知識コミュニティ KC_0 における知識共有メカニズム

Knowledge-sharing mechanism in the Knowledgeable Community KC_0

武田 英明　　飯野 健二　　西田豊明

Hideaki Takeda　　Kenji Iino　　Toyoaki Nishida

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

The Knowledgeable Community is a framework of knowledge sharing and reuse based on a multi-agent architecture. In this paper, we describe KC_0 system which is the first generation of the Knowledgeable Community. We discuss requirements for agents in the Knowledgeable Community first of all. According to the requirements, we describe communication among agents, agent programs, and agent organization. communication among agents is realized by using KQML, KIF and Ontolingua. Agent programs are realized by Common Lisp, Prolog in Lisp, and Emacs Lisp with interface programs to communication protocol (KAPI) and the agent programming tool (Kata.lisp). Agent organization includes some special agents like facilitator and mediatior. Facilitator checks network status to enable communication among agents, manages agents connected in the site, and handles messages among agents. We also show a testbed system which manages regional information for travel planning.

1　はじめに

われわれは、大規模知識ベースの実現として、知識の生産・共有・利用の枠組み**知識コミュニティ** (The Knowledgeable Community) を提唱し、そのプロトタイプの開発を進めている [4]。

本研究では、その第一段階である KC_0 システムの基本構成とこのシステムにおける知識共有の仕組みについて述べる。まずエージェントの持つべき能力についてのべ、その上でエージェント間コミュニケーションの実現法やエージェントの実現方法について述べる。

2　エージェントの設計指針

知識コミュニティ KC_0 におけるエージェントはネットワーク上に存在して、他のエージェントからメッセージを自律的に処理することのできる存在である。ここでのメッセージとは質問や回答あるいは処理依頼などである。

知識コミュニティの目標は全体として自律的 / 自己組織的知識システムとして働くことであるが、それは必ずしも個々のエージェントが全てそのような条件を満たすことを意味するわけではない。**知識コミュニティ**におけるエージェントは、極めて知的な応答をするものから、データベースシステムのような単純な応答しか許さないものまで多様なものを許すべきである。以下の項目はコミュニティの成員に期待されている能力である。

1. **Communication 能力**
   他のエージェントと適切な相互作用ができるための最低限の規約である。 communication のための共通のプロトコルを理解しなければならない。
2. **自己の概念体系の公開**
   各エージェントの知識が立脚する概念体系は公開されている必要がある。また、それらは他のエージェントが利用できる形でなければならない。
3. **自己の能力の公開**
   各エージェントはなんらかの方法で自分の持つ能力をコミュニティに対して公開しなければならない。これはエージェントが自分の能力として外部に知らしめたい能力であり、網羅的である必要もないし、場合によっては意図的に誤りの可能性もある。

人間のコミュニティのアナロジーでいえば、人間間で有用な会話が成立するには、会話能力があること (1 番目) はもちろん、ある程度背景が共通していること (2 番目)、さらに

連絡先: 奈良先端科学技術大学院大学 情報科学研究科, 〒 630-01 生駒市高山町 8916-5, 電話:07437-2-5262, fax:07437-2-5269,
E-mail: takeda@is.aist-nara.ac.jp, URL: http://flute.aist-nara.ac.jp/doc/people/takeda/

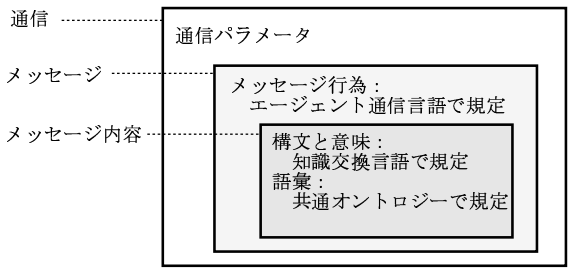

図 1: エージェント間でやりとりされるメッセージの階層

自分の役割 / 能力を理解していること (3 番目)、といえよう。ここで注意する点は、外界のモデルを持つことは必須ではない、ということである。自分以外にどんなエージェントがいて、どのような処理ができるかは必ずしも知っている必要はない。

