させるような写像、3) はエージェントとプロセス情報の分散管理を考えなければならない。*Helios* の実験システムの評価からとくに 2) を設計に重点的に反映させることにした。

Helios 1版の実装モデルは図2のような構成になる。これは図1を簡略化し、図1の左の問題解決器を1、右下の2つのエージェントを問題解決器2、3としている。

プロセスの種類は基本的には次の3種類がある。

エージェント・プロセス
問題解決器は論理的には階層構造であるが、ここでは内部構造を持たない問題解決器にその上層の皮と環境を付加して平坦な構造としている。環境が複製されるために、それらに一貫性を持たせる制御が必要となる。

デーモン・プロセス
分散環境での各マシン上には *Helios* デーモン・プロセスがひとつ存在し、そのマシン上でのエージェント・プロセスの生成・消滅を管理し、他のプロセスにエージェント・プロセスの所在情報を提供する。分散環境ではデーモン間の通信が必要となる。

インタフェース・プロセス
ユーザごとに生成され、ユーザとデーモン・プロセスとの対話 (デーモン・セッション)、およびユーザとエージェント・プロセスとの対話 (エージェント・セッション) を司る。デーモン・セッションでは、エージェントの一覧表示、指定したエージェントの詳細情報の表示、エージェントとの接続などが可能である。エージェント・セッションでは、対象エージェントに対するメッセージの送付の他に、エージェント情報の表示、エージェントとの接続の切断などができる。

この他に、問題解決器自身がデータベース管理システムのように独立したプロセスで動く場合、それは**問題解決器プロセス**として位置づけられ、エージェント・プロセスと問題解決器プロセスとは別プロセスとなる (たとえば図2の問題解決器3)。

論理モデルをこの実装モデルに写像するために、次のような考慮点が必要となる。

1) 論理的にひとつのエージェントが複数のプロセスに分割されるために、メッセージをどのプロセスに送るかの問題がある。プロセスの負荷に応じたスケジューリング、メッセージ内容に応じたディスパッチなども考えられるが、現在の実装では環境ごとに代表エージェントをプロセス立ち上げ時に決めている。これは実装の容易さからである。

2) 論理モデルではメッセージの経路は、非決定的な処理を除き、一意的に決めることができた。現在の実装では、問題解決器までの経路は代表エージェントの経路を辿るが、逆は同一プロセス内の皮を上に辿ることによってメッセージの逆変換を行なうことによりプロセス間通信を減少させている。この逆経路は分散実装の利点となっている。

3) 環境ごとの動的情報が複製されることによる一貫性の保持のために、プロセスの各層 (図2 での環境123と環境23) に対応した2相コミット・プロトコルを設定して、一貫性を保っている。環境内の動的な情報は、各プロセスで分割して管理することが可能なものも多いので、この複製による効率低下は大きくないと判断した。

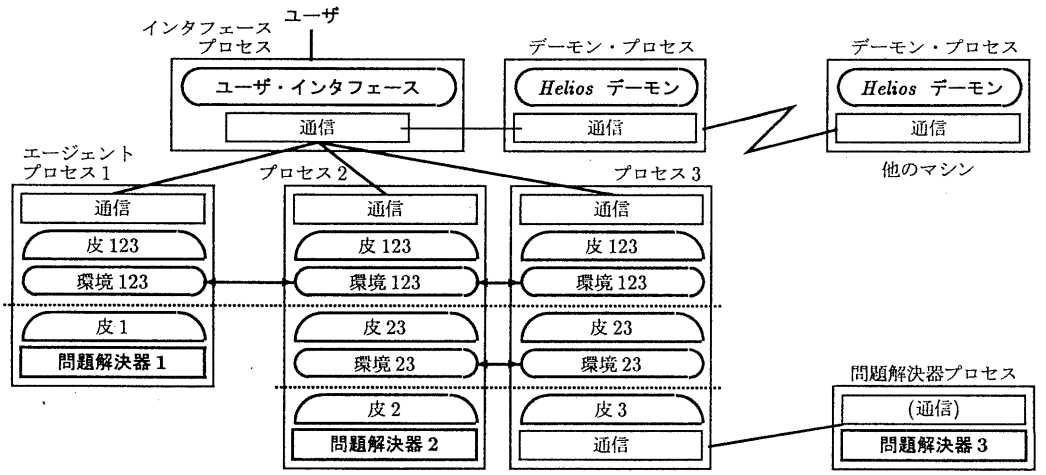

図 2: ***Helios*** の実装モデル (図 1 に対応)

