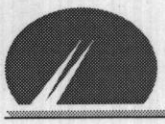
INFORMATION
SCIENCE
TECHNICAL
REPORT

NAIST-IS-TR94003
ISSN 0919-9527

知識の共有のための ソフトウェア技術

西田豊明

1994年1月



NAIST

〒630-01
奈良県生駒市高山町8916-5
奈良先端科学技術大学院大学
情報科学研究科

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-01, Japan

知識の共有のためのソフトウェア技術

西田豊明

奈良先端科学技術大学院大学 情報科学研究科

要旨

現在の人工知能技術をはじめとする情報処理技術の行き詰まりを解決するための方策の一つとして、大規模知識ベースシステムの構想が提案され、開発に着手され始めている。大規模知識ベースシステムの研究のねらいは、今日の知識ベースシステムの限界を越えるため、どの領域において知識ベースを構築する場合にも必要になってくる共通的知識を体系化して供給することによって新たに知識獲得すべき知識の量を大幅に削減することである。

特に、大規模な知識の共有・再利用を考えると、共有・再利用すべき知識を多数の自律的なエージェントに分散させ、エージェントの協調によって共有再利用に関わる種々の問題解決を図る協調型アーキテクチャが有望視される。協調型アーキテクチャによる知識の共有と再利用の研究ははじまったばかりであるが、ARPA の知識共有活動 (The ARPA Knowledge Sharing Effort) を中心に急速に成果が生まれつつある。

本稿では、まず大規模知識ベースの実現を目指した最近のアプローチの概観、主要な概念と技術について述べる。次に、協調型アーキテクチャによる知識の共有・再利用の方式の枠組みについて述べる。

1 大規模知識ベースシステムの研究の動機

1.1 知識ベースシステム

過去30年近くにわたって、いわゆる**エキスパートシステム** —実行可能な形式に整備された専門家の知識に基づく高度な専門的問題解決能力をもつコンピュータソフトウェア—の研究開発の努力が続けられてきた[39, 25]。我が国でも多くのエキスパートシステムが開発と実用化が行われた[16, 46]。エキスパートシステムの開発思想は、人工知能システムの問題解決能力は一般的な問題解決能力によるのではなく、問題領域や問題解決法に関する知識の質と量に依存するという**知識原理**である。

知識原理を具体化するために、**知識ベースシステム**というソフトウェアアーキテクチャが用いられてきた。知識ベースシステムは、問題解決法を**知識表現言語**によって表現した**知識**を格納した**知識ベース**と、その内容を解釈実行する**推論エンジン**から構成される。初期の研究では、専門家の知識を取り出し、それを知識表現言語で表現して知識ベース化することによって、狭い範囲についてではあるが高い問題解決能力を実現できることが実証された[7]。

1.2 知識ベースシステムの問題

知識ベースシステムの研究開発・実用化が進められて大規模な知識ベースシステムの開発が行われるようになると、いくつかの問題が顕在化してきた[49]。

知識獲得のボトルネック 知識ベースシステムを開発するためには、既存の知識ベースシステム構築用ツールの求めているレベルまで、専門家の知識を分析、抽出、体系化、詳細化、手続き化する必要がある。これまでは、専門家にインタビューをすることによって知識を抽出し、それを知識表現言語で表現し、プロトタイプを構築してそれを再び専門家に提示して詳細化・洗練するという知識獲得の作業を繰り返してきたが、それは膨大な労力を要するものであった。

狭さ 初期の知識ベースシステムの開発では、知識獲得が非常に効率の悪いものであったため、デモンストレーション効果のある効果のあるシステムを作るためには特定の狭い専門領域に特化せざるを得なかった。そのようなシステムは、設計時に考慮していなかった現象に対して大変もろく、専門家に比べて急激に問題解決能力が低下する傾向があった。

既存システムとの統合の困難さ 初期の知識ベースシステムはどれも既存のデータベースシステム、CADツール、リアルタイムシステムなどとの統合を考慮したものではなかったので、孤立的な性格が強く、システムとしての有用性を低下させるものであった。

知識メディアとしての弱さ 知識ベースシステムに蓄積される知識は基本的にはコンピュータ主体の問題解決を指向したものであるため、蓄積された知識自体は専門家や作業員にはブラックボックスに近く、蓄積されている知識そのものは役に立たない。また、知識表現言語で記述できない知識やノウハウは

放置される傾向があった。そのため、知識ベースシステムは問題解決に必要な知識をほとんど完全に知識ベース化するまでは役に立たない。

方法論の欠如　小規模なプロトタイプ開発と異なり、大規模で実用的な知識ベースシステムの構築には、一定の開発方法論が不可欠である。

1.3　大規模知識ベースのめざすもの

大規模知識ベースシステムの研究では、上に述べた今日の知識ベースシステムの限界を越えるため、どの領域において知識ベースを構築する場合にも必要になってくる共通的知識を体系化して供給することによって新たに知識獲得すべき知識の量を大幅に削減することをねらっている。

理想的な大規模知識ベースシステムは図1のように個々の知識ベースシステムに対して設計時または実行時に必要な知識を提供するインフラストラクチャとしての役割を果たす。

大規模知識ベースシステムの研究は、集積指向のアプローチと共有・再利用指向のアプローチの二つに大別できる。**集積指向のアプローチ**では、標準的な知識表現言語やモデリング言語を設定して、常識的知識や基礎的知識を包括的に収集し、蓄積することに重点を置いている (図 2a)。一方、**共有・再利用指向のアプローチ**では異なる方法論や概念化に基づく知識ベースシステムやソフトウェアシステムを協調させるための枠組みの構築に重点を置いている (図 2b)。

大規模データベースシステムは、定型性の高い大量のデータを集積し、効率よく利用するための土台とするための性能面を重視している。大規模データベースシステムは大規模なデータベースシステムと言い換えることができる。これに対して、大規模知識ベースシステムは、定型性が低い大量の知識や問題解決法を集積し、広い範囲の応用システムで再利用できるようにするための機能性を重視している。

大規模知識ベースシステムは、広い範囲の方法論に基づく知識を持つばかりでなく、新しく知識ベースシステムを構築するための知識部品ライブラリも提供する知識ベースシステムの母体になっているという点で、知識ベースシステムが単に大規模になったものとは区別される。

以下では、まず文献 [37] に従って大規模知識ベースの実現を目指した最近のアプローチの概観、主要な概念と技術について述べる。次に、文献 [38] に従って協調型アーキテクチャによる知識の共有・再利用の方式の枠組みについて述べる。

2　集積指向のアプローチ

集積指向のアプローチは、現在の人工知能の技術レベルを進めるのに一番有効な方策は、新しい知識表現法や推論方式の研究ではなく、ほどほどの知識表現言語を使って知識の体系を実際に作ることであるという信念に基づいている。集積される知識は、知識ベース部品やソフトウェア部品のように情報内容を解釈して問題解決に役立てる解釈装置（インタープリタ）が存在するものから、自然言語テキストやマルチメディア部品のように情報内容が人間にしか意味をなさないものまでの広がりがある。また、情報内容に関しても

