マルチメディア通信と分散処理 86－33
グループウェア 26－33
（1998. 1. 30）

Agentの形式的記述に関する一考察

山本 潮† 李 殷碩‡ 小野里 好邦†

† 群馬大学工学部情報工学科 / 〒376 群馬県桐生市天神町 1–5–1
‡ 韓国成均館大学工科大学 / 韓国京畿道水原市長安区泉川洞 300

あらまし 近年、様々なエージェントシステムの実現化とともに、それを実装するためのエージェント記述言語とその開発支援環境に関する研究が盛んに行われている。しかし、これらの記述言語や支援環境には段階的詳細化が行えない、実システムまで動作の確認ができない、抽象度の高い記述が不可能といった問題点があった。

本稿では、通信システムの仕様記述において使用される形式記述技法 (FDT) に着目し、これを用いてエージェントを記述するための方法について検討する。FDT の利用することで段階的詳細化や検証などが行えるようになり、より効果的かつ効率的にエージェントシステムを構築することができると期待される。

キーワード マルチエージェントシステム、記述言語、FDT、段階的詳細化、検証

A Study on Specifying Agents in Formal Description Language

Ushio Yamamoto†, Eun-Seok Lee‡ and Yoshikuni Onozato†

† Dept. of CS, Faculty of Engineering, Gunma University
1–5–1 Tenjincho, Kiryu 376, Japan
‡ Faculty of Engineering, Sung-Kyun-Kwan University
300, Chunchun-dong, Jangan-ku, Suwon,
Kyunggi-do 440-746, Korea

Abstract Recently, the researches on agent description languages and it's support environments have been worked in order to make agents which satisfy the user's requirement.

In this paper, we focus on the formal description techniques and it's languages which have been used for describing formal specification of communication systems, and propose agent description method with extended one of FDTs. Using FDT enables for the user to refine the agent description stepwisely and to verify and to valiate the described system automatically. Therefore, it is expected that the user can construct the agent system effectively and efficiently.

keywords Multiagent System, Description Language, FDT, Stepwise Refinement, Verification and Validation

1 はじめに

近年、エージェントという概念がコンピュータ通信ネットワークを利用するユーザを支援する新しいパラダイムとして注目されており、特に情報検索やフィルタリング [3]、電子商取引 [11] など様々な分野でその可能性が確認されている。また、このような人間の代わりに作業を行うエージェントを実際に開発するためのエージェント開発法、特にエージェント記述言語 [1][2] やそれを用いたエージェント開発環境の構築 [4] などが行われている。

しかしながら、上記のようなエージェント記述のための専用の言語やプログラミング言語は機能性に重点が置かれており、学習性や理解性が低く特に今後予想される多種多様なユーザによるエージェント開発を非常に困難とするものである。

さらに、開発されたエージェントが正しく動作するかどうかを確認することは、エージェントの運用と複数のエージェント間の協力作業を想定する際特に重要であるが、動作環境を実現しそこで直接確認することなく開発の初期段階でそれを可能とする効果的な方法がない。今後期待されるエージェント開発支援環境は、(1) ユーザのもつ抽象的なイメージから少しずつ具体的に記述していくことができ、(2) 記述されたエージェントがユーザの意図したように動作するかを実際の環境で動作させる前に確認することが可能で、(3) 記述されたエージェントが実際の環境で動作することができるように変換することができる、ような項目を満足しなければならない。

本稿では、形式的な方法でエージェントの仕様を作成し、実装の段階まで行くことなくエージェントの特性を検証すると共に仕様のレベルでの動作を可能とする。また必要に応じて例えば、他の非エージェントシステムとの連動のため特定のプログラミング言語への自動／半自動変換も可能とする方法論を考案する。ここでの形式記述法は通常通信システムの仕様記述に主として用いられる手法を変更、拡張するものである。提案する形式的記述法をエージェントシステム開発に導入することで、開発初期段階での仕様検証による早めのフィードバックから開発全体の負荷の軽減が期待できる。

以下、2 章では形式的記述によるエージェントシステムの開発法のコンセプトについて述べる。3 章では、形式記述言語として LOTOS をベースにした場合の記述について述べる。4 章では、LOTOS ベースの言語における記述例を示す。最後に 5 章で本稿のまとめと今後の課題について述べる。