4 マルチデータベースとしての *Helios*

本節では、*Helios* と関連研究との比較を行ないながら、その機能をデータベースの観点から整理したい。

4.1 *Helios* の位置づけ

分散環境で複数の自律性を持った異種データベースを含むシステムを構成するために、データベース間の関係を以下の 3 種類に分類することが多い (たとえば文献 [6, 7])。

1) 相互接続性 (interconnectivity)
2) 相互運用可能性 (interoperability)
3) 協調 (cooperation) または知的相互運用可能性

マルチデータベースはさまざまなシステムがある (たとえば [10] の分類図) が、これら 3 つの通信レベルを包摂したものと考えられる。自律性の強いデータベース同士のシステムを考えれば 3) の必要性が高くなってくるが、大規模分散環境ではこの要求が高い。そのために分散人工知能 (とくにマルチエージェント) の技術を取り込むことが行なわれている (たとえば上記 [6, 7] の他にも [5, 9] にも多くある)。前節までに述べた異種問題解決系としての *Helios* も、対象を必ずしもデータベースに限定していないが、この流れに沿ったものである。

分散環境で異種のデータやプログラムをオブジェクトとしてカプセル化し利用しようとするものに分散オブジェクト管理システム [6] があり、分散データベース・システムの次世代システムとして期待されている [3] が、*Helios* は次の点でさらに対象を拡げている。

- メッセージの内容の異種性を解消する機能がある。
- ボトムアップにエージェントの階層構造が構築できる。
- 制約充足問題のような協調 / 交渉戦略が組み込める。

また Papazoglu たち [7, 8] は、分散人工知能と分散データベースを融合することによって知的情報システム (共有永続情報の管理システム) を協調させる枠組を提案している。つまり分散データベースの各ノードを能動化し、協調する知的エージェントの集まりとの枠組を持たせようとしている。*Helios* とは以下の点で異なっている。

- 対象とするエージェントの粒度が大きく、新たなエージェントをボトムアップに構築するという視点がない。つまり単一環境である。
- 大域制約を持つことができない。

Helios では科学データベースのような大規模知識ベースの構築や分散人工知能固有のでの問題解決も取り込もうとしている点で、上記システム [6, 7] に比べ枠組が拡がっている。これはまたデータベース・システム自体の外部機能呼出による機能拡張にも有利に働くと考えられる。

加藤 & 大堀 [4] は共通の永続関数型言語を設定することによって異種のデータベース・プログラミング言語の統合を目指しているが、*Helios* では以下の点が異なっている。

- 対象言語に制限を設けていない。つまりメソッド・プロトコルでの型変換だけである。

- 協調を含むエージェント間の関係を問題解決器と独立に定義できる。

Helios のように対象を拡張することによる欠点として、意味論の定義の困難さが問題として残っている。

4.2 *Helios* のデータベース機能

ここではマルチデータベースで必要となる機能が *Helios* でいかに実現されているかを整理する。

オブジェクト識別子

エージェント間をまたがるオブジェクトのオブジェクト識別子はすべて論理的オブジェクト識別子を使用し、その名前づけの違いは環境のオントロジーで吸収する。

データ構造とデータ操作

各データベースはエージェントとしてカプセル化されており、メソッドを通してのみデータにアクセスできる。伝統的な意味での分散データベースは論理的にはひとつのデータベース管理システムであるので、これはひとつのエージェントとなる。輸出入メソッドとも