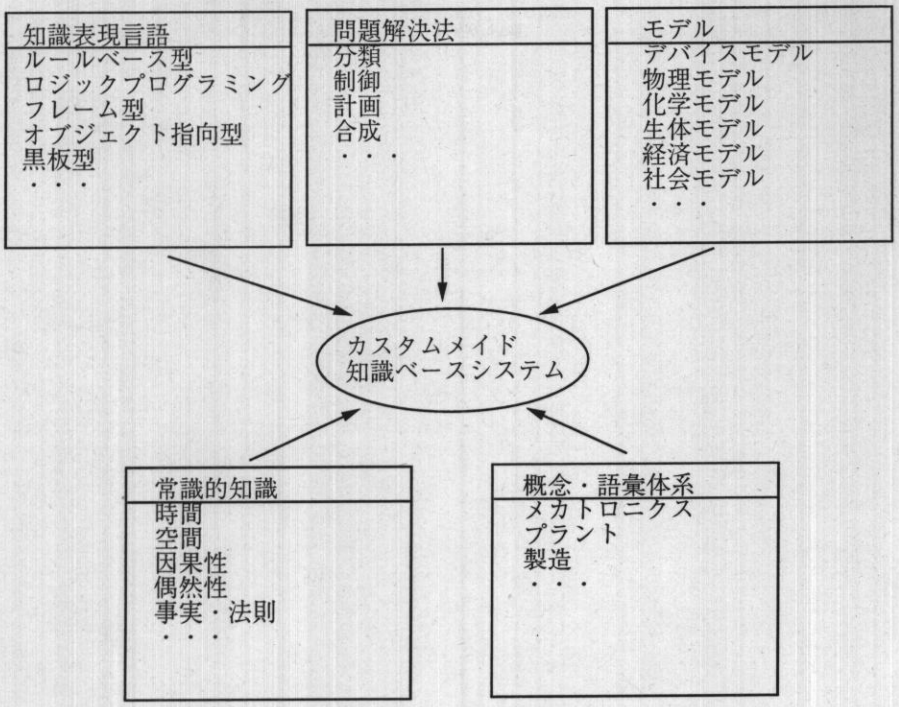


図 1: 知識ベースシステムの母体としての大規模知識ベースシステム

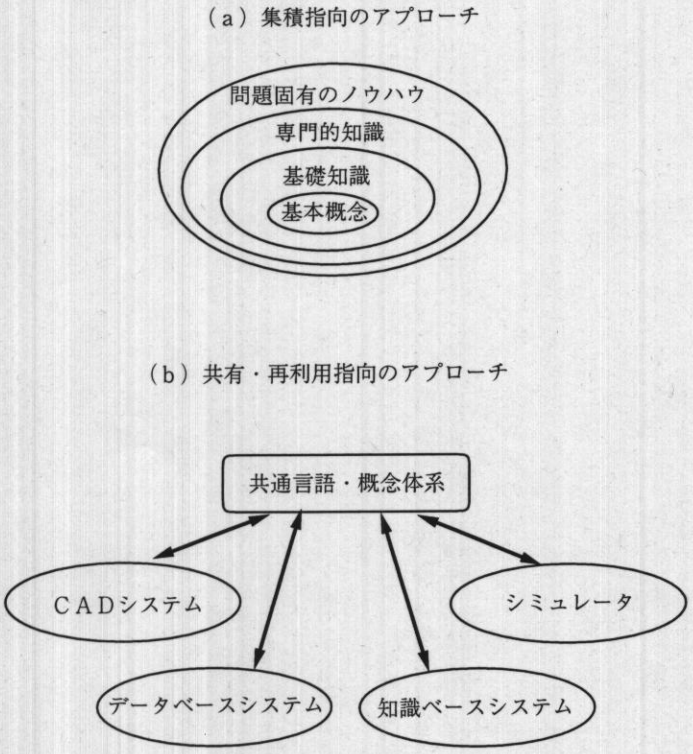


図 2: 大規模知識ベースシステムへの二つのアプローチ

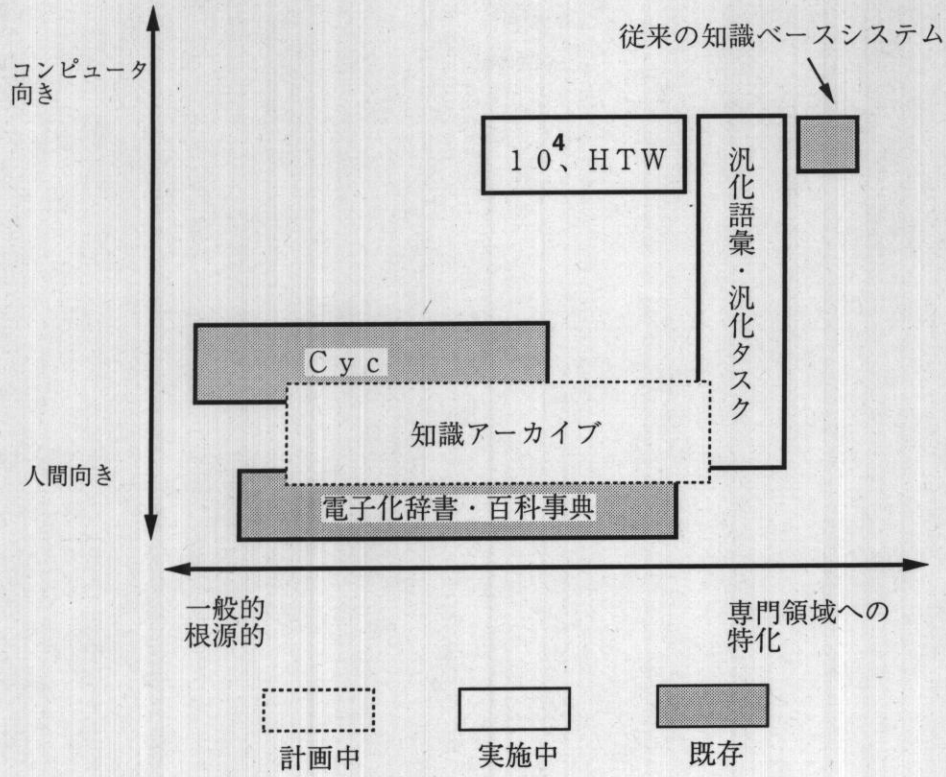


図 3: 集積指向のアプローチによるプロジェクト

基本的な認識の根幹をなすものから専門領域のノウハウまでの広がりがある。どのあたりの知識に重点を置いて収集するかはプロジェクトのねらいと目的に依存する。図3に、実施中または計画中のプロジェクトを集積される知識の実行可能性と専門性によって分類したものを示す。

2.1 Cyc プロジェクト

Cyc プロジェクトは、全ての知識に共通する常識的知識を包括的に収集することをねらって、アメリカの MCC(Microelectronics and Computer technology Corp.) で 1984 年から 10 年計画で行われている [23, 22, 47, 48]。目標は 200 人年をかけて 400 万項目を収集することである。すでに中間成果の配布が行われている。

知識ベースシステムの観点から言うと、Cyc で収集する常識的知識は既存の知識ベースシステムで収集された狭い専門的知識と人間の知識の間を埋めるものであり、別々に開発された知識ベースシステムを結合して協調させるための意味的な糊 (semantic glue) の機能を果たす。

知識の収集法に関しては、Cyclist と呼ばれる知識投入者が所定の知識源を理解して Cyc の知識表現で記述するという人手に基づくものである。これは、一定量の知識が集積されるまでは、機械学習や自然言語理解が有効に機能しないので、人手による収集が不可避であるという認識に基づいている。プロジェクトの初期の段階では、百科事典が知識源として選ばれた。現在は、計算機科学、アパレル、法律、化学から知識の収集が行われ

ているという。

Cycの知識ベースは、認識レベル(Epistemological Level, EL)とヒューリスティックレベル(Heuristic Level, HL)から構成される。ELは、Cycと外部ユーザ(コンピュータプログラムまたは人間)のコミュニケーションのための知識表現のレベルであり、効率的な推論を行うためにCyc内部で用いられるHLとは区別される。両者を対応づけるためにTell&Askと呼ばれるトランスレータが用いられる(図4)。

Cycにおける知識を記述するための知識表現言語はCycLと呼ばれる。ELのCycLは、フレーム型の言語を1階述語論理に基づく制約言語(CL)によって強化したものである。ELにおける知識の記述例を図5に示す。

