データベースシステム 104−35
（1995. 7. 20）

ネットワーク上の仮想データベースシステム

横田 一正　木實 新一
京都大学 大学院 工学研究科 情報工学研究専攻
〒606-01 京都市左京区吉田本町
{yokota,konomi}@kuis.kyoto-u.ac.jp

インターネットなどの広域ネットワークに拡がるさまざまな情報源を利用するためには、関連する情報源を効率的に探索するだけでなく、より有効な結果を求めるために複数の情報源を統合する必要がある。本稿では関連する複数の情報源の (緩やかに統合された) 集合を仮想データベースと呼ぶ。本稿では、異種分散協調問題解決系として研究開発されている *Helios* を仮想データベース構築の枠組として用いる。広域ネットワーク上の情報源はつねに変化しているので仮想データベースもそれに対応しなければならない。また従来のデータベースが前提にした閉世界仮説の成立しない開世界であるので、問合せの結果は必ずしも完全なものではない。本稿では、広域ネットワークのために提案されている多くの情報源探索ツールなどと比較しながら、仮想データベースの意義を検討する。

Virtual Database Systems on Networks

Kazumasa Yokota　Shinichi Konomi
Department of Information Science, Kyoto University
Yoshida-Honmachi, Sakyo, Kyoto 606-01, Japan

To utilize various information resources dispersed in wide-area networks such as the Internet, it is important not only to look for the related resources efficiently but also to integrate them for obtaining more effective answers. In this paper, we call a set of loosely coupled information resources *a virtual database*. In this paper, we take a heterogeneous distributed cooperative problem solver, *Helios*, as the framework for constructing virtual databases. As information resources are always varying in wide-area networks, virtual databases are also dynamic. Further, although conventional databases have used the closed world assumption, virtual databases are based on open worlds and their query processing should be different from conventional ones. In this paper, we discuss the significance of virtual databases, comparing with various softwares proposed for information retrieval tools in wide-area networks.

1 はじめに

インターネットの急速な拡大と利用者 / 情報発信者の増大によって、多種多様な情報を従来に比べると簡単に利用することができるようになった。しかし反面、データ間の重複や矛盾など制御しにくい問題点がますます大きくなっている [3]。

必要な情報を得るためには、その所在を探すだけではなく、複数の情報源の情報をまとめる必要がしばしばある。必要な情報源の所在情報を得るためには、

- 関連のディレクトリを探す、あるいは
- 直接関連しそうなエージェントに聞く

ことが必要である。複数の情報源に問合せなければならないのは、

- ひとつの情報源だけからでは必要な情報が得られない, あるいは
- 複数の情報源からの結果をまとめることによって、ひとつだけの情報源からの結果より良い結果を得ることができる

からである。後者の場合には、量的な向上 (たとえば問合せを満足する情報量の増加) と質的な向上 (たとえば解の詳細化や洗練化) が考えられる。

ここで情報源といっているのは、データベース、知識ベース、フラットファイル、あるいは情報を必要に応じて提供するソフトウエアなどを指している。インターネットのような広域ネットワークでは各情報源は従来の分散システムとは比較にならぬほど自律的であり、また必要なときに必要な情報源が利用できるかどうかの信頼性は高くない。そこで問合せ時に利用可能な情報源だけから、問合せ結果を生成することを考えなければならない。本稿では、問合せ時に利用することが可能で、処理の対象とする情報源群を仮想データベースと定義する。インターネット上の関連する情報源をすべて探索するすることはできないので、ここでは伝統的なデータベースで仮定している閉世界仮説は仮定することはできない。つまり仮想データベースは開世界という意味で不完全なデータベースである。本稿では仮想データベースを動的に構成するためのモデルを与え、そこでの問合せ処理を考察する。