共通メソッド名@共通シグナチャ
⇐ 固有メソッド名@固有シグナチャ | 変換情報

と定義されている。シグナチャは $\tau_1 \times \cdots \times \tau_n \rightarrow \tau$ ($\rightarrow \tau$ は省略可能) の形式指定可能で、固有シグナチャが内部データベースのビュー (の操作) に対応する。

一貫性制約

複数のエージェントにまたがるデータの一貫性制約については、環境の大域制約として記述され、必要に応じて各エージェントが参照する。

更新

トランザクション論理と同時実行制御の機能は問題解決器によって異なっているので、この異種性を制御するための機能が皮と環境に付加されている。

- トランザクション制御を行なっていない、副作用のある問題解決器で、実行上過負荷にならない場合には、皮が問題解決器の前像のコピーを生成し実行制御することにより、入れ子トランザクションの制御を可能にしている。
- トランザクション論理の異なった問題解決器が混在した場合には、もっとも弱いものに合わせる。したがってこれが原因で実行時エラーも生じうる。
- 複製された環境の大域制約の更新を含め、2相コミット・プロトコルを共通に採用する。

また同時実行制御については次のような制御を行なう。

- 副作用のない問題解決器で、実行上過負荷にならない場合には、環境が必要に応じてエージェントのコピーを行ない、複数のメッセージの処理を行なう。
- 同時実行ができないエージェントが複数のメッセージを受け付けた場合、皮が逐次実行の制御を行なう。
- デッドロックの検出、回復は環境が責任を持つ。

このような機能を環境と皮で分担するために、皮は問題解決器の能力についての自己モデルを持ち、エージェント立ち上げ時に環境に必要情報を伝える。

問合せ

Helios は制約解消系をもエージェントとして定義するために、メッセージも制約として一般化している。

- データベースの問合せを制約と考えると、過制約 (over-constraint) のときに解が得られず、少制約 (under-constraint) のときに解が得られることになる。過制約のとき、制約緩和の方略 (類似語や上位語を使用、制約数の減少、…) が環境に登録されていれば、自動的に緩和が行なわれ問合せが再実行される。
- 制約集合としての問合せが環境に与えられたとき、メッセージのディスパッチ情報を利用して問合せの実行プランを生成する。部分問合せ間の解の合成や実行の同期については皮が責任を持つ。
- 問合せ結果に対する応用のフィルターは、環境の解の評価機能として登録することが可能で、そのためのエージェントが自動生成される。

5 科学データベースへの応用

科学データベース (scientific database)[2] とは、一言でいって科学者が利用するデータベースのことである。データベースの概念を広義に捉えれば、多くの文献データベースが情報検索で利用されてきたが、ここでいう科学データベースとは、文献データベースとは異なり、科学の研究の対象として用いられるデータを集めたものを意味している。科学データベースとして、分子生物学 (遺伝子、タンパク質)、化学 (化合物、化学反応)、地球科学 (気象、地質、地震、環境)、宇宙科学 (天文、衛星)、医学 (疫病、臨床例)、薬学 (薬品、毒性)、など数多くのものが対象となっている。

1節で述べたように *Quixote* では多くの応用を対象としてきたが、その中で法的推論での法律データベースも、遺伝子データベースと同じ特徴を持っていること、つまり科学データベースと多くの共通点を持っているを発見した。つまり科学データベースは、自然科学のみならず、社会科学や人文科学でのデータベースも含めて、多くの応用の共通の枠組でありうると考えている。

このような科学データベースでは、従来のデータベースのように実世界のきれいな側面だけを切り出すことは難しく、むしろ雑然としているところにこそ存在価値を持っているということもできる。したがって、データの種類の他にデータの抽象階層を考えれば、これらを単一のデータモデルで管理するのは困難で、*Quixote* での遺伝子情報処理、法的推論、自然言語処理などの応用の経験から *Helios* のような異種分散協調の枠組が必須と考えている。

