モバイルエージェントの経路記述と選択機構

佐 藤 一 郎†

モバイルエージェントに処理を依頼する際には，複数のモバイルエージェントの中から移動先ホストにおける動作内容だけでなく，移動経路についても処理要求を満足するエージェントを選ぶ必要がある．そこで本論文では移動経路を基準とするモバイルエージェントの選択手法を提案していく．この手法は移動経路を記述する言語と，その言語により記述された式を比較する代数的順序関係から定式化される．前者はプロセス代数をもとにした仕様記述言語であり，後者はモバイルエージェントが所定のホストに効率的かつ要求された順序で移動できるかを判定するものとなる．そして，この手法のプロトタイプシステムを Java 言語によるモバイルエージェントシステム上に実装した．モバイルエージェントはこの言語による経路記述を実行して移動可能となり，さらにモバイルエージェントへのタスク割当て方法として，タスク依頼の要求経路を満足するモバイルエージェントを選択する機構をこの順序関係を利用して実現した．

Specification and Selection for Agent Mobility

Ichiro Satoh†

When assigning a task to mobile agents, one of the agents has to be selected according to not only their application-specific behavior but also their itineraries. This paper presents an approach for selecting mobile agents according to their itineraries. The approach offers a process algebra-based language for specifying the itinerary of mobile agents and an algebraic order relation between two itineraries specified as terms of the language. The relation can decide whether or not mobile agents can satisfy the itinerary required by a given task, in the sense that the agents can migrate to all the hosts required by the task in a permissible order specified by the task. A prototype implementation of this approach was built on a Java-based mobile agent system. It enables each mobile agent to specify its itinerary as a term of the language and to migrate over a network according to only the itinerary. Also, when it receives the request of a task, it can select a suitable mobile agent to perform the task by using the order relation.

1. まえがき

モバイルエージェントは複数のホストを巡回しながら処理を行うソフトウェアであり，分散システムにおけるコンピュータ間通信回数・遅延の削減や，通信切断を含む耐故障性の実現に有用な実装技術とされる．一方でモバイルエージェントはソフトウェアエージェント技術の 1 つとして数えられることがあるが，他のソフトウェアエージェント技術と異なり，自律性，反応性，自発性，学習，協調性などの知的特性は必ずしも要求されない．これは知的特性を実現するプログラムのサイズや実行コストが大きくなり，ホスト間移動コストの増加の原因になるからである．さらにモバイルエージェントはその移動先ホストにおいて利用できるプロセッサ時間やメモリ空間に制限が加えられることも多い．同様の理由により，個々のモバイルエージェントが実行時に移動先を発見することや，効率的な移動経路を生成できるとは限らない．仮に効率的な移動経路を生成できるとしても，その生成処理に必要なプログラムサイズおよび実行コストは小さくない．さらにモバイルエージェントの典型的な応用事例であるネットワーク管理システムにおいて，エージェントは複数のネットワーク機器を巡回して監視・制御を行うが[8]，通信切断やネットワーク機器の故障により経路情報が不完全であることがあり，動的に移動経路を発見・生成できないことがある．したがって，多数のホストを巡回しながら処理を行うモバイルエージェントの場合，特定ネットワークを前提に設計されることや移動経路があらかじめ与えられることも多い．また，セキュリティ上の理由からモバイルエージェントに移動先ホスト名や移動経路を埋め込み，所定以外のホストへの移動を制限することもある．以上からモバイルエージェントによる分散処理では汎用的かつ高機能な

† 国立情報学研究所/科学技術振興事業団
National Institute of Informatics/Japan Science and Technology Corporation

モバイルエージェントを利用できるとは限らず，特定の用途やネットワーク，移動経路に最適化されたモバイルエージェントを適宜選択することが重要となる．

一般のソフトウェアエージェントではタスク割当て機構を含めてエージェントの選択手法が数多く研究されているが，モバイルエージェントはホスト間移動性を持つため，これらの既存の選択手法をそのまま利用することはできない．また，モバイルエージェントを前提とした選択手法も必ずしも整備されていない．そこで本論文ではモバイルエージェントの選択方法，特に移動経路により最適なエージェントを選ぶ方法を定式化し，それに基づくタスク割当て機構を扱っていく．

以降の論文構成を述べる．次章においてモバイルエージェントモデルと選択手法を述べる．3 章では経路記述言語と選択手法の定式化を，4 章ではプロトタイプシステムの実装概要を，5 章では応用事例を説明する．6 章では関連研究を，7 章では本論文のまとめと課題を示す．