CycLによる大規模知識ベース構築作業と利用をサポートするために、継承、自動分類、知識の依存関係の管理、議論の生成・比較・撤回・矛盾解消を中心としたデフォールト推論、推論制御のためのメタレベル推論、制約処理、複数の特化された推論エンジンの管理などの豊富な推論機構と、スプレッドシート型のフレームエディタUE、概念の空間的配置などを行うグラフィックエディタMUEを中心としたインタフェースが開発された。

2.2 知識アーカイブ

コンピュータ上での実行可能性に関わらず自然言語、形式言語、図形言語、画像、音響などのさまざまなメディアの知識を大量に集積した**知識アーカイブ**の構築を目標とするプロジェクト化の努力が行なわれている[33, 51]。公表されている構想によると知識アーカイブの知識はいくつかの層に階層化される。量的に中心を占めると思われるのは、知識メディアで表現され、分析の対象となるように整備された**知識ドキュメント**である。知識ドキュメントは、自然言語テキスト、知識部品・プログラム、データに分類される。知識ドキュメントを蓄積した知識ドキュメントベースの上に、構造化された知識の層が作られる。機能面では、知識アーカイブは知識の収集・獲得・検索・翻訳・応用などのサービスを提供する。

2.3 10^4 プロジェクト、HTWプロジェクト

Cycプロジェクトや知識アーカイブとは対照的に具体的な工学的問題解決に的を絞り、知識ベースシステムで直接利用可能な知識の収集と体系化を主要目標として設定している。これらのプロジェクトでは、知識ベースが基本的には対象モデルと問題解決法(例えば、汎用診断エンジンや一般設計法)に分解できるという**モデルに基づく推論**(model-based reasoning)を基本パラダイムとし、モデルの記述と収集に重点が置かれている。

東京大学工学部における10^4プロジェクト[24]では、機構学における知識の体系化の試みとして、機構部品のモデルの集積を行っている。様々な観点からのモデルを関連づけるために**メタモデル**[20]という手法が用いられている。メタモデルは図6のように物理世界に関する常識的概念に基づいて異なるモデルにおける概念間の関係付けを行うための機構である。メタモデル機構で使われる概念に関

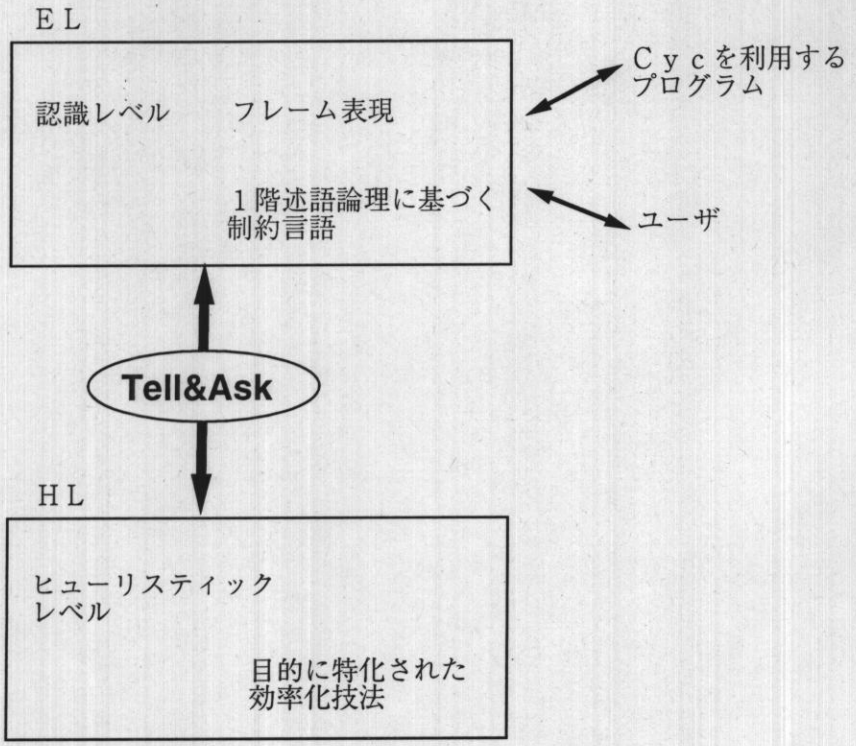


図 4: Cyc の知識ベースの構成

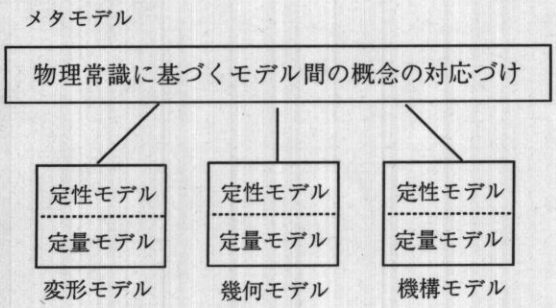


図 6: メタモデル機構

する記述を蓄積するための知識ベースは、**フィジカルフィーチャーデータベース**と呼ばれる。フィジカルフィーチャーの表現は定性プロセス理論 [9, 36] の枠組みを拡張したものであり、記号的なシミュレーションが可能である。10^4 というプロジェクト名は、中間的な収集目標として設定したフィジカルフィーチャーの個数を表している。

Stanford 大学知識システム研究所における HTW(How Things Work) プロジェクトでは、複数の領域にまたがる装置の設計と診断に必要な対象のモデルの構築、シミュレーションなどを支援する DME(Device Modeling Environment)[18, 17] の研究が行われているが、ここでも対象のエンジニアリングモデルの収集が中心的な関心の一つになっている。

2.4　汎化語彙、汎化タスク

10^4 プロジェクト、HTW プロジェクトと同様に専門性の高い知識に焦点を当てている。汎化タスク [4]、汎化語彙 [26, 31] などが提案されている。

大阪大学産業科学研究所で行われている MULTIS プロジェクトでは、知識ベースシステム部品を記述するための語彙 (**オントロ**

(a) フレーム型言語による “residents”(住民) の記述

```
residents
  instanceOf: (Slot)
  inverse: (residentsOf)
  makesSenseFor: (GeopoliticalRegion)
  entryIsa: (Person)
  specSlots: (lifelongResidents
              illigalAliens
              registerVoters)
  slotConstraints: ((coTemporal u v))
```

「residents はスロットの一種である。residentsOf は逆の関係を与えるスロットである。GeopoliticalRegion タイプのユニットだけがこのスロットを持つことができる。このスロットのエントリはふつう Person ユニットである。X の lifelongResidents スロット、illigalAliens スロット、または registeredVoters スロットのエントリは、また residents ユニットのエントリであると考えてよい。u の residents ユニットのエントリが v であれば、ふつう u と v は同時代のものである。」

(b) 制約言語 CL による “Buying”(購入) の記述

```
(Implies
 (LogAnd
    (allInstanceOf E Buying)
    (performedBy E X)
    (objectInvolved E Y)
    (seller E Z)
    (holdsDuring Y cost N SO)
    (occursIn E SO))
 (LogAnd
    (allInstanceOf (SkolemFunction (X Y Z) 'PAYING1) Paying)
    (performedBy (SkolemFunction (X Y Z) 'PAYING1) X)
    (objectInvolved (SkolemFunction (X Y Z) 'PAYING1) Z)
    (amountGiven (SkolemFunction (X Y Z) 'PAYING1) N)
    (subEvents E (SkolemFunction (X Y Z) 'PAYING1) X))))
```

全ての Buying(購入) のインスタンス E に対して、E の行為者を X、売り主を Z、コストを N とすると、E の部分イベントが存在し、その行為者は X、支払先は Z、金額は N である。

図 5: EL における知識の記述例

ジー) の整理と体系化が行われている。オントロジーはタスクオントロジーとドメインオントロジーに分類される。タスクオントロジーは、問題解決法を記述するのに適した量（数十から数百語）と粒度をもつ語彙（とその背後にある概念）体系であり、ドメインオントロジーは、問題解決の行われる領域（ドメイン）の対象や現象を記述するための語彙（とその背後にある概念）の体系である[1]。