3　エージェント間コミュニケーションの実現

エージェント間の通信は図 1に示すような階層構造をもっている。

メッセージ通信層　この層は計算機ネットワークの上でどのような方法でエージェント間の通信を実現するかを規定する。現在はイリノイ大学で開発されたメッセージプラットホーム、 ”Message Bus (MBus)” を利用して実現している。

MBus は unix のプロセス間でメッセージをやり取りするシステムである。マシン毎あるいはいくつかのマシン毎に ”mbus” と呼ばれるデーモン・プロセスを走らせる。 MBus を通じて通信を行ないたいプロセスは、まず mbus デーモンに対して自分を登録する。この mbus デーモンはプロセスが走っているマシンである必要はない。この後はこのプロセスは自分が mbus を通じて他のプロセスに対してメッセージを送ることができ、また他のプロセスからメッセージを受けとることができる。接続を切りたい時は、プロセスは、登録を抹消することを mbus デーモンに伝える。異なるマシンの mbus に登録されているプロセス間で通信をするときは、電子メイルを用いて行なう。

MBus 通信をエージェント間通信として容易に利用できるようにインタフェースを整備したものが KAPI(KQML API) である。 KAPI では、主として、 mbus への接続登録、切り離し、メッセージ送信、メッセージ受信の 4 つの関数を提供する。**知識コミュニティ**プロジェクトでは、 KAPI はスタンフォード大学などが集まって行なっている SHADE Project の中で開発された C 言語のライブラリを利用して、 Austin Kyoto Common Lisp (AKCL), Allegro Common Lisp, Lucid Common Lisp 用 KAPI($KAPI^{NARA}$) を開発した。

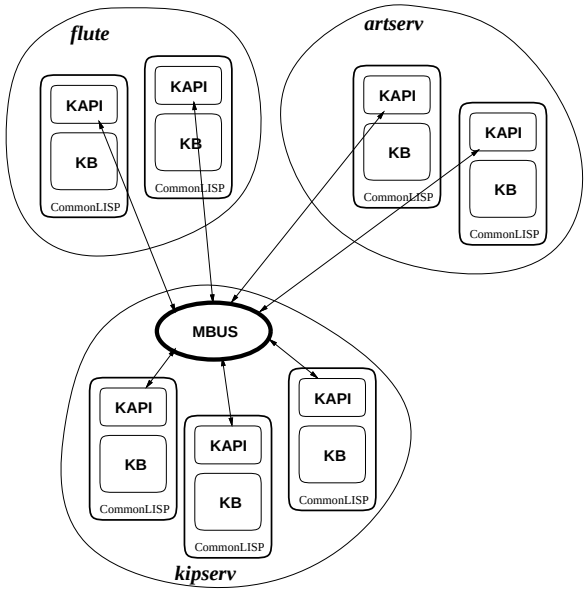

図 2: MBus and KAPI

また Emacs Lisp 用簡易版 KAPI、 KanKAPI も開発した。

KAPI は各 Lisp プロセス毎に利用され、 Lisp プログラムと mbus デーモンの中間に位置して、メッセージの送受などを行なう (図 2参照)。

メッセージ行為層　メッセージ行為層はエージェント間の communication 能力を実現するための共通なプロトコルを提供する部分であり、 KQML と KIF を主に用いている。

KQML　ここでのプロトコルは KQML[1] に従っている。KQML は発話行為をモデルとしたメッセージ行為のプロトコルである。 KQML ではメッセージは performative と呼ばれ、その種類 (performative type) によってそのメッセージを発した意味が指定される。また、同時に performative の中で許される項目の種類も規定される。 Performative type としては、 tell, ask-one, ask-about, ask-all, sorry などがある。また、項目としては、 sender (送り元)、 receiver(送り先)、 ontology (オントロジー)、 language(利用言語)、 content(内容) などがある。このうち、 language(利用言語) と ontology(オントロジー) はそれぞれ、 content (内容) の構文と意味を規定する。

KIF　現在、**知識コミュニティ**プロジェクトでは、利用言語として主として KIF[2] を用いている。 KIF(Knowledge Interchange Format) は知識の交換用の言語として提案されているもので、一階述語論理を基本とした記法によって各種の言語での表現を表現する言語である。

Ontolingua　エージェント間で共有する概念体系の表現としては、 ontology として表現し、具体的には Ontolingua[3]