2. 方　　針

モバイルエージェントにタスク処理を依頼する場合，そのタスク処理に必要なプログラムを持っていることに加えて，そのタスクの実行が要求されているホストに移動可能で，さらにその移動順序についてもタスク処理の要求を満足するモバイルエージェントを選ぶ必要がある．なお，プログラムの処理内容によるエージェントの選択は既存のソフトウェアエージェントの選択手法と共通するところが多いことから，本論文ではモバイルエージェント特有となる移動経路による選択を対象とし，処理内容による選択は扱わない．

2.1 移動経路による選択

モバイルエージェントに求められる移動経路はタスク依頼の内容によって相違する．たとえば，モバイルエージェントの典型的な応用事例である遠隔情報検索では，モバイルエージェントを複数のデータベースサーバに巡回させて，各サーバから情報を収集させる．このとき収集した情報をエージェント内部に単純に蓄積させるだけであればホストの移動順序と検索結果は独立する．したがって所定のサーバに移動できるエージェントを選択すれば十分であり，各サーバへの移動順序や移動回数は任意となる．一方，あるデータベースにおける検索結果を別のデータベースに反映させる場合は移動順序が処理結果に影響を与える可能性がある．この場合，所定のホストへの移動可能性だけでは不十分で，その移動順序もタスク依頼の要求を満足するエージェントを選ぶ必要がある．さらに，移動経路自体が処理結果に依存することがある．たとえば上述の遠隔情報検索の場合，あるデータベースサーバにおける検索結果により次に移動するサーバを変更することがある．

2.2 移動経路の定式化

モバイルエージェントは Java 言語などの汎用プログラミング言語で記述されたプログラムである．したがって，そのプログラムから移動経路を正確に抽出することは困難となる．そこで，移動経路を記述するための専用言語を導入し，各モバイルエージェントはこの言語による記述式をそれ自身の移動経路に関する仕様として保持することにする．この言語は CCS [10] や π-calculus [11]，ACP [2] などのプロセス代数（またはプロセス計算）をもとにして定式化され，ホスト名の列として移動先ホストとその移動順序を記述していく．また，処理内容による移動先ホストの変更はプロセス代数の選択演算子を利用して表現される．これは移動先候補ホスト名による経路分岐となり，一方のモバイルエージェント側プログラムに対してはこの移動先候補から 1 つのホストを選択・移動する API を導入して，エージェント側の処理内容をそれ自身の移動経路に反映できるようにする．ただし，このときそれ自身の経路仕様以外への移動は制限する．さらに，タスク依頼がエージェントに要求する移動経路もこの言語によりあらかじめ記述されていると仮定する．このときタスク依頼の要求記述では移動先ホストの変更や移動順序，移動回数などへの制限や許容範囲の表現性も必要となるが，ここではプロセス代数に非決定的演算子を拡張することにより対応していく．

2.3 モバイルエージェントの選択

本論文の目的はモバイルエージェントのタスク割当てを目標に，モバイルエージェントの選択手法を提案することにある．ただし，エージェントへの移動要求は多様かつ複雑となることから，厳密性を持った手法が望まれる．そこで，本論文では各エージェントの移動経路がタスク依頼の要求を満足するか否かを調べる代数的な順序関係を定式化し，これを利用してエージェントの選択が厳密に行えるようにする．なお，この手法の実装は所定のホスト上に用意される**エージェントプール**と呼ぶ機構により実現される（**図 1**）．これは先に述べた特定の用途やネットワーク，移動経路に特化した多様なモバイルエージェントの待機場所となるとともに，外部環境からタスクの実行依頼を受け取る．そして，この代数的順序関係に基づいてタスク依頼の要求を満足する移動経路を持ち，かつ移動効率の良いモバイルエージェントを選択するものである．

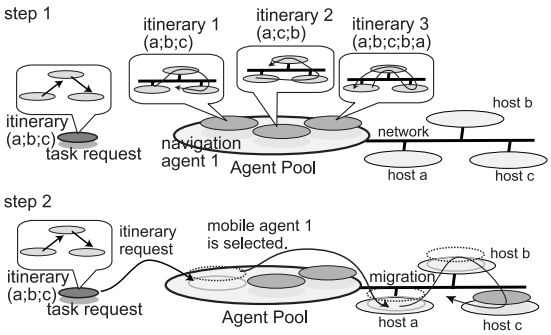

図 1 システム構成と基本動作
Fig. 1 System structure and its basic mechanism.