MULTIS プロジェクトでは、まずタスクオントロジーが収集された [27]。タスクオントロジーは、汎化語彙 (汎化名詞と汎化動詞) によって記述される。汎化語彙は専門分野の違いを越えて共通する問題解決法を記述するためのものである。

インタビューによる知識獲得システムでは専門家が自分の専門領域の語彙と汎化語彙と

[1]オントロジーに関する議論については文献 [29] 参照

の対応を取ることによって、専門的問題解決知識の事例化、再利用、自動合成が行われる。汎化語彙はコンピュータ向きの知識と人間の知識を対応づけに大きな役割を果たすことが期待される。これまでに、汎化語彙を用いてスケジューリングタスクに関する問題解決知識の収集が行われている。

より一般的には、オントロジーとは対象領域に存在する対象やその間の関係に関する**概念化** (conceptualization) とそれを表現するための**語彙** (terminology) を陽に規定したものである[2]。知識ベースにおけるオントロジーのタイプと役割を図7に示す。3.4節で述べるように、オントロジーを陽に扱うことは共有・再利用指向のアプローチでも非常に重要になる。

2.5 応用

集積指向のアプローチでは、目的を限定せずにまず集積することが重要であるというシーズ指向のアプローチであるので、具体的な応用はある程度の集積が行われてからになる。大規模知識ベースシステムの提供する情報のインフラストラクチャの存在は、知識ベースシステムばかりでなく本質的に知識を必要とする自然言語処理システムやCAIシステムの有用な知識源となる。反面、使用目的を明確にせずに収集した知識を実際の応用システムで利用できるまでにするにはかなりの労力を要することが懸念される。

[2]もともと哲学用語であり、存在 (existence) の体系的な取扱いを意味する。

表 1: オブジェクト指向とエージェント指向

キーワード	特性
オブジェクト指向	データと手続きのカプセル化、抽象化、メッセージ伝達、継承
エージェント指向	エージェントの自律性 (リフレクション、自己組織化)、高レベルの相互作用、協調問題解決

3 共有・再利用指向のアプローチ

共有・再利用指向のアプローチでは、大規模知識ベースシステムを緩く結合された**エージェント**の集まりとして実現する方向をめざしている。ここでいうエージェントとは、所定のプロトコルに基づくメッセージ交換によって外部のプログラムと通信し、一定の情報処理能力を提供するソフトウェアモジュールを指し、ふつう知識ベースシステム、データベースシステム、ソフトウェアツールなどのツールにメッセージ処理をするモジュールをかぶせてカプセル化することによって実現される。

一般に、エージェントという概念は、オブジェクト指向で言われるオブジェクトよりは抽象度が高い。エージェントはオブジェクト指向の技術などによって実現される責務 (commit) や説得 (persuade) などによって規定されている (表1)。一方、分散人工知能 [1] ではより高度な機能をもつエージェントの研究が行われている。例えば、文献 [3] ではエージェントは環境や環境に存在する他のエージェントの意図やプランを理解し、環境の変化を知覚して行動する能力を持つものとして規定している。しかし、エージェントの

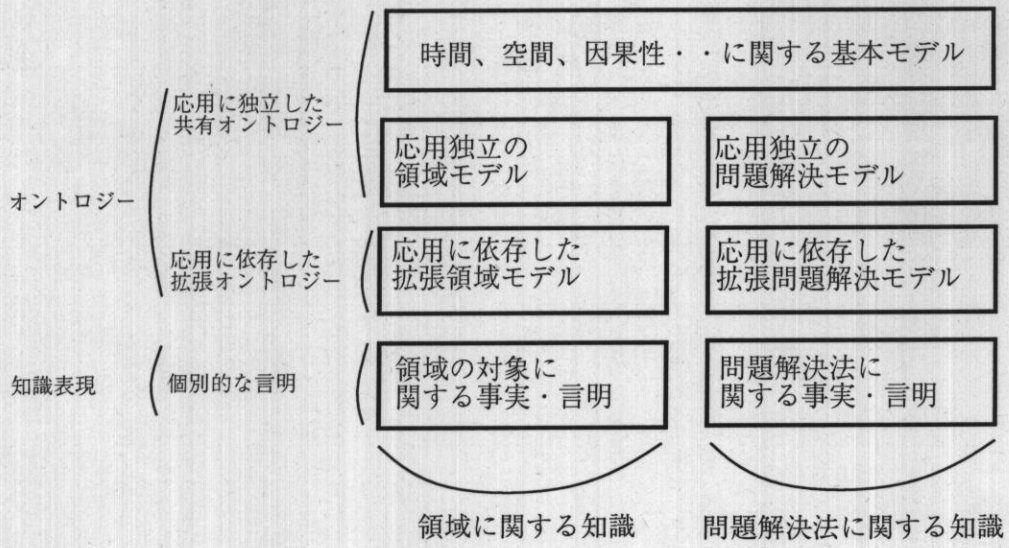


図 7: 知識ベースにおけるオントロジーの役割

外界や他のエージェントを理解する能力がエージェント間の協調にどのような影響を与えるかについてはまだ十分にわかっていない。

共有・再利用指向のアプローチのねらいは、

1. 従来の知識ベースシステムで課題とされていた既存システムとの統合の問題への本格的な解決の方向が示され、知識工学ばかりでなく、情報科学全体へのインパクトとなりえる。原理的には集積指向の大規模知識ベースシステムも一つのエージェントとして組み込まれる。

2. 既存のツールを組み込むためには、内部を修正する必要はなく、ツールの入出力を共通言語に相互変換するトランスレータを開発するだけでよい。

3. 問題解決のための単一の知識表現言語を要求されないので、従来問題とされてきた表現力と推論可能性のトレードオフが問題にならなくなる。知識ベースシステム間の共通言語はもはや問題解決を行うためのものではないので、単に表現力さえあればよい。

4. 協調のための機構は、エージェント収集のための枠組みとしても有効であり、エージェントの個数が十分でないときでも有用である。

5. 方式が、分散型計算環境と多人数による並列開発という現実によく適合している。

などである。

この方向の研究は、AAAI の知識共有と再利用のための作業部会 [32, 40]、デジタル図書館 [19]、知識コミュニティプロジェクト [35, 34] などで行われている。また、具体的な応用への適用は上に述べた PACT プロジェクトの他では、DICE プロジェクト [44] などで行われている。

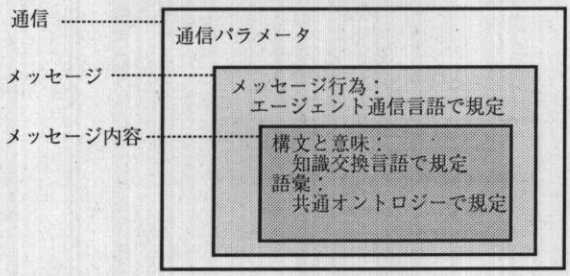


図 8: エージェント間でやりとりされるメッセージの階層

3.1 共有・再利用指向の技術の枠組

エージェント間で適切な相互作用が行われることを保証するためには、エージェント間で交換されるメッセージは共通の言語と規約に基づいたものでなければならない。はじめにエージェントがソフトウェアモジュールである場合について考えてみよう。

エージェント間でやり取りされるメッセージは図8のように、3層に分けられる。最も外側の層は通信層であり、メッセージを実際に伝達するために必要となる低レベルの通信に関する情報が記述される。その内側の層はメッセージ層であり、メッセージによる行為を**エージェント通信言語**を用いて記述する。最も内側の層であるメッセージ内容は、メッセージにおいて参照される処理対象の記述である。構文と意味は**知識交換用言語**で規定され、語彙は**共通オントロジー**で規定されたものが用いられる。知識の共有再利用では、内側の二つの層に焦点があてられる。