さまざまな情報源の中から必要な情報源を探すツールとして、すでに、archie, gopher, netfind, netscape, WWW (World Wide Web) などさまざまなツールが提供されている。これらを基に、ユーザからの問合せに対して、必要な情報の所在を自動的に探し、問合せを実行し答えを返すシステムとしてソフトボット (Softbot; software robot)[6] が提案されている。またロボットにより数万サイトの情報源を収集し分類し、それらに対する検索サービスを行っている Yahoo (http://www.yahoo.com) や Lycos (http://lycos.cs.cmu.edu) のようなものも現れている。これらに対し、本稿での仮想データベースは、単に個々の検索結果の収集だけではなく、それぞれの情報源が互いに協調することによってひとつの問合せ処理を行うことを目的としており、多様な情報源に対し一定の制約を与えなければならない。

一方、現在の広域ネットワーク上の複数の情報源間の統合については、従来のネットワークとは質的に異なっている。それぞれの情報源は高い自律性をもっているので、データの表現形式や格納形式、データに含まれる用語や概念の差異、など多くの異種性を解消が必要がある。このためにデータベースの分野ではマルチデータベース [10, 9] の研究があり、大規模知識ベースに関しては ARPA プロジェクト [5, 7] を中心にして多くの提案がなされている。本稿では、異種分散協調問題解決系として研究開発されている Helios[15, 1, 14] を基にした、広域ネットワーク上のモデルを考える。これは情報の大域性に関する制約を緩和するためである。そしてこのアプローチを、マルチデータベース、ARPA プロジェクト、Softbot など他のシステムと比較する。

本稿ではまず 2 節で関連研究をいくつか説明し、本稿の仮想データベースの設計方針を検討する。3 節では異種分散協調問題系として研究開発された Helios の概要を説明する。そして 4 節では Helios に基づいて仮想データベースを考え、他の関連研究との違いを明確化する。5 節では仮想データベースでの問合せを検討する。

2 関連研究

必要な情報源の探索のために Softbot[6] が知られている。Softbot の情報源探索機能の特長は以下の 3 つにまとめられる。

- **統合性**: ftp, telnet, mail などの機能の他に、archie, gopher, netfind などの探索機能を含んでおり、インターネットに対して統合された環境をもっている。
- **柔軟性**: 上のさまざまな機能を動的に組み合わせることにより、必要とする情報源を探索することができる。

- **学習機能**: 実行時に収集した情報によって探索方法を自動的に変更することができる。

このような探索のために、Softbot では情報源モデルを Prolog のある種の拡張によって記述している。これは実行時に得た情報を元に学習機能によって更新される。しかしこのような柔軟性はインターネットのような不特定多数の情報源自体の探索を行なうには有効だが、探索された情報源を他と統合するためには拡張が必要である。本稿では Helios によるエージェント化と環境とその階層化機構を利用することによってそれを考える。

複数のデータベースや知識ベースなどから必要な情報を取り出そうとする研究のひとつに SIMS (Services and Information Management for Decision Systems)[2] がある。これは Oracle や Loom など個別の情報をドメインモデルで表現し、それら情報の種別と所在を情報源モデルで表現する。問合せは知識表現言語としての Loom で表されるが、それが SIMS に送られると情報源モデルによって情報源ごとの部分問合せに分割され実行プランが生成される。それはさらにドメインモデルによって修正され、実行される。この問題点は、情報源モデルが静的でかつそれに組み込まれる情報が大域的であるために、インターネットのような環境には向かないことである。Helios は中間的な大域情報として環境を設定することにより、大域性の制限を緩め、動的な変化を局所化することによってこれを改善することを目指している。

ARPA モデルでは、知識表現のプラットフォーム (知識交換言語) として KIF[7] を、エージェント通信言語として KQML[5] を用いている。全体のアーキテクチャは連邦性を採っており、強力で柔軟なシステムが構築できる。ただしここで問題にしたいのは、特定のドメインの知識表現と、小規模な問題へのこれらの適用の柔軟性である。KIF は表現の可能性は与えるがそれがドメイン (エージェント群) の交換言語として妥当な表現であるかどうかは別の問題である。Helios モデルでは、プラットフォームを環境ごとに複数設定することでこれを改善している。つまり局所化が可能な問題については、必要最小限の問題解決系の修正ですませることを考えている。