2 基本コンセプト

2.1 エージェントの記述要件

エージェントシステムを記述していく場合において、必要な記述要素は以下のようなものがある。

i) エージェントの持つべき機能

ii) ユーザまたは他のエージェントとの通信プロトコル

iii) 行動に必要な知識

iv) 複数のエージェントよる協調行動のための戦略など

以下 i)~iii) について少し考察する。

2.1.1 エージェントの持つべき機能

ここでいう機能とは、エージェント自身が自分以外の他の実体と関連することなく単独で何かを行うことができるという意味である。これは、計算などを除くと基本的に次の 2 つになる。

- ファイルへのアクセス：指定されたファイルを開き、読み書きを行う。
- ある場所から別の場所への移動：ある処理を行うために、現在の場所から適切な場所へ移動する。

これらの基本的な機能を使用する際には、何らかの条件に依存した形式になる場合もあるので、記述においては条件による分岐が記述できなければならない。

2.1.2 通信プロトコル

通信プロトコルは、他の実体 (サーバ、エージェント、人間) との情報交換を行うための手順である。これが通信する相手に対して適切に定義されなければ、その相手とのインタラクションは不可能になる。通信相手が記述時に特定されている場合には、相手の通信プロトコルが分かっていれば記述することは可能であるが、そうでない場合には記述ができないことになる。それゆえ、通信プロトコル部分は実行時に動的に変更できる仕組みが必要となる。

2.1.3 知識

エージェントがより知的に振る舞うためには、そうするための知識が必要となるが、そうでない場合でも少なからず何らかの知識を必要とする。知識と

いっても、信念や意図などメンタル状態と関連する深い知識を表現する場合や、例えばユーザ名や何かの金額などのようなある属性に対する値程度の知識のみを表現する場合もある。

これまで提案されているエージェント記述言語においては、より実装に近い言語ではあまり知識について考慮されておらず、抽象的記述が可能な言語の方が深い知識を記述する傾向にある。

本稿では、まず信念や意図などを扱わない形式で議論を進める。

2.2 エージェント開発支援

先に述べたように、エージェントシステムを効果的に開発していくためには、(1) 段階的に詳細化が可能であること、(2) 記述されたエージェントの動作を実際の環境で動作させる前に確認することが可能であること、(3) 記述されたエージェントがある環境で動作しない場合は、その環境で動作可能な形式に変換することができること、を満足することが必要である。このことから、エージェント開発支援として、次のような構成要素が必要となる。

- Agent Specification Module : ユーザの望むエージェントまたはエージェントシステムを記述することを支援
- Agent Testing Module : FDT で記述されたエージェントまたはエージェントシステムがユーザが意図したように動作するかを確認することを支援
- Agent Translation Module : 実際の環境でエージェントが動作することができるように、記述されたエージェントを別の形式に変換

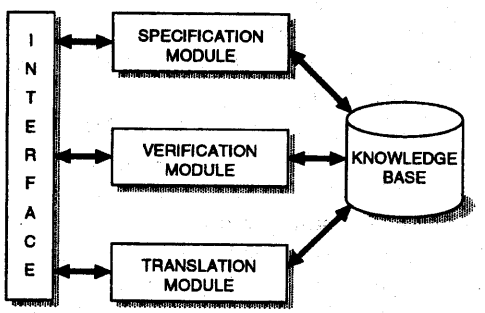

図 1: エージェント開発支援に必要な構成要素

上記の構成要素を効果的に実現するために、本稿ではエージェントの記述法として形式的記述技法 (FDT : Formal Description Technique)[5] を利用する。形式的記述技法は、主に通信システムの仕様記述において用いられる方法であり、システムの仕様を形式的に記述することを可能とし、またプログラミング言語での記述よりも抽象的にシステムを記述することを可能とする。記述されるものが形式的になるため、それをコンピュータで扱いやすくなり、その結果として (半) 自動的な検証や実装言語への変換なども行うことができるようになり、それに関する研究も行われてきている [6]-[9]。また、抽象的なシステム記述から徐々に具体的なシステム記述へと記述の抽象度を変えながら記述する段階的詳細化を行うことが可能であり、人間のもつ初期の曖昧なイメージから、少しずつ具体化するというプロセスを行うことができる。

それゆえ、形式的記述技法を用いてエージェントシステムを記述していくことにより、設計者はより詳細な設計を意識することなく自分のイメージから記述してくことができ、エージェントの動作を確認するために、実際の環境で動作させる前にそれらの動作を検証することが可能となり、より安全な開発を行うことができるようになる。

