L-073

論理型モバイルエージェントフレームワーク Maglog
Maglog: A Logic-based Mobile Agent Framework

本村 真一 † **川村 尚生** ‡ **菅原 一孔** ‡
Shinichi MOTOMURA Takao KAWAMURA Kazunori SUGAHARA

1. はじめに

コンピュータネットワークの発達に伴い，ソフトウェアアーキテクチャは一つの仕事を複数のコンピュータの共同作業として行う方向へ向かっている．このような分散環境はネットワークのスピードやトポロジの変化，構成要素であるコンピュータの追加，更新，削除が時々刻々と起こるオープンシステムである．オープンシステムにおけるソフトウェアは一つの固定的な形を静的に持つことはできず，多数のコンポーネントから動的に構成されるような仕組みが必要になる．このようなソフトウェアアーキテクチャとしてモバイルエージェントシステムが注目を集めている．モバイルエージェントシステムとは，ユーザの代理となるエージェントと呼ばれるプログラムが，ネットワーク上に複数存在するコンピュータを移動しながら与えられた問題の解決をはかるシステムである．モバイルエージェントの自律性を実現する要素として，推論機構やプランニング機構を持つことが考えられる．本論文では，これらの機構の記述に適した論理型言語 Prolog に基づいたモバイルエージェントフレームワーク Maglog（Mobile AGent system based on proLOG の略）[1, 2] を提案する．同様のモバイルエージェントフレームワークとして Jinni[3] や MiLog[4] が提案されている．これらと比較して，Maglog では次の特徴を実現することにより，モバイルエージェント環境下における Prolog プログラムの容易な記述性を実現する．

- エージェント間通信，エージェントの移動及び手続きライブラリの利用をフィールドという単一の要素で実現した．
- エージェントの移動は往復を対にし，併せて強マイグレーションを実現したことで，プログラマによるエージェントの位置情報の管理を不要とした．ここで言う強マイグレーションとは，エージェントが移動した後，移動前の状態から継続して実行を再開できることを言う．これに対して，移動後，特定の手続きから実行を継続する必要があることを弱マイグレーションと呼ぶ．

2. Maglog の構成

図 1 に Maglog の概要を示す．Maglog の基本要素であるエージェント，エージェントサーバ，フィールドの概要を次に示す．

- エージェント は Prolog を実行するプログラムであり，Prolog の手続きを保持している．

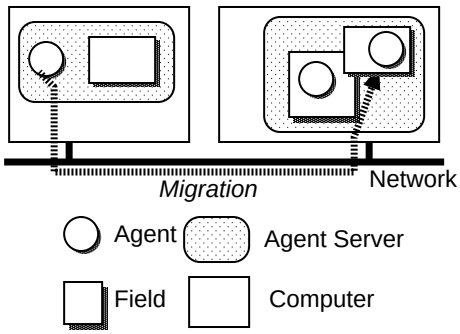

図 1: Overview of Maglog

- エージェントサーバ はエージェントの実行環境である．エージェントの実行やフィールドの保持，作成，削除，エージェントの移動に関する機能を提供している．
- フィールド はエージェントが実行する手続きを保持しており，静的なフィールドと動的なフィールドがある．

2.1 エージェント間通信

Maglog のエージェント間通信はエージェント同士が直接通信するのではなく，フィールドを介した間接通信である．Maglog ではそのために次の組み込み述語を提供している．

```
fasserta(Clause,Field)
fassertz(Clause,Field)
fretract(Clause,Field)
```

これらの組み込み述語により手続きをフィールドに対して追加，削除することができる．これら述語のそれぞれの動作は Prolog の `asserta`，`assertz`，`retract` に相当する．これらの述語を利用してエージェント間の同期，非同期通信を実現するために，エージェントには，フェイルモードとブロックモードと呼ぶ 2 つの動作モードを用意している．これらの動作モードは `fretract` 実行時にそれぞれ次の動作を行う．

1. フェイルモードではフィールドに一致する手続きがない場合，手続きの削除に失敗し，Prolog における fail が起きる．このモードは非同期通信を提供する．
2. ブロックモードではフィールドに一致する手続きがない場合，一致する手続きが追加されるまで実行が中止される．このモードは同期通信を提供する．

†鳥取大学大学院工学研究科
‡鳥取大学工学部

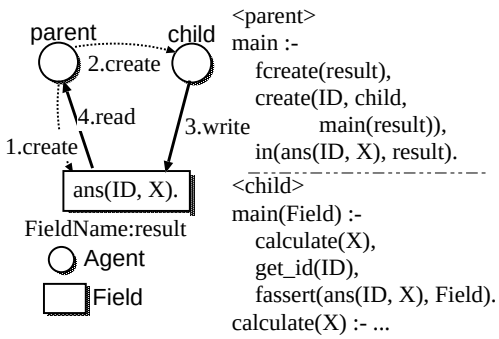

図 2: Agents can communicate synchronously through a field.(block mode)