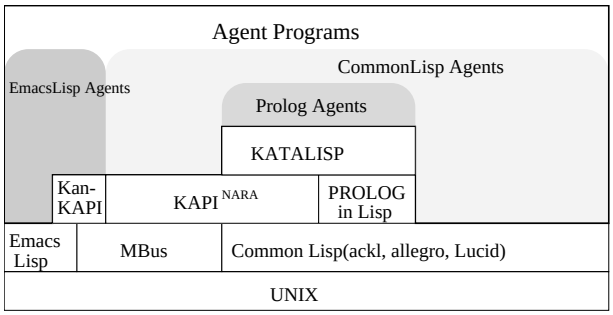

図 3: Agent programming environment

を採用している。 Ontoligua は ontology を書くための言語で、 KIF の表現を拡張して、 class や関係を定義できるようにしている。 ontolingua 上でフレーム型のオントロジーを定義することで、クラス階層をもったオントロジーを記述することが出来る。

4 エージェントの構成

エージェントを作成するとき、エージェントの起動、メッセージ処理、メッセージ待ち、終了動作などをプログラム本来の動作に加えて記述する必要があるが、これらはエージェント特有のプログラミングが必要であり、容易ではない。また、これらは各種のエージェントで共通な部分が多い。エージェントを開発する時により本来のエージェントのプログラミングに専念できるような環境が必要である。この一貫として、今回 Prolog 型知識ベースを容易にエージェント化するツール kata.lisp を用意した。このツールにおいては外部からの ask perfomative によるメッセージはメッセージ内容が Prolog の質問と解釈され、メッセージ内容を変数の束縛によって置き換えたものが、 tell perfomative で最初のメッセージの送信元へ返答される。また、予めいくつかの述語を宣言をしておくとこで、 Prolog の推論でその述語の unification に失敗したとき、他のエージェントへ質問をする。そして、その回答を用いて推論を継続する。

現在の**知識コミュニティ**におけるエージェントの開発環境を図 3にまとめる。

全体は unix 上で構築されている。そして、 MBus、 Common Lisp、 Emacs(Emacs Lisp) がその上のプロセスとして存在する。また、 KAPI は Common Lisp の program であり、 MBus の関数を外部関数として利用している。 KATALISP は Lisp 上での prolog インプリメンテーションを利用する lisp program である。 KanKAPI も KAPI と同様で、 MBus を利用した Emacs Lisp のプログラムである。

現在、 3 種類のエージェント・プログラミングが可能である。第一は KAPI を使った Common Lisp プログラミング

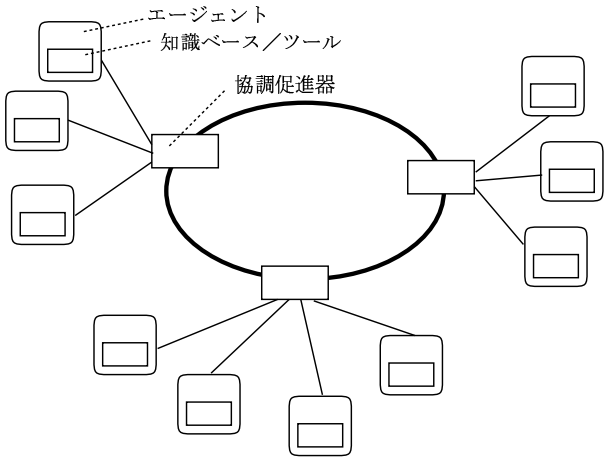

図 4: Organization of Agents in the Knowledgeable Community

によるエージェントの作成である。 2 番目は KATALISP を利用して prolog 型ルールの集合としてのエージェントである。ただし、これは Common Lisp の中で定義されている。 3 番目は KanKAPI による Emacs Lisp で書かれたエージェントである。

5 エージェント組織

メッセージを処理するという立場からみると知識コミュニティにおけるエージェントは表 1に示すような形で分類される。各エージェントは必ずしも他のエージェントの存在やその能力を知っているわけではない。 このため、エージェント組織に関わる特別なエージェントが存在して、メッセージの伝達や宛先の推定、形式の変更などを行なう。エージェント組織でのエージェントの存在 (どんなエージェントが現在ネットワーク上に存在しているか、あるいは存在しうるか) については協調促進器 (facilitator) と呼ばれるエージェントが担当する。また、宛先の変更、メッセージの表現形式の変更などは仲介器 (mediator) が行なう。