以下、FDT を考慮に入れて上述のエージェント開発支援の構成要素について説明する。

2.2.1 Agent Specification Module

Agent Specification Module はエージェントの動作を記述するためのツールを提供する。ユーザの負荷を軽減するために、エージェントのもつべき基本的な機能であるファイルのアクセスと移動に関してはあらかじめ定義されている。これらの機能を定義するため、エージェントとその環境に関するモデルを示す。

エージェントが動作する環境には、エージェントが活動するための場が存在し、その中でのみエージェントは動作することができるとする。エージェントは同じ場の中にいるエージェント、サーバ、あるいはユーザなどとは直接通信を行うことができる。しかしながら、エージェントのもつべき基本的な機能に関しては場を介して行う。例えば、エージェントがある場所から別の場所へ移動する場合を考える (図 2)。このとき、エージェントは自分が今いる場に対して次にどの場に移動するかを通知する。この通知によって場は移動のための処理 (途中結果の保存や移動しやすいように変換するなど) を行い、エージェントを通知された次の場へ移動させる。次の場では、移動してきたエージェントを自分の場に置き、その結果エージェントはその場においてまた動作を始める。

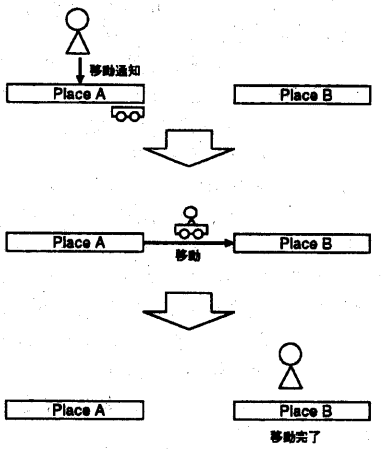

図 2: エージェントの移動のモデル

また、前述したように、他の実体との通信を行わなければならない場合には、その相手との通信プロトコルを記述しなければならない。これに関しては、何らかの情報がなければ記述することができない。ここでは、(1) 双方用の通信プロコトルが用意されている場合には、それに必要な手続き名やパラメータなどをユーザに示して、必要なところにそれだけを記述する、(2) 相手側のみの通信プロトコルが分かっている場合には、それに基づいて自分の方のプロトコルを自動合成することが可能であるので [12]、ユーザによけいな負荷を与えることはない。

2.2.2 Agent Testing Module

Agent Testing Module は記述されたエージェントがユーザの望んだ動作をするかどうかを実際に動かす前に確認するためのモジュールである。FDT に基づく検証では、次のようなエラーを発見することが可能である。

- 論理エラー発見：例外受信、通信デッドロック、チャネルオーバーフローなどが発見される。
- 意味エラー発見：自動生成されたテストシーケンスを使うことにより、ユーザの意図しない動作を行っているエラーが発見される。

もしある言語で記述されたエージェントと相互作用のあるエージェントやサーバなどに関して同じ言語による記述が存在するのであれば、それぞれを単独に検証するだけでなく、全体として整合性のとれた動作を行うことができるかどうかを確認することも可能である。これにより、これまでは動かすまで、あるいは動かしても正しくエージェントが動作するかどうか判断することができなかったことがなくなり、より安全かつ効率的に開発を行うことができるようになる。

2.2.3 Agent Translation Module

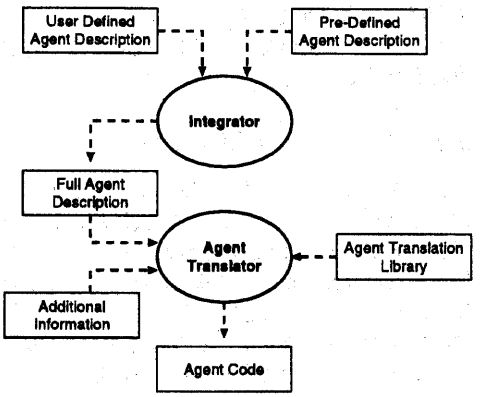

図 3: Agent Implementation Module

新たに記述されたエージェント (システム) が実際のコンピュータネットワークにおいて動作するためには、(1) エージェントの記述を解釈して実行することができるようなインタプリタあるいは仮想機械を各計算機に用意する、(2) 計算機の現状況のみで実行可能にするために、エージェントの記述を別の形式に変換することが必要となる。Agent Translation Module はそれを行うためのモジュールである。

図 3にこのモジュールの実行プロセスを示す。このプロセスは 2 ステップからなっており、それぞれエージェント記述統合ステップとエージェント変換ステップである。最初のステップでは、ユーザが記述した部分と既に定義されているエージェント記述の部分とを統合して完全なエージェントの記述とする。このステップにおける入力情報としては、ユーザが定義したエージェント記述、定義済みのエージェント記述である。出力情報としては、完全なエージェント記述である。

