FIPA-KQML GATEWAYの実装

1E－1

嵯峨宏則*　伊藤聖吾*　須栗裕樹**　児玉英一郎*　宮崎正俊*
*岩手県立大学ソフトウェア情報学部　**株式会社コミュニケーションテクノロジーズ

1. はじめに

近年、エージェントに関する数多くの研究が行われ、実社会への応用が期待されている。
エージェントの定義は様々あるが、我々は、周りの状態と自分自身が持つ心的状態に基づき、プランを生成、実行するソフトウェアをエージェントと呼んでいる。ここで心的状態とは、エージェントの行動の判断基準になるもので、AGENT-0では、信念、判断、選択があり、FIPAでは確信、不確信、選択、意図がある。また、複数のエージェントとそれらが協調してタスクを遂行するために必要なエージェント管理サービス（メッセージ転送や名前解決などを行う）を合わせたものをマルチエージェントシステムと呼ぶ。マルチエージェントシステムには、FIPA準拠のものやKQMLベースのものが存在している。我々は、これらを分類するためマルチエージェントシステムのアーキテクチャを「心的状態」、「エージェント通信言語」、「トランスポート（プロトコルまでを含めた相互接続のためのインタフェース）」の3要素で定義する。このような定義のもと、マルチエージェントシステムはFIPAアーキテクチャとKQMLアーキテクチャに大別される。ここで問題点として、これらの異なるアーキテクチャのマルチエージェントシステム同士は互いに通信することができないということが挙げられる。我々は、この問題点を解決するためマルチエージェントシステムの相互接続に関する研究を行ってきた[1][2]。本論文では、これらの異なるアーキテクチャを持つマルチエージェントシステム間の相互接続を可能にするゲートウェイの実装について報告する。

2. ゲートウェイの概要

FIPAアーキテクチャとKQMLアーキテクチャのように異なるアーキテクチャのマルチエージェントシステム同士をゲートウェイを用いて相互接続するためには、「心的状態」、「エージェント通信言語」、「トランスポート」をゲートウェイは相互に変換できなくてはならない。図1にこのゲートウェイを利用した相互接続のモデルを示す。

3. ゲートウェイの実装

本研究では、このゲートウェイのプロトタイプシステムを以下の制限下で実装した。
(1)アーキテクチャは、FIPAアーキテクチャとKQMLアーキテクチャの二つに限定する。

Implementation of FIPA-KQML Gateway
*Hironori Saga, *Seigo Ito, **Hiroko Suguri,
*Eiichiro Kodama, *Masatoshi Miyazaki
*Faculty of Software and Information Science,
Iwate Prefectural University.
**Communication Technologies.

(2)エージェント通信言語で記述されたメッセージ内の命題部分を表現するコンテント言語は、FIPAアーキテクチャではSL、KQMLアーキテクチャではKIFとする。
(3)トランスポートはサポートしない。代わりにゲートウェイはメッセージを標準入出力を介して行う。
(4)AADL(Agent Architecture Description Language)はゲートウェイの動作を定義するために接続するマルチエージェントシステムのアーキテクチャを記述する言語であるが、これはKQMLからACLへの変換を指示する場合の(:stdin KQML :stdout FIPA)と、ACLからKQMLへの変換を指示する場合の(:stdin FIPA :stdout KQML)のみを実装する。
図2に、この制限下での実装モデルを示す。

3.1 実装モデルの構成要素

以下、図2の各構成要素について順に説明する。
ACLのメッセージ：ACLのメッセージはコミュニカティブアクト（メッセージの振舞を示す）とSENDER、RECIEVER、CONTENT、LANGUAGE、ONTLOGY、IN-REPLY-TO、REPLY-WITH、CONVERSATION-IDのパラメータからなっている。ACLでのコミュニカティブアクトは、INFORM、CONFIRM、DISCONFIRM、REQUESTの4種類で、それぞれ、通知、確信の通知、否定の通知、要求を意味している。
KQMLのメッセージ：KQMLのメッセージもACLのメッセージと同様であるが、KQMLでのパフォーマティブ（FIPA ACLのコミュニカティブアクトに相当）はTELLとACHIEVEの2種類で、それぞれ、通知、要求を意味している。
AADLによる動作記述ファイル：動作エンジンの動作を規定したファイル。KQMLからACL、ACLからKQMLの翻訳を指示する。ゲートウェイの起動時に、ファイル名がコマンドラインから渡される。KQMLへのメッセージをACLのメッセージに変換したい場合は、(:stdin KQML :stdout FIPA)と記述し、ACLのメッセージをKQMLのメッセージに変換する場合は(:stdin FIPA :stdout KQML)と記述する。
動作エンジン：AADLで記述された動作記述ファイルを読み込み、それに従ってKQML＋KIFパーザ/ジェネレータ、ACL＋SLパーザ/ジェネレータの動作を制御する。この動作エンジンはJAVA2を使用して実装した。
心的状態データベース：メッセージを翻訳する際に生成される中間メッセージ（メッセージの意味を表現したもの）を蓄積する。この心的状態データベースのスキーマは、キー、メッセージ識別子（どのメッセージからレコードが生成されたかを識別するための識別子）、送信受信エージェント名、受信エージェント名、命題表現（コミュニカティブアクトとコンテント命題より生成されるFPとREに対応した命題表現）、FP/REフラグ（命題表現がFPかREかを示すフラグ）、コンテント言語、オントロジー、返答要求識別子、返答識別子、会話識別子からなっている。このデータベースには、PostgreSQL7.02を使用している。
KQML＋KIFパーザ：動作エンジンの制御に基づき、標準入