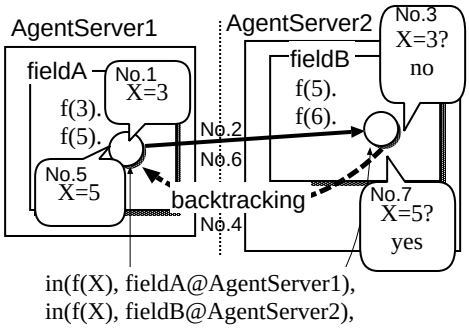

図 3: Backtracking and unification between two computers.

図 2 にエージェントが同期通信を行なっている例を示す．例において，親エージェントはフィールドを作成した後，子エージェントを作成している．作成された子エージェントは処理を実行し，その結果を先のフィールドに格納する．そして親エージェントはフィールドから子エージェントの実行結果を取得している．このように，フィールドを利用することでエージェント間通信を宣言的に記述することができる．

2.2 エージェントの移動

Maglog ではエージェントの移動にもフィールドを利用する．

別のフィールドに移動し手続きを実行する次の組込述語を提供している．

```
in(Goal,Field)
```

第二引数の `Field` が移動先のフィールドを指しており，他のエージェントサーバ上のフィールドを指す時は次のように記述する．

```
in(Goal,Field@AgentServerAddress)
```

ここで `Field` はフィールドを識別する名前である．また `AgentServerAddress` はエージェントサーバの識別子であり，エージェントサーバが稼動するコンピュータの IP アドレスとエージェントサーバの名前の組み合わせから構成されている．この述語により，エージェントは `Goal` で指定された手続きを移動先フィールドで実行する．そして実行が終了すると自動的に移動前のエージェントサーバに戻る．`fasserta`，`fassertz`，`fretract` についても同様に，第二引数部分のフィールド指定においてエージェントの移動が行える．

図 3 に 2 つのコンピュータをまたがってユニフィケーションとバックトラッキングが行なわれている例を示す。最初，fieldA に存在するエージェントが fieldB に移動しマッチングを行っている．次に fieldB においてマッチングが fail したため，エージェントは自動的に fieldA に戻り，次の節を用いて再度 fieldB に移動してマッチング処理を行っている．この例で示しているように，エージェントの移動についても宣言的に記述することができる．これは，エージェントの移動が自動的であるため，プログラマがプログラムの記述においてエージェントの移動開始とその戻りを意識する必要がなく，またエージェントの移動において強マイグレーションを実現しているためである．

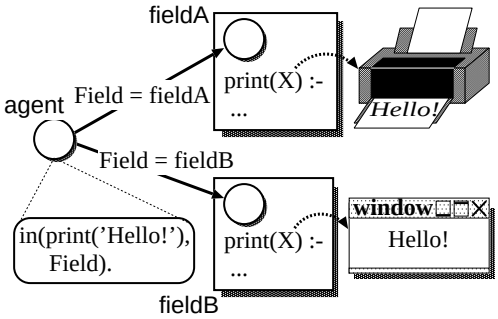

図 4: Dynamic Change of agent's behavior through entering a field.

2.3 手続きライブラリの利用

フィールドは Prolog の手続きを保存しておくことができる．この機能により，事前に作成しておいた手続きをライブラリとして提供することができる．この特徴により，プログラマは組込述語以外の手続きを利用する場合も，新たな述語として手続きを作成することで Prolog 構文を用いてプログラムを記述できる．

また，フィールドに手続きを保存できる特徴を利用して，エージェントの振る舞いを動的に変化させることができる．この例を図 4 に示す．エージェントが fieldA に存在する時に手続き print(X) を実行するとプリンタに出力され，fieldB に存在する時はディスプレイに出力されている．エージェントが，fieldA と fieldB という異なるフィールドで同じ名称の手続きを実行した時に，異なる振る舞いが行なわれていることを示している．

3. 実装

Maglog の実装言語としては Java を用いている．Maglog に必要とされるマルチプラットフォーム，ネットワーク対応，その他豊富なクラスライブラリが提供されてい

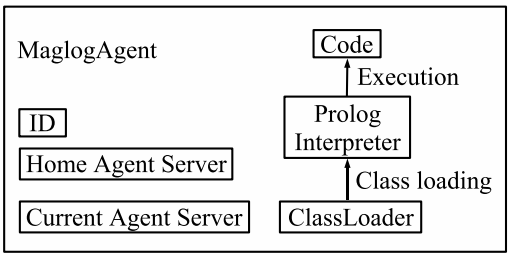

図 5: Structure of an agent.