エージェント通信言語 (agent communication language) は、メッセージによるエージェントの行為 — 情報伝達 / 質問 / 依頼 / ... — を規定し、行為に必要な情報を記述するために用いられる。

知識交換用言語 (knowledge interchange format) は、メッセージによって伝達される情報内容 — 例えば、情報伝達の場合は伝達される情報内容、質問の場合は質問内容、依頼の場合は依頼内容 ... — を記述するための言語である。知識交換用言語は、あくまでも情報交換のためのものであり、問題解決法を実現するための内部知識表現言語とは必ずしも同一のものでなくても良い。

共通オントロジー (common ontology) は、外部表現言語で表現される概念の体系を概念記述言語によって表現したものである。多くの領域で使える (複数の) オントロジーの開発は大規模な知識の共有を実現するために重要な課題である。

各エージェントはこのように標準化されたメッセージを理解することは要請されるが、それにどのように対応するかは自由であり、エージェントの構築者に任されているのであるが、そこにも一定の規約が必要である。

人間もエージェントとなる場合は、共通言語と規約はソフトウェアに正しく伝達できるとともに人間にも理解しやすいように抽象的でコンパクトなものでなければならない。そのようなメディアの研究は Stefik が提唱し [45]、引き続き研究が行われている [14]。特に、コンカレントエンジニアリングではシステムで行われる種々の決定の相互の依存関係の記述が重要になる [41, 21]。

ARPA の知識共有活動で提案されているエージェント系のアーキテクチャを図9に示す。メッセージ交換はエージェント間で直接行われるのではなく、(現在のところは非常に単純な) ツール固有の知識と標準の知識交換言語と相互変換する**協調促進器**

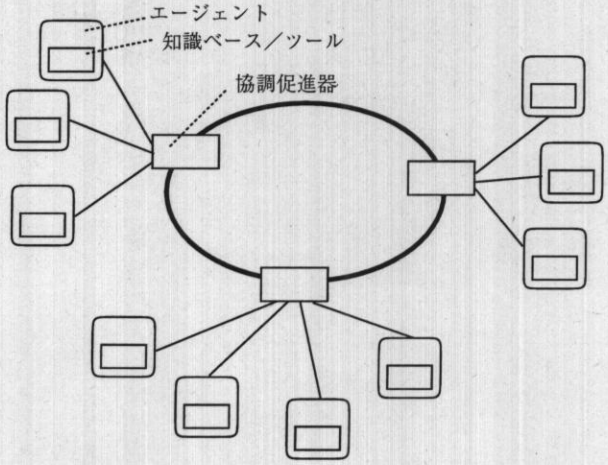


図 9: 連邦アーキテクチャ

(facilitator) を介して間接的に行われる (図 10)。協調促進器は、エージェント間の相互作用を促進するための特殊なエージェントである。協調促進器の果たす役割は、

1. エージェントから発信されたメッセージ内容を解析して宛先を決定する。

2. エージェントの取り扱うツール依存のメッセージ表現と共通言語によるメッセージ表現の間の相互変換を行う。

3. エージェントの初期化や実行のモニタリングを行う。

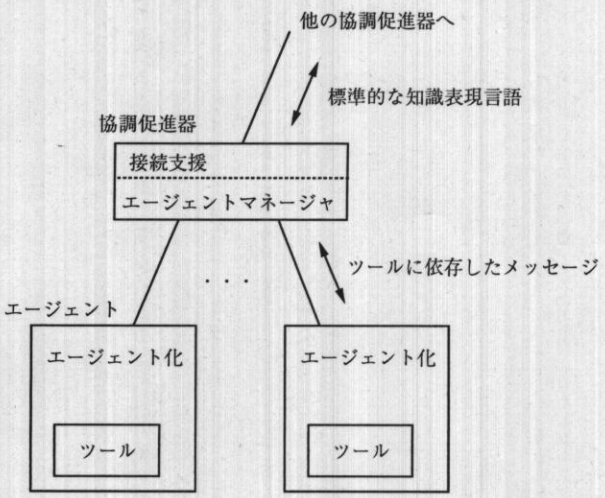


図 10: 連邦アーキテクチャにおけるエージェントの構成

これによって各エージェントは自分自身の言葉で推論でき、ほかのエージェントに情報を求め、他のエージェントに協調促進器を介して要求された情報を提供したりする。このような分散アーキテクチャは**連邦アーキテクチャ** (federation architecture) と呼ばれる。

エージェントの相互作用をさらに向上させるためには、さらに種々のサービスが必要である。例えば、どこにどのような能力をもつエージェントがいるか？といった質問に応答したり、エージェントの仕様変更などに自動的に対応する機能が必要になる。[34] などの先駆的研究があるが、今後一層の研究が必要である。

3.2 エージェント通信言語

ちょうど自然言語の発話が発話行為 (伝達、質問、依頼、 ...) と発話内容 (伝達内容、質問内容、依頼内容、 ...) に分解できるように、エージェント間で取り交わされるメッセージは、メッセージのタイプを規定する**メッセージ行為** (performative) と**メッセージ内容** (contents) に分解することができる。エージェント通信言語はメッセージ行為のタイプとプロトコルを規定する。

人工知能分野では、分散情報処理環境におけるエージェント間の相互作用のためのプロトコルの研究は分散人工知能 (Distributed AI)[1] における人間社会における相互作用の計算モデル化の研究として行われている。これまで、入札に基づく契約の行為をモデル化した契約ネット [43]、日常の約束行為をモデ

ル化したAgent0のプロトコル[42]、グループにおける合意形成過程をモデル化したCOSMO[50]、意図と知覚を中心としたもの[3, 2]などが提案されている。

以下では、KQML(Knowledge Query and Manipulation Language)[8]について述べる。KQMLは、ARPAの知識共有活動の外部インタフェースワーキンググループで開発された。エージェント通信に関するかなり基本的な機能だけを規定している。

KQMLは、知識ベース間の知識レベルでの相互作用のための言語として設計されたものであり、エージェントでのメッセージ送受の方法を規定するのではなく、個々のメッセージの意味内容を定義する。KQMLでは、メッセージは次のようなCommon Lispにおけるキーワード構文を用いて表現される。

```
((メッセージ行為タイプ)
  {(空白) :(キーワード) (空白) :(LispのS式)}*)
```

例えば、

```
(tell   :language KIF
        :ontology motors
        :in-reply-to q1
        :content (fastens frame12 motor1))
```

は、メッセージ内容記述言語としてKIF、オントロジーとして`motors`、メッセージ内容として`(fastens frame12 motor1)`をもつメッセージを表している。

KQMLメッセージは、それを受け取ったエージェントが一定の行為を遂行することが意図されていることに由来して、**遂行語**(performative)と呼ばれる。KQMLは実際にメッセージが伝達される低レベルの通信層に独立に定義された言語であり、次の仮定に基づいている:

1. 各エージェントは離散的なメッセージを伝達する単一方向の通信リンクによって結合されている、
2. これらのリンクには無視できない程度のメッセージ伝送の遅延が存在する、
3. エージェントに届くメッセージはどのリンクからのものであるか同定できる、
4. エージェントはメッセージを伝送するリンクを指定できる、
5. メッセージは相手先のエージェントに送信順に届けられる、
6. メッセージの伝送は信頼できる。

KQMLによるメッセージを理解し、然るべき応答を返すエージェントは、**KQMLエージェント**と呼ばれる。KQMLエージェントを外部からみると、**仮想的な知識ベース**(VKB: Virtual Knowledge Base)に基づく応答をしているとみなすことができる。