3 Helios の概要

Helios の動機は、複雑な問題は単一の言語や単一のパラダイムの枠内で表すことが困難なことが多い、ということであった [15]。効率的に問題を表現するためには、複数の言語によって表現された問題解決器間での協調処理としてモデル化するのが素直な場合が多い。ここでの問題解決器は、ユーザプログラムだけでなく、データベースシステム、知識ベースシステム、制約解消系などを総称したものである。したがってそれはまたマルチデータベースシステムの拡張としても [14]、ソフトウエアエージェント [8] としても考えることができる。

3.1 エージェントと環境

Helios では任意の問題解決器を皮と呼ぶモジュールで覆うことによってエージェントとして定義することができる。文献 [8] では、このようなエージェント化として、transducer (変換器) 法、wrapper (コード挿入) 法、rewrite (書換え) 法の 3 つを挙げている (図 1) が、Helios のエージェント化もこれらに対応している[1]。データベース管理システムのようなサーバに対しては変換器を別につくり、外部呼出し機能を付加するだけで必要機能を充足できるものはコード挿入法、そうでないものには書替え法によってエージェント化する。これらエージェントで、自らメッセージを発信できるものを能動的、そうでないものを受動的と呼ぶ。

エージェント間の通信 / 協調のための場を環境と呼ぶ。ここでいう環境は対象となるエージェント間でのみで共有される大域情報を持つだけでよく、すべてのエージェント間での共有情報を必要としているのではないので、問題に適したプラットフォームを選択することができる。複数のエージェントをもつ環境もひとつの大きな問題解決器と考えられるので、これも皮で覆うことによってまたエージェント化することができる。このようにして構成されたエージェントを複合エージェント、そうでないものを単純エージェントと呼ぶ。複合エージェントを使うことによってエージェントをネストできる。これは問題解決器間の依存関係に対応させることができる。

環境には、各エージェントが発行するメッセージの内容を互いに理解するために共通の型システムやオントロジーが定義されている。これは環境内でのみ有効なもので、各環境ごとに定義することができる。インターネットのような広域ネットワーク上のすべてのエージェント / 情報源が共有しうる型システムやオントロジーは非現実であるので、このような局所的な定義は有効である。

[1] Helios の「皮」にはコード挿入法のイメージがあるが、3 つに対応している。

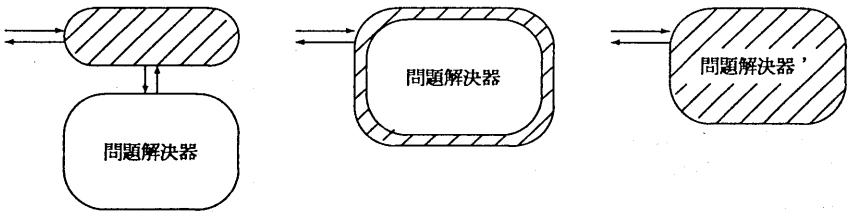

図 1: エージェント化の種類

各エージェントは自分の機能を外に対してメソッドという形で提供しており、エージェント間はメッセージでやりとりされる。エージェントからのメッセージは環境によって受けとられ、環境は下記の 3 通りの方法で他のエージェントに送付する。

- アドレス指定方式:
  メッセージの宛先として他のエージェント名が明示的に指定されており、そのエージェントに送付する。
- メソッド指定方式:
  該当メソッドをもっているすべてのエージェントにメッセージを送付する。
- 機能指定方式:
  メッセージの処理内容が書かれており、その機能を持っているエージェントすべてにメッセージを送付する。

このようなメッセージの配布 (ルーティング) を行なうために、Helios は各エージェントが自己モデルを持つことを義務づけている。環境は各エージェントから自己モデルを受け取り解析して、

- エージェントディレクトリ
  (エージェント名→エージェントの物理アドレス)
- メソッドディレクトリ
  (メソッド名→エージェント名)
- 機能ディレクトリ (機能名→エージェント名)

の 3 種類のディレクトリを構築する。機能ディレクトリについては、実行時に必要機能が満たされていないと判断されたエージェントについてはディレクトリから該当情報が削除される。