二番目のステップであるエージェント変換ステップでは、最初のステップで統合されたエージェント記述をある形式の言語へ変換する。このステップでは、入力として与えられるエージェント記述に対応する詳細な情報が Agent Translation Library から付加される。このステップの結果として、インタプリタのない計算機上でのエージェントの実行を可能にするエージェント記述の形式が出力される。例え

ば、現在多くの環境において使用される JAVA のコードに変換すれば、多くの計算機上で実行させることが可能になると考えられる。

3 LOTOS への適用

本節では、FDT の一つである LOTOS に対する適用例を示す。

3.1 LOTOS

LOTOS[10] は国際標準化された FDT の 1 つである。その基本コンセプトは、システムは外部から観測可能な動作を定義することによって記述することができる、というものである。内部状態は気にせずに外部との信号やサービスプリミティブなどの入出力のみを記述するので、高い抽象度をもって仕様を記述することができる。

LOTOS を用いた通信システム開発の支援を行うための方法論やツールなどは、仕様記述支援、検証、C などのプログラミング言語への変換など多岐に渡って研究・開発されており、エージェントの開発においても LOTOS をベースにすることによりこれらの成果を利用することができる。

4 エージェントの記述例

ここでは、先に述べた手法を用いて列車とホテルの予約を行うエージェントを記述する例について述べる。

エージェントのおおまかな処理手順は次の通りである (図 4)。

1. ユーザから要求として移動・宿泊の日付を獲得する (簡単にするため、1 泊とする)。
2. 要求をもって列車予約のサーバへ移動し、予約を行う。
3. 予約が完了した場合はホテル予約のサーバへ移動し、予約を行う。列車の予約が不可の場合は、もとの場所に戻り結果をユーザに伝えて終了する。
4. ホテルの予約が完了した場合はもとの場所に戻り、結果をユーザに伝えて終了する。そうでない場合は、列車予約サーバへ移動して予約をキャンセルし、もとの場所に戻ってユーザに結果を伝えて終了する。

この動作を行うエージェントの記述に関して、サーバとの通信プロトコルが公開されている場合に

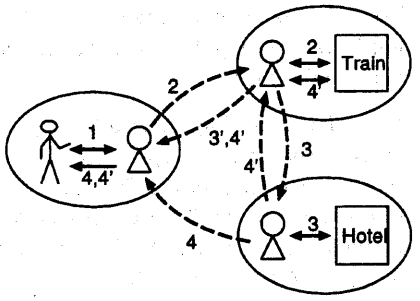

図 4: 列車・ホテルの予約

は、実際にユーザが記述する部分は図 5に示される部分であり、既定義部分は図 6に示される通信プロトコルの部分とエージェントの移動の部分になる。

5 まとめ

本稿では、ユーザを支援するエージェントシステムの開発を支援するための方法として、通信システムの仕様記述に用いられる形式的記述技法 (FDT) を用いた支援法について述べた。FDT はシステムを抽象的にかつ形式的に記述することが可能であるため、ユーザの曖昧なイメージから徐々に具体的な設計へと詳細化することができ、かつ (半) 自動的な検証・実装言語への変換などを行うことができる。これを用いてエージェントシステムを開発することにより、これまでの問題点であった段階的詳細化や実環境での実行前の動作確認を行うことができるようになり、安全かつ効率的な開発を行うことができるようになると期待される。

現在、開発支援環境のプロトタイプをワークステーション上に実装を行っている段階である。これを完成させることと、知識や協調作業に関する記述などの検討が今後の課題である。

参考文献

[1] D. T. Chang and D. B. Lange, "Mobile Agent: A New Paradigm for Distributed Object Computing on the WWW," OOPSLA '96 Workshop, 1996.

[2] F. G. McCabe, K. L. Clark, "April – Agent PRocess Interaction Language", In *Lecture Notes in Artificial Intelligence, Intelligent Agents*, pp. 324 – 340, 1994

[3] T. Oates, M. V. N. Prasad and V. R. Lesser, "Cooperative Information Gathering," UMass Computer Science Technical Report, 94-66, 1994.

[4] D. L. Martin, A Chayer and G.L. Lee, "Development Tools for the Open Agent Architecture," Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp.387–404, April 1996.