現在までに、情報伝達(`tell`, `deny`, `untell`)、質問(`ask-if`, `ask-all`, `stream-all`, `reply`, `sorry`など)、依頼(`achieve`, `unachieve`など)、生成器関係(`standby`, `ready`, `next`, `discard`など)、通知(`subscribe`, `monitor`)、ネットワーク処理(`register`, `unregister`など)、仲介処理(`broker-one`, `recommend-one`, `recruit-one`など)などのための30種類余りのタイプのメッセージ行為が定義されている。ユーザは新しいメッセージ行為を所定の形式で定義することによって与えられたメッセージ行為集合を拡張することができる。

これらはVKBに対する様々な要求を表している。いくつか例を示す。図11の例では、あるエージェントが別のエージェントとの間にストリームを設定して、情報の提供を受けている。図12の例では、`:content`で提供された内容について、将来変更があればそれを逐次通知してくれるよう要求している。図

(1) stream 設定

⇒
```
(standby :language KQML
         :ontology K10
         :reply-with g1
         :content (stream-about
                        :language KIF
                        :ontology motors
                        :reply-with q3
                        :content motor1))
```

(2) 承認

⇐
```
(ready :reply-with 2F0B
       :in-reply-to g1)
```

(3) 回答の要求

⇒
```
(next :in-reply-to 2F0B)
```

(4) 回答

⇐
```
(tell :language KIF
      :ontology motors
      :in-reply-to q3
      :content (= (val (torque motor1)
                       (sim-time 5))
                  (scalar 12 kgf)))
```

(5) さらなる回答の要求

⇒
```
(next :in-reply-to 2F0B)
```

(6) 回答

⇐
```
(tell ...)
```

⋮

(7) 打ち切り

⇒
```
(discard :in-reply-to 2F0B)
```

二つのエージェント: client と server 間のやり取りを示す。⇒ は client から server へ、⇐ は server から client へ、それぞれメッセージが送られることを示す。

図 11: stream を用いたやり取りの例

```
(subscribe :reply-with s1
           :language KQML
           :ontology K10
           :content (stream-about
                        :language KIF
                        :ontology motors
                        :content motor1)))
```

図 12: subscribe による変更通知の依頼

```
(advertise :language KQML
           :ontology K10
           :content (subscribe
                        :language KQML
                        :ontology K10
                        :content (stream-about
                                     :language KIF
                                     :ontology motors
                                     :content motor1)))
```

図 13: advertise による自己能力の宣言

13の例は、自分の能力を他のエージェントに知らせている。

協調促進に関わる遂行語として、

1. broker-one, broker-all, recruit-one, recruit-all (advertise を用いて、要求された処理能力を持つと宣言したエージェントに処理を依頼する)、

2. recommend-one, recommend-all (要求された処理能力を持つエージェントを推薦する)

が用意されている。エージェントはこれらの遂行語を含んだメッセージを協調促進器に送ることによって対応するサービスを受けることができる。

エージェントポリシー (agent policy) は、エージェントが受け取ったメッセージをどのような態度で処理するかを規定する。多くの応用においてエージェントポリシーは次のようなものである。

1. 正直さ (honesty): KQML で規定された意味論に基づいてメッセージ処理を行う、
2. だまされやすさ (gullibility): ほかのエージェントも自分と同じ信念を持つと信じて処理を行う、
3. 親切さ (helpfulness): ほかのエージェントも自分と同じ目標をもつと信じて処理を行う、
4. 責任 (responsiveness): 応答が期待されているメッセージ行為にはいつか必ず応答する、
5. 感情移入 (empathy): ほかのエージェントが必要とするメッセージ行為を明示されなくても推し量ることができる、
6. 適切性 (pertinence): ほかのエージェントの役にたたないと考えられるメッセージは送付しない。

3.3 知識交換フォーマット

知識交換フォーマットは、異なる問題解決用知識表現言語で記述されたツールを包含したエージェント間で知識交換を行うための中間言語としての役割を果たす。一般に、知識交換フォーマットが有用であるための要件は、

1. 表現力が豊かであり、インプリメンテーションに独立した知識レベルでメッセージ内容を記述できる、
2. 構文と意味が厳密に定義され、それ関して詳細な合意が得られる、
3. 自己拡張性がある、

などであると考えられる。

KIF(Knowledge Interchange Format)[10] は、1階述語論理を拡張して、項の定義、メタ知識(知識に関する知識)、集合、非単調理論などを記述できるようにした知識交換フォーマットである。例えば、「全ての作家は読者のうちの誰かから誤解を受ける」を表わす KIF 表現は次のようなものになる。

```
(forall ?w
   (=> (writer ?W)
       (exists ?R)
          (and (reads?R ?W)
               (not (understands ?R ?W))))
```

KIF を厳密に定義するために、モデル理論による意味論が与えられている。

KIF は宣言的な言語であるため変換時に手続き的な情報が失われてしまうことがある。

3.4 共通オントロジー

異なるエージェント間で協調的な問題解決が行われるためには、メッセージ交換に用いる言語の構文・意味だけでなく、オントロジーに関してもエージェント設計者とエージェントそのものの両方において合意が得られていなければならない。

Ontolingua は、異なる知識表現ツールに基づくエージェントの間に共通のオントロジーを定義することによる知識の共有の促進をねらったものである [12, 11, 13]。対象領域に現れる対象のクラス、関係、関数、オブジェクト、法則を記述するためには、KIF を拡張した構文と意味論に基づく宣言的な表現を用いる。Ontolingua はフレーム型の概念化(フレームオントロジー)をサポートするために、数十個の2階の述語を提供している。

図14に Ontolingua による概念定義の例(「(文献の) 著者」の概念定義)を示す。ここ

```
(define-class AUTHOR (?author)
  "An author is a person who writes things.
   An author must have created at least one document.
   In this ontology, an author is known by his or her real name."
  :def (and (person ?author)
            (= (value-cardinality ?author author.name) 1)
            (value-type ?author author.name biblio-name)
            (>= (value-cardinality ?author author.documents) 1)
            (<=> (author.name ?author ?name)
                 (person.name ?author ?name))))
```

?author が著者であるとは、?author が person であり、関係 author.name で規定されるちょうど一つの対象が存在し、それはクラス biblio-name のインスタンスでなければならず、author.documents で関係づけられる少なくとも 1 個の対象が存在し、author.name という関係と person.name という関係が同値であることである。

図 14: Ontolingua による概念定義の例

では、

value-cardinality:
⟨?domain-instance, ?binary-relation⟩
↦ ?n

value-type:
⟨?domain-instance, ?binary-relation, ?class⟩
↦ F/T

がフレームオントロジーとして提供された2階の関数、述語であり、それぞれ対象?domain-instance が2項関係?binary-relation によって対応づけられる対象の個数とタイプを表わす。

このようにして定義される概念体系を異なる知識表現言語に変換するトランスレータが開発されている。文献 [12] によるとこれまでに LOOM, Epikit, Algernon, pure KIF へのトランスレータが開発されている。また、CycL, KEE, EXPRESS などへのトランスレータの開発も容易であろうという見通しが得られている。例えば、図 14の定義の述語論理系の言語 Epikit への変換は図 15のようになる。ただし、このような変換は限られたものであり、あらゆる変換が可能というわけではない。

共通オントロジーに関する研究ははじまったばかりであり、

1. オントロジーをどのように設計するか、
2. オントロジーに関する合意をどのようにして形成するか、
3. オントロジー間の整合性をどのようにして吟味するのか、
4. 異なるオントロジーの間の対応付けをどのように行うか、

など、多くの興味ある課題が残されている。オントロジーに関する議論は [29, 28] などでなされている。

3.5 応用システムでの利用

共有・再利用指向のアプローチはコンカレントエンジニアリング [6] とも密接な関係があり、領域指向の効率的な情報交換方式の基盤を与えるものとなろう。

```
(=> (author $author) (person $author))
(=> (author $author)
    (exists $y (author.name $author $y) (biblio-name $y)))
(=> (author $author)
    (=> (author.name $author $y)
        (author.name $author $y2)
        (= $y $y2)))
(=> (author $author)
    (exists $y (author.documents $author $y)))
(=> (author $author)
    (<=> (author.name $author $name)
         (person.name $author $name)))
```