Helios での環境は局所的でかつネストされているので、上記ディレクトリで必要エージェントが検索されなかった場合には、この環境を含む複合エージェントの環境にメッセージは送付される。このようにして外の環境にエージェントを探しにいくが、もっとも外側の環境はユーザであると位置付けられている。

KQML では協調促進器 (facilitator) のルーティング機構として

- アドレス指定方式
- 購読 (subscribe) 方式
- ブローカ方式
- リクルート方式

の 4 つをもっている [5]。購読方式は、PACT[4] の知的ルーティングのように情報提供側 (advertiser) と情報利用側 (subscriber) が互いに協調促進器に対して必要な情報を提供し、協調促進器がそれらを突き合わせる (matchmaking) 方式である。Helios では環境が各エージェントから必要情報 (自己モデル) を収集するので、静的な環境では同一であると考えられる。ブローカ方式はすべてのメッセージのやりとりを協調促進器経由で行なうのに対し、リクルート方式では協調促進器を経由するのは最初の問合せだけである。Helios ではすべてのメッセージは環境を経由しており、ブローカ方式が標準となっている。

このようなエージェント間の通信のアーキテクチャとして、直接通信法と連邦アーキテクチャに分類されることが多い (たとえば文献 [8, 12])。Helios の環境は、連邦アーキテクチャの協調促進器に対応しており、単にメッセージのルーティングだけでなく、問合せの分割や実行の監視、オントロジーによる概念 / 語彙変換も行なうようになっている。ただし連邦性では一般に、エージェントの階層構造を採らないため、いくつかのエージェントを抱えている協

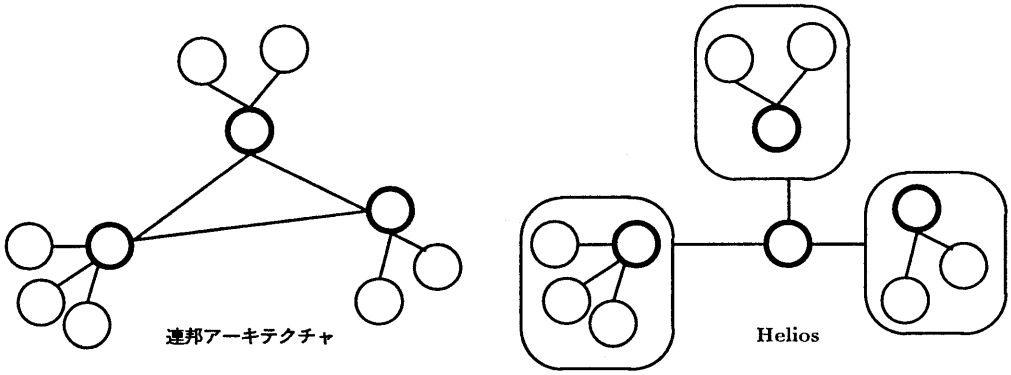

図 2: 協調促進器と環境 (上の図の太丸)

調促進器間でのやりとりが必要となるが、Helios では複合エージェントによりつねにひとつの環境の中で情報がやりとりされる (図 2) ので、より単純なやりとりとなる[2]。情報源 (エージェント) の動的な変化に対してはどちらもこれを局所化することができる。階層化については一長一短があるが、Helios ではデータベースのモジュール化をより重視した結果である。

Helios ではこのように環境毎に情報の大域性を設けられるように、環境定義言語 (ENVL; environment definition language) と皮定義言語 (CAPL; capsule definition language) をもっている。

3.2 プロトコルと代理エージェント

Helios のもうひとつの特徴としてトランザクションに基づいたメッセージプロトコルがある [1]。これは、エージェント間の交渉 / 協調もメッセージ列の論理的単位と考えることができるので、これをトランザクションに対応させ交渉 / 協調を制御しようというものである。たとえば契約ネット [13] の場合は図 3のようになる。こので a は仕事の依頼者エージェント、m は環境 / マネジャ、c は契約 (候補) 者エージェントを表している。n は仕事を提示したエージェントの数を、m はその中から入札に応じたエージェントの数を表している。仕事を行なうためには一定の資源を保存する必要があるため、入札した時点でそれらは施錠されることになる。入札が失敗すると、失敗者の資源は解錠される。エージェントはそれぞれ自律的に動いているので、それら資源の共有は自然に起こり、トランザクションに基づいた交渉 / 協調プロトコルは自然である。