るため最適であると考えている．そしてエージェントの実行エンジンとしては，Prolog から Java へのソースコードトランスレータである Prolog Café[5] を拡張している．Prolog Café では Prolog から Java の呼び出しが可能であり，この機能を利用して，Maglog ではフィールドに手続きライブラリを構築することが可能となっている．

3.1 エージェント

エージェントの構成を図 5 に示す．ID はエージェントの識別子を示している．Home Agent Server はエージェントが作成されたエージェントサーバを示しており，これはエージェントが移動した際における戻り先のエージェントサーバである．Current Agent Server は現在エージェントが存在するエージェントサーバを示している．

エージェントは Prolog 手続きを実行するための Java プログラムであり，コンパイルされた Java クラスである．エージェントは移動に際して，エージェントが実行する Prolog 手続きをコンパイルした Java クラスを除いた，Prolog interpreter や Class Loader 等の最小限の Java クラスだけを移動し，必要に応じて Java クラスを取り寄せるオンデマンドクラスローダを内蔵している．これによりエージェントの移動にかかるコストを軽減している．

また JVM（Java Virtual Machine）ではセキュリティ上の理由から実行領域を参照，改変する API が用意されていない．そのため，通常，エージェントの移動は弱マイグレーションになる．Maglog では，エージェントの実行を行う Prolog インタープリタも含めて移動することで，強マイグレーションを実現している．

3.2 エージェントサーバ

エージェントサーバはエージェントの実行環境であり，エージェントの実行や移動のための機能を提供している．移動の仕組みとしては RMI を用いたものと XML-RPC を用いたものを提供している．エージェント移動の実行速度を重視する場合は RMI が利用できる．しかし，ファイアウォール等により RMI の利用が制限されている環境では，ネットワークプロトコルとして HTTP を利用した XML-RPC が利用できる．ここでは特に XML-RPC を利用した移動の詳細を述べる．エージェントの移動は，エージェントの直列化と RPC（Remote Procedure Call）により実現している．次に移動の手順について示す．

1. エージェントが移動を開始する際，エージェントサーバはエージェントの動作を中止し，エージェントを XML 文書に変換することで直列化する．
2. エージェントサーバは，移動先のエージェントサーバに対して，先の XML 文書を引数とした RPC を実行する．
3. RPC により呼び出されたエージェントサーバは，受け取った XML 文書からエージェントを復元し，エージェントの実行を再開する．
4. エージェントの実行が終了すると，エージェントサーバは先のエージェントを再度 XML 文書に変換し，呼び出し元のエージェントサーバの RPC を呼び出す．こうしてエージェントの往復が完了する．

RMI についても同様に直列化と RPC により移動を実現している．

3.3 フィールド

フィールドにはスタティックフィールドとダイナミックフィールドの 2 種類がある．ただし，スタティックフィールドには Java クラスにコンパイルされた Prolog 手続きや Java の実行クラスのみ保存することができ，動的に書き換えることはできない．その反面実行速度に優れており，また手続きライブラリとしても利用することができる．ダイナミックフィールドは，エージェント同士がデータ交換を行う共有スペースとして利用できる．

先にも述べたように，エージェント間の同期通信もフィールドを用いて実現している．ブロックモードによる同期通信時において `fretract` を実行した際，ユニフィケーション可能な手続きが存在しなかった場合，Java スレッドの wait を実行し該当する手続きが追加されるまでエージェントの実行を中止する．`fasserta` もしくは `fassertz` が実行されると，Java スレッドの notifyAll を実行し，wait により実行を中止していたエージェントが再びユニフィケーションを試みる．

4. 他システムとの連携

Maglog は Prolog を拡張しているため，数値演算や GUI のような処理の記述には向かない．そのため他の言語及びシステムとの連携を行えることが重要である．Maglog から Java の利用においては，スタティックフィールドを利用することで容易に連携することができる．また Maglog エージェントは Java クラスであるため，Java から呼び出すことも簡単である．その他，他システムとの連携を行うために，エージェントサーバは XML-RPC を利用して次の機能を提供している．

- エージェントの作成と削除
- ダイナミックフィールドの作成と削除
- ダイナミックフィールドに対する手続きの追加と削除
- ダイナミックフィールドのリストの取得
- エージェントのリストの取得