力から得たKQMLのメッセージを解析し中間メッセージを作成後、心的状態データベースに登録する。

KQML+KIF ジェネレータ:動作エンジンに基づき、心的状態データベースの内容をもとに、KQML のメッセージを生成し、標準出力に書き出す。

ACL+SL パーザ:動作エンジンの制御に基づき、標準入力から得た ACL のメッセージを解析し中間メッセージを作成後、心的状態データベース登録する。

ACL+SL ジェネレータ:動作エンジンの制御に基づき、心的状態データベースの内容をもとに、ACL のメッセージを生成し、標準出力に書き出す。

これらのジェネレータとパーザは全て JAVA2 を使用して実装した。心的状態データベースには JDBC 経由でアクセスする。

3.2 動作概要

入力としてKQMLのメッセージを与えた場合を例にゲートウェイの動作について説明する。入力されるメッセージは以下のものとする。

```
(tell
        :sender s
        :receiver r
        :in-reply-to abc
        :reply-with def
        :language KIF
        :ontology ipu-gateway-ontology-v1
        :content(= a b))
```

まず、動作エンジンが起動され、この際 AADL で記述されたファイル（このとき AADL の内容は(:stdin KQML :stdout FIPA)である）を読み込む。ゲートウェイはこのファイルの情報をもとにKQMLパーザを起動する。標準入力から KQML のメッセージが入力されるとKQML パーザはパフォーマティブを解析し、パフォーマティブとcontentパラメータから以下のようなFPとREに対応した命題を生成する。

FP に対応する命題：B_s(= a b)∧¬B_s (Bif_r(= a b) ∨ Uif_r(= a b))

FP と RE に対応する命題は、パフォーマティブが tell の時、「エージェント s は a = b だと確信していて、かつ、エージェント s は、エージェント r が a = b だと思っているかどうか、あるいは、a = b だと判断できるかどうかということを確信していない」ことを可能前提条件としている。

RE に対応する命題：B_r(= a b)

また、このメッセージを送ることにより、エージェント s は、「エージェント r が a = b だと確信する」ことを期待している。

その後、KQML パーザはその他のパラメータ（sender、receiver、in-reply-to、reply-with、language、ontology）も利用して中間メッセージを作成し、JDBC 経由で心的状態データベースに登録し処理を終了する。その後、動作エンジンはFIPAジェネレータを起動する。FIPAジェネレータはJDBC経由で心的状態データベースから登録されている中間メッセージを受け取り、命題表現部分を解析してコミュニカティブアクトとメッセージの content パラメータを生成する。そして、その他のパラメータ(sender、receiver、in-reply-to、reply-with、ontology)も作成し、language パラメータを sll.5 に変更しそれらを組み合わせてメッセージを生成する。そして、そのメッセージを標準出力に出力し、処理を終了する。出力されるメッセージは以下のようになる。

```
(inform
        :sender s
        :receiver r
        :in-reply-to abc
        :reply-with def
        :language sll.5
        :ontology ipu-gateway-ontology-v1
        :content(= a b))
```

4. おわりに

本研究では、異なるアーキテクチャを持つマルチエージェントシステム間の相互接続を可能とするゲートウェイの実装について報告した。今後は実装したゲートウェイの評価を行い、本ゲートウェイの有効性の確認を行う予定である。

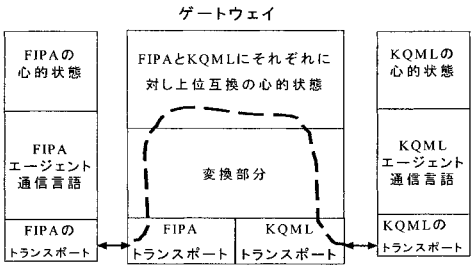

図1　ゲートウェイを利用した相互接続のモデル

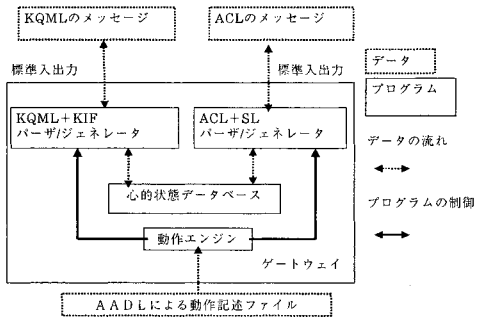

図2　ゲートウェイの実装モデル

参考文献

[1] 須栗 裕樹, 児玉 英一郎, 宮崎 正俊：エージェントシステム間を相互接続するゲートウェイ,情報処理学会第61回(平成12年後期)全国大会講演論文集(3), pp.373-374, (2000).

[2] Hiroki Suguri, Eiichiro Kodama and Masatoshi Miyazaki：GATEWAY THAT INTERCONNECTS HETEROGENEOUS MULTI-AGENT SYSTEMS, Proceedings of the 9th International Conference on Human-Computer Interaction, (to appear).