[2]ただし実装上環境が複製されるかどうかは別の問題である。

```
nego_query(To,Query,Price,Res,[Sup,Inf,Diff]) ←
    query(To,Query,Price,Res),Price ≦ Inf,
    Res:=Price, commit(To).
nego_query(To,Query,Price,Res,[Sup,Inf,Diff]) ←
    query(To,Query,Price,Res),Price > Sup,
    rollback(To).
nego_query(To,Query,Price,Res,[Sup,Inf,Diff]) ←
    query(To,Query,Price,Res),Res=="accept",
    commit(To).
nego_query(To,Query,Price,Res,[Sup,Inf,Diff]) ←
    query(To,Query,Price,Res),int(Res),
    Res ≦ Price, rollback(To).
nego_query(To,Query,Price,Res,[Sup,Inf,Diff]) ←
    query(To,Query,Price,Res),Res=="reject",
    New_Price:=Price+Diff, New_Price ≦ Sup,
    nego_query(To,Query,New,New_Res,[Sup,Inf,Diff]).
nego_query(To,Query,Price,Res,[Sup,Inf,Diff]) ←
    query(To,Query,Price,Res),int(Res), Res ≦ Sup,
    X:=(Price+Res)/2, Y:=Price+Diff,
    New_Price:=mini(X,Y),
    nego_query(To,Query,New,New_Res,[Sup,Inf,Diff]).
```

図 4: 値切り交渉のための方略記述

プロトコルは ENVL で定義することができ、その環境内では大域的である。このプロトコルに従って各エージェントは方略 (strategy) をメソッドのインプリメンテーションとして CAPL で定義することができる。たとえば図 4はその方略の記述例である。

エージェント間の通信を効率化するために、各エージェントはメッセージと共に、一定の判断能力と権限を持った代理エージェントを生成することができる。環境によって

メッセージの流れ	意味	メッセージ	文献 [13] でのメッセージ
$a \rightarrow m$	仕事の依頼	*query*	
$m \rightarrow c \times n$	仕事の提示	*begin-trans+plan*	task-announcement
$c \times k \rightarrow m$	入札	*evaluate*	bid
$m \rightarrow c$	仕事の依頼	*query*	announced-award
$m \rightarrow c \times (k-1)$	落選の通知	*rollback*	
$c \rightarrow m$	結果の受取り	*answer*	final-report
$m \rightarrow c$	終了	*commit*	
$m \rightarrow a$	解を送付	*answer*	

図 3: トランザクションに基づいた契約ネット

選択されたエージェントにメッセージと共に送られることによって、環境を経由したエージェント間の通信は、相手先エージェントと代理エージェントの間で済ませることができる。

4 仮想データベース

Helios での仮想データベースはひとつのエージェントとして定義できる。構造面から考えれば、それは

(皮、環境、エージェント集合)

によって定義する。エージェント集合は情報源モデルで定義される。

4.1 Helios の情報源モデルとディレクトリ

Helios の環境は、ある目的をもった問題解決器 (エージェント) の集まりによって定義される。たとえば遺伝子情報処理のある環境では、配列情報や酵素反応などのデータベース群、および類似判定や構造予測などのツール群の集まりに対し、共通のメッセージプロトコルが定義され、環境を定義することができる。静的にはこれらがひとつの複合エージェントとして定義される。新たなエージェントがこの環境に参加する場合、他のエージェントに影響を及ぼさないという前提であれば、その環境で定義された共有情報に同意するか、その保守的な拡大が可能な場合にのみ参加できる。

閉じた分散環境での Helios の情報源モデルは、上に同意したエージェントに対し

(エージェント名、物理的なマシンアドレス)