5. Maglog の応用

Maglog の応用として次に示すアプリケーションを開発している．

1. P2P e-Learning System[6]

   Maglog を用いて P2P（Peer-to-Peer）型の e-Learning システムを開発した．このシステムには 2 つの特徴があり，ひとつは P2P アーキテクチャを採用することで拡張性と耐障害性の向上を図っている．もうひとつの特徴としてシステムの構成要素であるエージェントが問題データだけを持つのではなく，採点機能やチャット機能を有している．図 6 にユーザインターフェースの画面を示しているが，これは Squeak により記述している．この e-Learning システムは 2,000 行の Maglog コードと 4,000 行の Squeak コードで実現している．

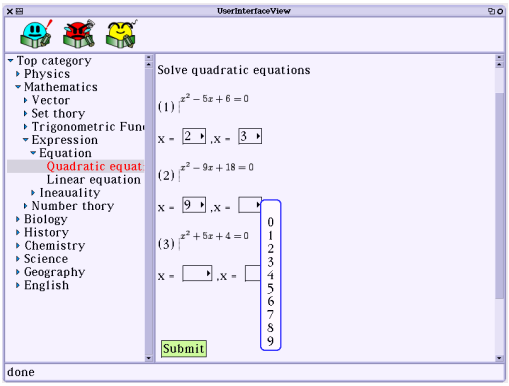

図 6: P2P e-Learning system.

2. スケジュール調整アプリケーション [7]

   エージェントが移動し，各々のスケジュールを参照することで，人手を介さず自動的に会議予約等スケジュールの調整を行うアプリケーションを開発した．このアプリケーションは 400 行の Maglog コードと GUI として 4,000 行の Java コードで実現している．

3. HECS System[8]

   Maglog は HECS（Heterogeneous Constraint Solving）システムの一部に使用されている．HECS は IPA（The Information-technology Promotion Agency）の平成 14 年度，平成 15 年度における未踏ソフトウェア創造事業に採択されている，分散環境下における協調制約解消システムである．

6. おわりに

論理型モバイルエージェントフレームワーク Maglog を提案した．エージェント間通信及びエージェントの移動をフィールドを用いて実現し，特にエージェントの移動については，強マイグレーションとエージェントの自動的な往復を組み合わせて実現した．これらの特徴により，論理型言語を用いたモバイルエージェントの記述が容易に行えることを示した．しかし，Maglog では移動を往復にしたことにより，ネットワーク障害に際してエージェントの消失が発生する頻度が多くなる可能性がある．今後，Maglog フレームワークにこのような障害に対応できる機能の追加を行う予定である．

参考文献

[1] Kawamura, T., Kinoshita, S., Sugahara, K. and Kuwatani, T.: A Logic-based Framework for Mobile Multi-Agent Systems, *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (Hexmoor, H.(ed.)), pp. 754–759 (2003). Boston, Massachusetts, USA.

[2] Kawamura, T., Kinoshita, S. and Sugahara, K.: Implementation of a Mobile Agent Framework on Java Environment, *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems* (Gonzalez, T.(ed.)), pp. 589–593 (2004). MIT, Cambridge, USA.

[3] Tarau, P.: Inference and Computation Mobility with Jinni, *The Logic Programming Paradigm: a 25 Year Perspective* (Apt, K., Marek, V. and Truszczynski, M.(eds.)), Springer, pp. 33–48 (1999).

[4] Fukuta, N., Ito, T. and Shintani, T.: MiLog: A Mobile Agent Framework for Implementing Intelligent Information Agents with Logic Programming, *Proceedings of the 1st Pacific Rim International Workshop on Intelligent Information Agents*, pp. 113–123 (2000).

[5] Banbara, M. and Tamura, N.: Translating a Linear Logic Programming Language into Java, *Proceedings of the ICLP'99 Workshop on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages* (M.Carro, I.Dutra et al.(eds.)), pp. 19–39 (1999).

[6] Kawamura, T., Kinoshita, S. and Sugahara, K.: A Mobile Agent-Based P2P e-Learning System, *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems* (Gonzalez, T.(ed.)), pp. 873–877 (2004). MIT, Cambridge, USA.

[7] Kinosita, S., Kawamura, T. and Sugahara, K.: Mobile Agent based Schedule Arrangement System, *Proceedings of the 5th IEEE Hiroshima Student Symposium (HISS)*, pp. 205–206 (2003).

[8] Banbara, M., Tamura, N., Inoue, K., Kawamura, T. and Tamaki, H.: Java Implementation of a Distributed Constraint Solving System, Exploratory software project, Information-technology Promotion Agency Japan (2003).