6 協調促進器

知識コミュニティ KC_0 では連邦アーキテクチャ [5] によってエージェントを構成する (図 4参照)。すなわち、協調促進器はサイト毎 (かならずしも一つのホストとは限らない) に存在し、そのサイトにあるエージェントを担当とする。

協調促進器 (facilitator) の機能は主に 3 つに分けられる。

- ネットワーク・マネージャ

  メッセージ処理デーモンの監視、他のサイトの存在、機能に関する知識、他のサイトの協調促進器との連絡、ネットワーク監視エージェント (inspector) への情報提供

表 1: Types of agents

<table>
<tr><td colspan="4">メッセージ伝達エージェント: 協調促進器 (Facilitator)</td></tr>
<tr><td rowspan="6">メッセージ処理
エージェント</td><td colspan="3">宛先なしメッセージ処理エージェント: 仲介器 (Mediator)</td></tr>
<tr><td rowspan="6">宛先付メッセージ
処理エージェント</td><td>エージェント組織
特有エージェント</td><td>Ontology server
Concept associator
Instance server
...</td></tr>
<tr><td colspan="2">情報サーバ</td></tr>
<tr><td colspan="2">推論サーバ</td></tr>
<tr><td colspan="2">計算サーバ</td></tr>
<tr><td colspan="2">問題解決器</td></tr>
<tr><td colspan="2">...</td></tr>
</table>

- エージェント・マネージャ
  そのサイトにあるエージェントに関する知識、エージェントの登録、削除、機能変更、エージェント・プロセスの生成と終了
- メッセージ・マネージャ
  メッセージの転送、メッセージID、日付などのメッセージへの付加、メッセージ交換のログ

7 Test Bed

現在、知識コミュニティ K_0 の test bed として、KC-Kansai プロジェクトを行なっている。これは関西地区への訪問に関する情報をエージェントとして実現して、旅行計画などの支援を行なうものである。

そのプロトタイプである KC-Kansai Nara(ver 1.3)[1]では、図5に示すようなエージェントを用意して、これらのエージェントに知識を分担させている。

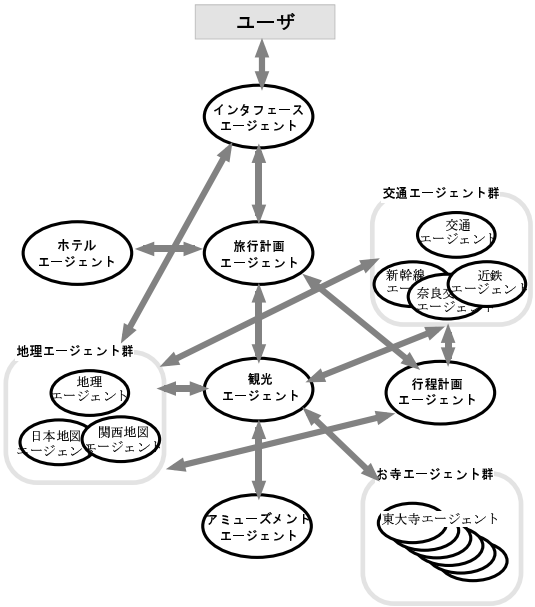

図 5: KC-Kansai でのエージェントの構成

8 おわりに

本研究では知識コミュニティ KC_0 におけるエージェントの構成の方法とエージェント間におけるコミュニケーションの方法について述べた。特にエージェントの持つべき3つの能力がどのような形でシステム化されるかについて述べた。ここで述べたシステムの要素はエージェントによる知識共有を実現するための基盤的技術である。今後はこの基盤的技術をさらに充実させて、より容易なエージェント作成と実行を可能にすることと共に、実際的な問題を対象に知識システムの試作をする必要がある。

参考文献

[1] 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).

[2] 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.

[3] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992.

[4] 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.

[5] 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.

[1] 詳細は `http://flute.aist-nara.ac.jp/doc/nishida-lab/demo/`