3. エージェント 移動経路の形式化

ここではモバイルエージェントの選択の基礎となる移動経路の記述言語と順序関係を定義する．

3.1 移動経路の記述言語

タスク依頼が要求する移動経路は下記の構文 $\mathcal{E}$ により，各モバイルエージェントの移動経路は $\mathcal{S}$ により記述されるとする．

定義 3.1 以下のような構文規則を持つ式の集合 $\mathcal{E}$ を次のように定義し，その要素を $E, E_1, E_2, \ldots$ と記す．

E	::=	$\mathtt{0}$	（移動終了）
	|	ℓ	（移動）
	|	$E_1 \,;\, E_2$	（逐次合成）
	|	$E_1 + E_2$	（選択合成）
	|	$E_1 \,\#\, E_2$	（非決定的選択合成）
	|	$E_1 \,\%\, E_2$	（非決定的可換合成）
	|	$E_1 \,\&\, E_2$	（非同期合成）
	|	E^*	（閉包）

ここで ℓ はホスト名の集合 $\mathcal{L}$ の要素である．なお，各演算子の結合力は強い方から ;，+，&，#，% の順となる．そして，上記のうち規則 $E_1 \,\#\, E_2$ と $E_1 \,\%\, E_2$，$E_1 \,\&\, E_2$，E^* を除いた式の集合を $\mathcal{S}$ とし，その要素を $S, S_1, S_2, \ldots$ と記す． □

構文 $\mathcal{E}$ および $\mathcal{S}$ の各演算子の意味は定義 3.3，3.2，そして定義 3.4 の順序関係をまたないといけないが，ここで基本的な記述式の非形式的な意味を述べておく．

- $\mathtt{0}$ は移動終了を示す．なお，適宜省力することがある．
- ℓ は名前 ℓ のホストへの移動を示す．なお，ここでは TCP/IP などが利用できる通信ネットワークを仮定し，任意のホストどうしが通信接続可能とするとともに，ℓ はホストの全域的ネットワークアドレスとする．
- $E_1 \,;\, E_2$ は経路 E_1 に従って移動した後に経路 E_2 に従って移動することを表す．
- $E_1 + E_2$ は経路 E_1 と E_2 のどちらか一方に従って移動するが，その選択は処理内容により明示的に決められる．
- $E_1 \,\#\, E_2$ は経路 E_1 と E_2 の非決定的選択を表す．これはタスク依頼の経路要求において E_1 と E_2 のどちらの経路をたどってもよいことを示す．
- $E_1 \,\%\, E_2$ は経路 E_1 と E_2 の可換合流性を表す．これはタスク依頼の経路要求において経路 E_1 と E_2 の順序は入れ換えて移動してもよいことを示す．
- $E_1 \,\&\, E_2$ は経路 E_1 と E_2 の移動が非同期に実行してよいことを示す☆．
- E^* はタスク要求においてエージェントが経路 E を 0 回以上の任意回繰り返して移動してもよいことを表す．

この言語はラベル付き遷移システムとして動作が定義される．なお，順序関係に議論を集中するため，自明な等価関係は構造合同性[11] によりあらかじめ定義しておく．

定義 3.2 $\equiv$ は構造的合同性を与える $\mathcal{E}$ 上の同値関係であり，下記のように定義される．

(1) $\mathtt{0} \,;\, E \equiv E$
(2) $E + \mathtt{0} \equiv E \quad E + E \equiv E \quad E_1 + E_2 \equiv E_2 + E_1 \quad E_1 + (E_2 + E_3) \equiv (E_1 + E_2) + E_3$
(3) $E \,\#\, E \equiv E \quad E_1 \,\#\, E_2 \equiv E_2 \,\#\, E_1 \quad E_1 \,\#\, (E_2 \,\#\, E_3) \equiv (E_1 \,\#\, E_2) \,\#\, E_3$
(4) $E \,\%\, \mathtt{0} \equiv E \quad E_1 \,\%\, E_2 \equiv E_2 \,\%\, E_2 \quad E_1 \,\%\, (E_2 \,\%\, E_3) \equiv (E_1 \,\%\, E_2) \,\%\, E_3$
(5) $E \,\&\, \mathtt{0} \equiv E \quad E_1 \,\&\, E_2 \equiv E_2 \,\&\, E_1 \quad E_1 \,\&\, (E_2 \,\&\, E_3) \equiv (E_1 \,\&\, E_2) \,\&\, E_3$
(6) $E^* \equiv \mathtt{0} \,\#\, (E \,;\, E^*)$