[5] ISO, "Information technology — Open Systems Interconnection — Guidelines for the application of Estelle, LOTOS and SDL," ISO/IEC TR 10167, 1991.

[6] E.S. Lee, K. Mori, N. Shiratori and S. Noguchi, "Construction and Implementation of a Specification Environment SEGL based on G-LOTOS," Transaction of IPSJ, pp. 314–424, 1991.

[7] N. Shiratori, H. Kaminaga, K. Takahashi and S. Noguchi, "A verification method for LOTOS specification and its application," in Protocol Specification, Testing and Verification IX, ed. E. Brinksma et al., pp., North-Holland, Amsterdam, 1990.

[8] ISO, "OSI Conformance Testing Methodology and Framework Part : General Concepts," ISO/DP9649.

[9] Nomura,S. Hasegawa,T and Takizuka,T., "A LOTOS Compiler and Process Synchronization Manager," Proc. of 10-th IFIP WG6.1 International, Symposium on Protocol Specification, Testing and Verification, 1990.

[10] ISO 8807, "Information Processing Systems — Open Systems Interconnection — LOTOS — A formal description technique based on the temporal ordering of observational behaviour," 1989.

[11] J.G. Lee, J.Y. Kang and E.S. Lee, "ICOMA: An Open Infrastructure for Agent-based Intelligent Electronic Commerce on the Internet," Proc. of 1997 International Conference on Parallel and Distributed System, 1997.

[12] B. B. Bista, Z. Cheng, A. Togashi and N. Shiratori, "A New Approach for Protocol Synthesis Based on LOTOS," IEICE Trans. on Fundamentals, Vol. E77-A, No. 10, pp. 1646 – 1655, 1994

```
specification agent[user,tp,hp,place]:exit

(* Move and Comm processes are imported *)

type Host is
  sorts Host
  opns tsrv,hsrv,myhost :-> Host
endtype

behaviour

User_Req[user] >> accept id:String,date:String in
  ( Reserve_Trn[tp](id,date) >> accept res:Ack in
      ( [res = OK]-> Reserve_Htl[hp](id,date) >> accept res:Ack i
                       ( [res = OK]-> Result[user](OK)
                         []
                         [res = NG]-> Cancel_Trn[tp](id,date)
                                        >> Result[user](NG) )
        []
        [res = NG]-> Result[user](NG) ))

where

  process User_Req[user]:exit(String,String):=
    user?id:String?date:String; exit(id,date)
  endproc

  process Reserve_Trn[tp](id:String,date:String):exit(Ack):=
    Move[place](tsrv) >> Comm_Trn[tp](Reserve,id,date)
                      >> accept ack:Ack in exit(ack)
  endproc

  process Cancel_Trn[tp](id:String,date:String):exit(Ack):=
    Move[place](tsrv) >> Comm_Trn[tp](Cancel,id,date)
                      >> accept ack:Ack in exit(ack)
  endproc

  process Reserve_Htl[hp](id:String,date:String):exit(Ack):=
    Move[place](hsrv) >> Acc_HRC[tp](Reserve,id,date)
                      >> accept ack:Ack in exit(ack)
  endproc

  process Result[user](ack:Ack):exit:=
    Move[place](myhost) >> user!ack; exit
  endproc
endspec
```

図 5: エージェントの記述例 (ユーザ記述部分)

```
type Comm_act is
  sorts Comm_act
  opns Reserve,Cancel :-> Comm_act
endtype

type  Ack is
  sorts Ack
  opns  OK,NG,Ready :-> Ack
endtype

process Comm_Trn[sap](act:Comm_act,id:String,date:String):exit(Ack):
  ( [act = Reserve]->
      sap!act!id!date; sap?ack:Ack; exit(ack)
    []
    [act = Cancel]->
      sap!act!id!date; sap?ack:Ack;
      ( [ack = OK]-> exit
        []
        [ack = NG]-> Comm_Trn[sap](act,id,date) ))
endproc

process Acc_HRC[sap](act:Comm_act,id:String,date:String):exit(Ack):=
  ( [act = Reserve]->
      sap!act; sap!Ready; sap!id!date; sap?ack:Ack; exit(ack)
    []
    [act = Cancel]->
      sap!act; sap!Ready; sap!id!date; sap?ack:Ack;
      ( [ack = OK]-> exit
        []
        [ack = NG]-> Acc_HRC[sap](act,id,date) ))
endproc

process Move[place](location:String):exit:=
  place!location; exit
endproc
```

図 6: エージェントの記述例 (既定義部分)