図 15: 図 14の定義の述語論理系の知識表現言語への変換

4 PACT プロジェクトにおける知識の共有と再利用への取り組み

PACT(Palo Alto Collaborative Testbed) プロジェクト[5]は、異なるサイトに分散した異種のツールの協調に基づく分散型設計に関する実証的研究を行うための実験環境であり、1991年に開始された。参加した主な研究機関は、Stanford大学、Hewlett-Packard、Lockheed、Enterprise Integration Technologiesである。これらの研究機関では知的CADのためのツールとして、

- NEXT-Cut(機械設計とプロセスプランニング, Stanford Center for Design Researchで開発)、
- Designworld(デジタル回路設計、シミュレーション、組立、検査システム, Stanford大学Computer Science学科とHewlett-Packardで開発)、
- NVisage(設計ツールのための分散型知識ベース型統合環境, Lockheed Information and Computing Scienceグループで開発)、
- DME(Device Modeling Environment: モデル形成とシミュレーション, Stanford Knowledge Systems研究所で開発)

が異なる時期に個別に開発されていた。PACTプロジェクトでは、これらのツールを連携させて、簡単なロボットマニピュレータの分散シミュレーションと単純で増進的な再設計を行う実験を行った。各ツールではマニピュレータの異なる側面のモデル化と、異なる工学的視点(力学、デジタル回路、ソフトウェア)からの推論が行われる[3]。このような研究はコンカレントエンジニアリングのなかに位置づけられる。

実験では、15台のワークステーション上に31個のエージェントが実現された。同一サイトのエージェント間の通信にはサーバ・クライアント方式が用いられ、遠隔地のエージェント間の通信には電子メールのためのプロトコルであるSMTP(Simple Mail Transfer Protocol)が用いられた。

[3]PACTでは連邦アーキテクチャを用いてエージェントを統合しているが、おもしろいことに、Designworld[15]では、連邦アーキテクチャが内部構造としても用いられている。

エージェント間の信頼性のある相互作用を保証するための共通言語として、KQML、KIF、Ontolinguaが用いられた。これらは、ARPAの知識の共有イニシアティブで開発されたものである。

PACTプロジェクトの第一の特徴は、分散型アーキテクチャを用いていることである。これと対照となるのは、サブシステムを共通の設計モデルを介して相互作用させる集中型のアーキテクチャである。この考え方は、異なるサブシステムが相互作用するためには、共通の言語、概念体系、...を必要とするという点は正しく捉えているが、個々のサブシステムが高度に専門化されて難解なものになってしまうとともに、共通の設計モデルが対象の全ての側面を記述した非常に複雑で巨大なものでなければならなくなる、という難点がある。分散型のアーキテクチャでは、共通の設計モデルを用いないのでシステムに含まれるどのサブシステムも過度に複雑化しなくてもすむ可能性がある。ただし、共通の設計モデルがなくても整合性のある設計が可能かどうか、検討が必要である。

第二の特徴は、エージェント間の通信による相互作用のための規約がインプリメンテーションの行われるシンボルレベルではなく問題解決の行われる知識レベルで行われていることである。このため、設計の種々の側面について作業する各エージェントの挙動をその情報が内部的に符号化されているフォーマットや低レベルの通信制御から独立して記述できる。これらの特徴はさらに協調促進器によって強化されている。

第三の特徴は、Ontolinguaと呼ばれる概念記述言語を用いて、エージェント間で交換されるメッセージで参照される概念体系(オントロジー)の形式的定義を与えて共通化することによって知識の共有のレベルが形式を越えて内容まで進められたことである。

しかし、現状ではエージェント間の協調はアドホックな合意に基づくものであり、また共通オントロジーの構築もかなり労力を要するため、エージェント系の規模拡大が困難であるという限界も認識されている。

5 展望

大規模知識ベース実現には、集積指向のアプローチと共有・再利用指向のアプローチの統合が必要である。最も単純な考え方は、大規模知識ベースを一つのエージェントとしてエージェント系のなかに組み込むことであるが、この方式は柔軟性を欠く。大規模エージェント系として大規模知識ベースを実現した方が柔軟性が高いと考えられる[34]。今後、そのような方向での研究の進展が期待される。さらに、次のような方向での研究が期待される。

5.1 知識メディア

PACTではエージェントはソフトウェアモジュールとして実現されるが、人間もエージェントで有り得る。MITのDICEプロジェクト[44]では、エージェントは一般に専門家とソフトウェアの共同体であると規定している。また、グループウェアへの統合[30]も考えられる。

このように、人間とコンピュータが協調して作業をする現実の環境を見ると、コンピュータ上での問題解決用の知識と人間の専門家が理解する知識を**知識メディア**[45]とし

て統一的に扱う必要がある。分散した設計環境における知識メディアの利用をサポートする表現の枠組みとしてSHADE(SHAred Dependency Engineering)[14]の研究が進められつつある。SHADEの提案された背景には共通オントロジーや知識表現は重要であるが、現在の技術レベルでは共有すべき情報を不完全にしか規定できないという認識である。記述が不完全であっても、メッセージに一定の記述子を与え、予約購読(subscription)、刊行(publication)などのサービスを行うことによって十分な有用性が確保できるという主張がされている。知識メディアの研究は、CSCWと分散問題解決を統合するアプローチとして今後の発展が期待される。

5.2 自律性

次の段階では、知識体系の形成、保守、利用の各段階においてより自律性の高いエージェント系の構築が目指されることになるであろう。**知識コミュニティ**構想[35]では、そのようなエージェント系は次のような特性を持つものとして捉えられている:

1. 人間と同様の知識源(書物や人間)から提供される人間向きの知識を知識コミュニティ向きのものに変換する、
2. 知識によって説明されていない現象や知識の矛盾を見つけ出し、それを未解決問題として明示する、
3. 妥当な仮説をたて、それを更新する、
4. 新しい知識が与えられると、知識の体系を自律的に再構成する、
5. 他の人間やエージェントとの間のインタフェースとなる秘書システムがある。

6 まとめ

大規模知識ベースシステムの研究開発の動向について集積指向のアプローチと共有・再利用指向のアプローチに分けて解説した。現在の人工知能技術をはじめとする情報処理技術全体の大きな発展をみるために、大規模知識ベースシステムによる知識インフラストラクチャの開発が期待される。今後は、大規模知識ベースシステムのライフサイクルを考慮した開発方法論の展開が必要になるであろう。

協調型アーキテクチャによる知識の共有と再利用の方向での研究ははじまったばかりである。今後の研究では、

1. 実証的研究: 具体的応用に協調型アーキテクチャを適用して、その有効性を示すこと、
2. 理論的研究: 知識表現、推論、分散協調、知識体系、知識体系の形成と自己組織化の形式化と理論的分析、
3. システム化研究: 協調型アーキテクチャによるエージェント系構築のためのツールとソフトウェア環境の整備、既存の枠組との統合

など、具体的な問題に即した深い取り組みが必要とされる。

参考文献

[1] Bond and Gasser, editors. *Readings in Distributed Artificial Intelli-*

gence. Morgan-Kaufmann Publishers, INC., 1988.

[2] Birgit Burmeister, Afsaneh Haddadi, and Kurt Sundermeyer. Generic configurable cooperation protocols for multi-agent systems. (personal communication), 1993.

[3] Birgit Burmeister and Kurt Sundermeyer. Cooperative problem-solving guided by intentions and perception. In *Proceedings of MAAMAW-91*, 1991.

[4] B. Chandrasekaran. Generic tasks in knowledge-based reasoning. *IEEE Experts*, pp. 23–30, 1986.