の集合で与えられる。各マシン上での Helios デーモンがエージェントの起動 / 管理を司っている。これに対し、広域ネットワーク上の情報源については、文献 [11] に micro-*Quixote* によって ftp でアクセス可能なファイル群を仮想的なデータベースとみなしうることを示した (その一部の例は図 5)。これに従って、Helios の (拡張) 情報源モデル

```
ftp::{
    ftp_site[category=mule,
             where="etlport.etl.go.jp",
             directory="/pub/mule"];;
    ftp_site[category=gnu,
             where="prep.ai.mit.edu",
             directory="/pub/gnu"];;
};;
file::{
    FILE/[category=C,where=Where] <=
        ftp:ftp_site[category=C,
                     where=Where,directory=P],
        ext:ftp_ls[where=Where,directory=P,
                   file=FILE],
        ext:ftp_file_date[file=FILE],
        ext:ftp_file_owner[file=FILE];;
};;
ext::{
    ftp_ls[where=Where,directory=D,file=F] <=
      &true||{F ## ftp_ls[site=Where,directory=D],
               F =< &top[where=Where,what=What] };;
    ftp_ls[where=Where,directory=D,file=F] <=
      ftp_ls[where=Where,directory=D,file=F] ||
      {F ## ftp_ls[site=Where,directory=D],
       F =< &top[where=Where,what=What] };;
    ftp_file_date[file=FILE]|{ FILE.date == Date } <=
      &true||{ Date ## ftp_file_date[file=FILE] };;
    ftp_file_owner[file=FILE]
                  | { FILE.owner == O } <=
      &true||{ Owner ## ftp_file_owner[file=FILE] }
};;
```

図 5: micro-*Quixote* による仮想的データベース (一部)

は micro-*Quixote* によって記述する。

情報源モデルにしたがって、そのとき利用可能な情報が環境のディレクトリに設定される。そのとき利用可能でないものに対しては、必要に応じて継続的にアクセスの可能性を問い合わせることができる。しかし Softbot のように

単なる情報収集だけではないので、通常のエージェントと同じ制約が課せられるのはいうまでもない。

4.2 情報源の異種性

各情報源の異種性はエージェント化によって (CAPL による定義によって) 構文的には共通化される。環境は (ENVL によって) 共有すべき型システムを定義し、各エージェントはそれによってメソッドを公開する。メソッド以外は隠蔽されるので必ずしも共通化する必要はない。メソッドのインプリメンテーションとして各問題解決器の固有のデータと共通型データの変換が組み込まれている。一般にメソッドは以下の形をしている。

メソッドID@入力型→出力型
⇐ 型変換 | インプリメンテーション

Prolog のような型のない言語で記述された問題解決器であったとしても、環境においては強い型づけがなされる。

メッセージの内容自体の異種性については、それぞれのエージェントの概念 / 語彙体系を他の協調相手のエージェントのものに変換する必要がある。このために環境ではオントロジーを持っている。それは基本的には

(エージェントID: 概念 / 語彙
⟺ エージェントID: 概念 / 語彙)

の集合として定義される。これはデータベースでは、値の変換だけでなく属性名の変換も行なう。属性名や値の結合・分離はエージェントの皮で構文的な変換として行なう。たとえば CAPL と ENVL の定義によって図6のような変換が可能となる。

固有のデータ:	月日年 =07/20/95
(皮)	↕
エージェント→環境:	年 =1995, 月 =07, 日 =20
(オントロジー)	↕
環境→エージェント:	y=1995, m=July, d=20

図 6: CAPL と ENVL のデータ変換の例

5 仮想データベースでの問合せ

Helios では関係する情報源をまとめてひとつのエージェント化しているので、問合せはつねにひとつのエージェントに対してなされる。

1) 皮によって必要なら問合せを変更し、内部の環境に送付
2) 環境はディレクトリと資源モデルにしたがい、実行プランを作成する。これには以下のものがある。
   (a) 分散環境を意識した空間的なプラン
   (b) 解を合成するための同期プラン
   (c) 代替解あるいは全解を求めるためのプラン
3) エージェントに (部分) 問合せを送った後の実行監視 (現在は時間監視のみ)