6 おわりに

本稿では、異種分散協調問題解決系として現在研究開発している *Helios* がマルチデータベースの拡張として考えられることを述べた。現在 C 言語で実装している *Helios* は、効率評価のための実験的な実装 (1994 年 3 月) の後、基本機能を持った第 1 版を 1994 年 7 月末、協調機能を強化した第 2 版を 1994 年 11 月末に完成させる予定である。開発期間が限られているため、本稿で述べたマルチデータベース機能がどこまで実装できるかは未定であるが、このシステムは、種々の問題解決器を協調させる枠組であると共に、類似のシステムの実験環境ともなりうる。したがって、その後は異種分散協調環境のツールと位置づけ、1995 年 3 月までにフリー・ソフトウェアとしてリリースする予定である。

謝辞

限られた短い期間でシステムの設計と実装を精力的に行なっている *Helios* グループのメンバーに感謝する。

参考文献

[1] A.K. Elmagarmid and C. Pu (eds), Special Issue on Heterogeneous Databases, *ACM Computing Surveys*, vol.22, no.3, 1990.

[2] IEEE Computer Society: *Special Issue on Scientific Databases, Bulletin of the Technical Committee on Data Engineering*, vol.16, no.1, Mar., 1993.

[3] 情報処理学会データベースシステム研究会, "データベース 2001 年に向けて — 今後 10 年、何を研究・開発すべきか —", *92-DBS-90*, 1992.

[4] K. Kato and A. Ohori, "An Approach to Multilanguage Persistent Type System", *Proc. IEEE 25th Hawaii Int. Conf. on System Sciences*, pp.810-819, Jan., 1992.

[5] Y. Kambayashi, M. Rusinkiewicz, and A. Sheth, *Proceedings of First International Workshop on Interoperability in Multidatabase Systems*, IEEE Computer Society, Kyoto, 1991.

[6] F. Manola, S. Heiler, D. Georgakopoulos, M. Hornick, and M. Brodie, "Distributed Object Management", *Int. J. of Intelligent and Cooperative Information Systems, vol.1., no.1, pp.5-42*, 1992.

[7] M.P. Papazoglu, S.C. Laufmann, and T.K. Sellis, "An Organizational Framework for Cooperating Intelligent Systems", *Int. J. of Intelligent and Cooperative Information Systems, vol.1., no.1, pp.169-202*, 1992.

[8] M.P. Papazoglu, "On the Duality of Distributed Database and Distributed AI Systems", Proc. CIKM, Washington DC, 1993.

[9] H.-J. Shek, A.P. Sheth, and B.D. Czejdo (eds.), *Proceedings of Second International Workshop on Interoperability in Multidatabase Systems*, IEEE Computer Society, Vienna, 1993.

[10] A.P. Sheth and J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", in [1].

[11] K. Yokota and H. Yasukawa, "Towards an Integrated Knowledge-Base Management System — Overview of R&D on Databases and Knowledge-Bases in the FGCS Project", *Proc. FGCS'92*, Tokyo, June 1-5, 1992.

[12] K. Yokota, H. Tsuda, and Y. Morita, "Specific Features of a Deductive Object-Oriented Database Language *Quixote*", *Proc. ACM SIGMOD Workshop on Combining Declarative and Object-Oriented Databases*, Washington DC, USA, May 29, 1993.

[13] 横田一正, "オブジェクトの形式化とスキーム", 電子情報通信学会データ工学研究会, DE88-25, Nov.24, 1988.

[14] 横田一正, 柴崎真人, "データベースに判決は予測できるか?", 情報処理学会データベースシステム研究会 & 電子情報通信学会データ工学研究会合同ワークショップ (EDWIN), July 21-23, 1993.

[15] 横田一正, 相場亮, "マルチエージェントによる異種問題解決系の構想", *MACC'93*, 穂高, Dec. 15-17, 1993.
(修正版が, 奥乃博 (編), 『マルチエージェントと協調計算 III』, 近代科学社, 1994 に収録.)