[5] Mark R. Cutkosky, Robert S. Engelmore, Richard E. Fikes, Michael R. Genesereth, Thoas R. Gruber, William S. Mark, Jay M. Tenenbaum, and Jay C. Weber. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, Vol. January 1993, pp. 28–38, 1993.

[6] Presun Dewan. Toward computer-supported concurrent software engineering. *IEEE Computer*, Vol. January 1993, pp. 17–27, 1993.

[7] Edward A. Feigenbaum. The art of artificial intelligence: I. themes and case studies of knowledge engineering. In *Proceedings of IJCAI-77*, pp. 1014–1029, 1977.

[8] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, P. Pelavin, S. Shapiro, and C. Beck. Specification of the KQML agent-communication language. Technical Report EIT TR 92-04, Enterprise Integration Technologies, 1992. (Updated July 1993).

[9] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, Vol. 24, pp. 85–168, 1984.

[10] Michael R. Genesereth and Richard Fikes. Knowledge Interchange Format version 3.0 reference manual. Technical Report Logic-90-4, Computer Science Department, Stanford University, 1990.

[11] Thomas R. Gruber. Ontolingua: A mechanism to support portable ontologies Version 3.0. Technical report, Knowledge Systems Laboratory, Stanford University, 1992.

[12] Thomas R Gruber. A translation approach to portable ontology specifications. In R. Mizoguchi, H. Motoda, J. Boose, Gaines B., and Quinlan R., editors, *Proceedings of the Second Japanese Knowledge Acquisition for Knowledge-based Systems Workshop JKAW '92*, pp. 89–108, 1992.

[13] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-4, Knowledge Systems Laboratory, Stanford University, 1993.

[14] Thomas R. Gruber, Jay M. Tenenbaum, and Jay C. Weber. Toward a knowledge medium for collaborative product development. In *Artificial Intelligence in Design '92, Proceedings of the Second International Conference on Artificial Intelligence in Design*, pp. 413–432, 1992.

[15] Pierre Nam Huyn, Michael R. Genesereth, and Reed Letsinger. Automated concurrent engineering in Designworld. *IEEE Computer*, Vol. January 1993, pp. 74–76, 1993.

[16] 日経AI(編). エキスパートシステム最前線 — 統合化への加速. 日経BP社, 1992. 別冊.

[17] 岩崎由美. 定性推論の応用に関する展望. 情報処理, Vol. 32, No. 2, pp. 163–170, 1991.

[18] Yumi Iwasaki and C. Low. Model generation and simulation of device behavior with continuous and discrete changes. Technical Report KSL-91-69, Knowledge Systems Laboratory, Stanford Universit, 1991.

[19] Robert E. Kahn and Vinton G. Cerf. The digital library project, volume 1: The world of knowbots (DRAFT). Technical report, Corporation for National Research Initiatives, 1988.

[20] 桐山孝司, 冨山哲男, 吉川弘之. 設計対象モデル統合化のためのメタモデルの研究. 人工知能学会誌, Vol. 6, No. 3, pp. 426–434, 1991.

[21] Mark Klein. Capturing design rationale in concurrent engineering teams. *IEEE Computer*, Vol. January 1993, pp. 39–47, 1993.

[22] Douglas B. Lenat. Toward programs with common sense. *CACM*, Vol. 33, No. 8, pp. 33–49, 1990.

[23] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-based Systems*. Addison-Wesley, 1989.

[24] 松本幹雄, 川合稔, 中田秀基, 山本文緒, 冨山哲男, 吉川弘之. 設計向き大規模知識ベースの研究. 1991年度人工知能学会全国大会（第5回）論文集, pp. 717–720, 1991.

[25] 溝口理一郎. エキスパートシステムI～III. 朝倉AIライブラリ, 1993.

[26] Riichiro Mizoguchi. Task ontology and its use in a task analysis interview system. In *Proceedings of JKAW '92*, 1992.

[27] 溝口理一郎, 小川均, 小島昌一, 進藤静一, 益田亮, 長橋一彦, 増井庄一, 坂間千秋. 知識の再利用を目指したタスク分析とタスクオントロジーの整備. Technical Report SIG-F/H/K/S/I-9201-6(12/4), 人工知能学会, 1992.

[28] Riichiro Mizoguchi. Knowledge acquisition and ontology. In *Proceedings International Conference on Building and Sharing of Very-Large Scale Knowledge Bases '93 (KBKS '93)*, pp. 121–128. Ohmsha, Ltd, 1993.

[29] 元田浩, 溝口理一郎, 西田豊明. 知識の共有と再利用ワークショップ報告. 人工知能学会誌, Vol. 8, No. 5, pp. 666–671, 1993.

[30] 村永哲郎, 守安隆. グループウェアのための情報共有技術. 情報処理, Vol. 34, No. 8, pp. 1006–1016, 1993.

[31] 中村祐一, 堀雅洋. 知識ベースシステム構築のための問題解決部品の同定 — タスクの性質を反映した部品空間について —. Technical Report 研究会資料 AI-91-68, 電子情報通信学会, 1992.

[32] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, Vol. 12, No. 3, pp. 36–56, 1991.

[33] 日本システム開発研究所 (編). 知識アーカイブ研究開発計画. Technical report, 機械振興協会・経済研究所, 1992.

[34] Toyoaki Nishida and Hideaki Takeda. Towards the knowledgeable community. In *Proceedings International Conference on Building and Sharing of Very-Large Scale Knowledge Bases '93 (KBKS '93)*, pp. 157–166. Ohmsha, Ltd, 1993.

[35] 西田豊明. 知識コミュニティ. 北野宏明 (編), グランドチャレンジ — 人工知能の大いなる挑戦 —, pp. 176–189. 共立出版, 1993.

[36] 西田豊明. 定性推論の諸相. 朝倉書店, 1993.

[37] 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.

[38] 西田豊明. 大規模知識ベース. 情報処理, Vol. 35, No. 2,, 1994.

[39] 大須賀 節雄. AI マップ — AI 研究のあり方. 人工知能学会誌, Vol. 7, No. 5, pp. 796–809, 1992.

[40] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, and R. Neches. The DARPA knowledge sharing effort: Progress report. In Charles Rich, Bernhard Nebel, and William Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*. Morgan Kaufmann, 1992.

[41] C. Petrie. The Redux' server. In: the Proceedings of the First International Conference on Intelligent and Cooperative Systems, Rotterdam, May 12-14, 1993.

[42] Yoav Shoham. AGENT0: A simple agent language and its interpretation. In *Proceedings, AAAI-91*, pp. 704–709, 1991.

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

[44] Duvvuru Sriram and Robert Logcher. The MIT Dice project. *IEEE Computer*, Vol. January 1993, pp. 64–65, 1993.

[45] Mark Stefik. The next knowledge medium. *AI Magazine*, Vol. 7, No. 1, pp. 34–46, 1986.

[46] 日経インテリジェントシステム (編). 特集 — 進化する知識ベースシステム. 日経 BP 社, 1992. 別冊 1992 年秋号.

[47] 寺野隆雄. Cyc 大規模知識ベース – 知識への第 1 歩か? 非常識な計画か? コンピュータ科学, Vol. 1, No. 2, pp. 114–118, 1991.

[48] 寺野隆雄. 大規模知識ベース技術の動向と課題. 1992 年度人工知能学会全国大会チュートルアル講演テキスト, pp. C13–C27, 1992.

[49] 寺野隆雄. 知識システム開発方法論. 朝倉書店, 1993.

[50] Stephen T. C. Wong. Messages and protocols for cooperative systems communications. (to appear in: Proc. MACC '92), 1993.

[51] 横井俊夫. 知識処理と自然言語処理の融合としての大規模知識ベース — 電子化辞書から知識アーカイブへ —. 人工知能学会誌, Vol. 8, No. 3, pp. 286–296, 1993.