なお，以降では括弧を適宜省略することがある．また，$\equiv$ の 1 回以上の閉包も $\equiv$ と略記する． □

定義 3.3 $\mathcal{E}$ の遷移関係 $\stackrel{\cdot}{\longrightarrow} (\subseteq \mathcal{E} \times (\mathcal{L} \cup \{\tau\}) \times \mathcal{E})$ を下記の推論規則により定義する．

$$\frac{-}{\ell \xrightarrow{\ell} \mathtt{0}} \qquad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \,;\, E_2 \xrightarrow{\ell} E_1' \,;\, E_2} \qquad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 + E_2 \xrightarrow{\ell} E_1'}$$

$$\frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \,\&\, E_2 \xrightarrow{\ell} E_1' \,\&\, E_2} \qquad \frac{E_1 \equiv E_2 \quad E_1 \xrightarrow{\ell} E_1' \quad E_1' \equiv E_2'}{E_2 \xrightarrow{\ell} E_2'}$$

$$\frac{-}{E_1 \,\#\, E_2 \xrightarrow{\tau} E_1} \qquad \frac{-}{E_1 \,\%\, E_2 \xrightarrow{\tau} E_1 \,;\, E_2}$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \,;\, E_2 \xrightarrow{\tau} E_1' \,;\, E_2} \qquad \frac{E_1 \xrightarrow{\tau} E_1'}{E_1 + E_2 \xrightarrow{\tau} E_1'}$$

☆ 演算子 & はプロセス代数における並列合成に相当するが，定義 3.3 よりホスト移動は原始的事象としてインタリーブ（interleave）実行される．したがって，$E_1 \,\&\, E_2$ は経路 E_1 と E_2 が混在しながら逐次的に移動されることになる．

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \,\&\, E_2 \xrightarrow{\tau} E_1' \,\&\, E_2} \qquad \frac{E_1 \equiv E_2 \;\; E_1 \xrightarrow{\tau} E_1' \;\; E_1' \equiv E_2'}{E_2 \xrightarrow{\tau} E_2'}$$

なお $E_0 \xrightarrow{\tau} E_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} E_{n-1} \xrightarrow{\tau} E_n$ を $E_0(\xrightarrow{\tau})^n E_n$ と略記する．□

モバイルエージェントの代表的な移動経路パターン[1),19)]が記述できることを示す．ここでは以降の議論を簡単化するため $\mathcal{S}$ 上に変換するマクロ（*Travel*, *Star*, *Turn*）を導入する．また，移動ホスト名（$L \subseteq \mathcal{L}$）のリスト表現 $[\ell_1, \ell_2, \ldots, \ell_n]$（ただし，$\ell_1, \ldots, \ell_n \in L$）を導入する．このとき [] を空リスト，$car(X)$ はリスト X の先頭要素，つまり ℓ_1 となり，$cdr(X)$ はリスト X から先頭要素を除いたもの，つまり $[\ell_2, \ldots, \ell_n]$ とする．なお，これらは $\mathcal{S}$ 上に変換するものであり，記述言語自体を変更するものではない．

$$\begin{aligned}
Travel(X) &\stackrel{\text{def}}{=} car(X)\,;\, Travel(cdr(X))\\
Travel([]) &\stackrel{\text{def}}{=} \mathsf{0}\\
Star(X|h) &\stackrel{\text{def}}{=} (car(X)\,;\, h)\,;\, Star(cdr(X)|h)\\
Star([]|h) &\stackrel{\text{def}}{=} \mathsf{0}\\
Turn(X) &\stackrel{\text{def}}{=} car(X)\,;\, Turn(cdr(X))\,;\, car(X)\\
Turn([]) &\stackrel{\text{def}}{=} \mathsf{0}
\end{aligned}$$

ここで，$h \in \mathcal{L}$，X を $\mathcal{L}$ 上の要素からなるリストとする．たとえば $Travel(X)$ は複数ホストのリスト X を先頭から順に移動する経路に相当する．そして，$Star(X|h)$ は複数ホストのリスト X を順に移動することになるが，各ホストに移動するたびに所定ホスト h に戻る，つまりスター形状に移動することになる．ここで $X = [a, b, c]$ とするときの状態遷移を途中まで示す．

$$\begin{aligned}
Star([a,b,c]|h) &\stackrel{\text{def}}{=} (a\,;\, h)\,;\, Star([b,c]|h)\\
&\xrightarrow{a} h\,;\, Star([b,c]|h)\\
&\xrightarrow{h} Star([b,c]|h)\\
&\stackrel{\text{def}}{=} (b\,;\, h)\,;\, Star([c]|h)\\
&\xrightarrow{b} h\,;\, Star([c]|h)\\
&\xrightarrow{h} Star([c]|h)
\end{aligned}$$