実行不可能になったエージェント、あるいは実行可能になったエージェントがあれば、必要なタイミングで、ディレクトリを縮退させたり追加したりする。

環境を離れた問合せは、基本的にはエージェント間の協調処理に委ねている。エージェントから環境に送付される問合せは、環境にとっては、トランザクション制御を除けば新規問合せと変わりない。ただし仮想データベースでは、エージェントがたとえ問合せ実行時であっても、生成 / 消滅する可能性があるので、それを制御する必要がある。

6 おわりに

Helios の目的は、広域ネットワーク上のさまざまな情報源を単に収集することではなく、それらを協調的に働かせることによってひとつの問合せ処理を行なわせることである。そのための共通情報をもつことに同意した情報源 (エージェント) のみが Helios の情報源モデルに登録される。この意味で Helios は開放的な広域ネットワーク上での仮想データベースの構築を目指しながらも、閉じたデータベースを考えている。しかし広域ネットワークの性格上それらが必ずしもつねに利用可能ではないということから、伝統的なデータベースの閉じた世界とは異なっている。

本稿では述べられなかったが、環境に参加するエージェントはソフトウエアに限っているわけではない。エージェントが人であれば、それは CSCW へと発展させることもできそうである。これは今後の検討課題としておく。

参考文献

[1] A. Aiba, K. Yokota, and H. Tsuda, "Heterogeneous Distributed Cooperative Problem Solving System Helios and Its Cooperation Mechanisms", *Proc. FGCS'94 Workshop on Heterogeneous Cooperative Knowledge-Bases*, Dec. 15, 16, 1994. *to appear in IJ-*

CIS, 1995.

[2] Y. Arens, C.Y. Chee, C.-N. Hsu, and C.A.Knoblock, "Retrieving and Integrating Data from Multiple Information Sources", *Int. J. of Intelligent and Corporative Information Systems*, Vol.2, No.2, pp.127-158, 1993.

[3] C.M. Bowman, P.B. Danzig, U. Manber, and M.F. Schwartz, "Scalable Internet Resource Discovery", *Communications of ACM*, vol.37, no. 8, pp.98-107, 114, 1994.

[4] M.R. Cutkosky, R.S. Engelmore, R.E. Fikes, M.R. Genesereth, T.R. Gruber, W.S. mark, J.M. Tenenbaum, and J.C. Weber, "PACT: an Experiments in Integrating Concurrent Engineering Systems", *IEEE Computer*, January, pp.28-38, 1993.

[5] The DARPA Knowledge Sharing Initiative External Interface Group, "Specification of the KQML Agent-Communication Language — plus example agent policies and architecture", June, 1993.

[6] O. Etzioni and D. Weld, "A Softbot-Based Interface to the Internat", *Communications of ACM*, vol.37, no.7, pp.72-76, 1994.

[7] M.R. Genesereth and R.E. Fikes, "Knowledge Interchange Format Version 3.0 Reference Manual", Stanford U., *Logic Group Report Logic-92-1*, June, 1992.

[8] M.R. Genesereth and S.P. Ketchpel, "Software Agent", *Communications of ACM*, vol.37, no. 7, pp.48-53, 147, 1994.

[9] 上林弥彦, "マルチデータベースの研究課題", 情報処理, vol.35, no.2, 1994.

[10] A.K. Elmagarmid and C. Pu (eds), *ACM Computing Surveys: special issue on Heterogeneous Databases*, vol.22, no. 3, 1990.

[11] 新部裕, 高橋千恵, 横田一正, "Micro *Quixote* の実現とその拡張機能", 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, July 20-22, 1994.

[12] 西田豊明、"協調型アーキテクチャによる知識の共有と再利用"、人工知能学会誌、vol.9, no.1, pp.23-28, 1994.

[13] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transaction on Computers*, Vol. C-29, No.12, pp.1104 - 1113, December 1980.

[14] 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, July 20-22, 1994.

[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", 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.