そして *Turn* は複数ホストのリスト X を順に先頭から順に移動してすべてのホストに移動すると逆順に戻る経路を表す．次にタスク依頼の経路要求の記述の例を示す．

$$\begin{aligned}
Tour(X|h) &\stackrel{\text{def}}{=} Tour'(X)\,;\, h\\
Tour'(X) &\stackrel{\text{def}}{=} car(X)\,\%\, Tour'(cdr(X))\\
Tour'([]) &\stackrel{\text{def}}{=} \mathsf{0}\\
Hub(X|h) &\stackrel{\text{def}}{=} (car(X)\,;\, h)\,\%\, Hub(cdr(X))\\
Hub([]|h) &\stackrel{\text{def}}{=} \mathsf{0}
\end{aligned}$$

ここで $Tour(X|h)$ は $X = [a, b, c]$ とするとき下記のように展開される．

$$\begin{aligned}
Tour([a,b,c]|h) &\stackrel{\text{def}}{=} Tour'([a,b,c])\,;\, h\\
&\stackrel{\text{def}}{=} (a\,\%\, Tour'([b,c]))\,;\, h\\
&\stackrel{\text{def}}{=} \cdots\\
&\equiv (a\,\%\, b\,\%\, c)\,;\, h
\end{aligned}$$

したがって，$Tour([a, b, c]|h)$ はホスト a，b，c を任意の順番で一巡するとホスト h に戻ることを示す．一方，$Hub(X|h)$ はホスト h に戻りながらリスト X 上のホストを任意の順番で移動する経路要求を表す．

3.2 移動経路による順序関係

次にエージェントの移動経路 $\mathcal{S}$ がタスク要求における移動経路 $\mathcal{E}$ を満足するかの判定方法を双模倣性[10)]をもとに順序関係として定式化していく．なお，その順序関係ではエージェントの移動回数も調べられるようにする．

定義 3.4 ある自然数 n $(n \geq 0)$ に対して次の 3 つの条件を満足する関係 $\mathcal{R}^n$ $(R^n \subseteq \mathcal{E} \times \mathcal{S})$ を考える．

(1) $\forall \ell \forall E'$: $E \xrightarrow{\ell} E'$ ならば
$\exists S'$: $S \xrightarrow{\ell} S'$ かつ $(E', S') \in \mathcal{R}^{n-1}$

(2) $\exists E'$: $E\,(\xrightarrow{\tau})^*\,E'$ かつ $(E', S) \in \mathcal{R}^n$

(3) $\forall \ell \forall S'$: $S \xrightarrow{\ell} S'$ ならば $\exists E'$: $E\,(\xrightarrow{\tau})^*\,\xrightarrow{\ell}$ E かつ $(E', S') \in \mathcal{R}^{n-1}$

ここで，$n = 0$ のとき $n - 1$ は 0 とする．そして $(E, S) \in \mathcal{R}^n$（ただし $n \geq 0$）の最大集合となるものを充足順序関係と呼び，E と S 上の充足順序関係を $E \sqsupseteq_n S$ と記述する．□

充足順序関係（$E \sqsupseteq_n S$）の非形式的意味を述べておく．エージェントの経路 S はタスク要求の経路 E を充足する必要があるが，**(1)** は E が要求するホストの移動順序を S が満足し，さらに E が処理結果により移動先を選択できる場合は同様の選択を S も持ち，その各選択先でも S は E を同様に充足することを示す．**(2)** は E における非決定性演算子（# や %，E^*）による選択候補のうち少なくとも 1 つを S が充足すれば十分であることを示す．**(3)** は S の移動経路であれば E の移動経路となること，つまり S が E により要求されている経路以外には移動しないことを示している．なお，$\sqsupseteq_n$ の n は E の充足に必要なホスト移動回数を表す．

ところで，充足順序関係 $\sqsupseteq_n$ はその定義域を $\mathcal{S} \times \mathcal{S}$ とするとき前順序関係となる．このほかにも次のような性質を持つ．

性質 3.5 任意の $E \in \mathcal{E}$，$S \in \mathcal{S}$，$n_1 \leq n_2$ とするとき，$E \sqsupseteq_{n_1} S$ ならば $E \sqsupseteq_{n_2} S$ □

性質 3.6 任意の $E_1, E_2 \in \mathcal{E}$，$S_1, S_2 \in \mathcal{S}$

とするとき，$E_1 \sqsupseteq_{n_1} S_1$ かつ $E_2 \sqsupseteq_{n_2} S_2$ ならば，$E_1 ; E_2 \sqsupseteq_{n_1+n_2} S_1 ; S_2$ と $E_1 + E_2 \sqsupseteq_{max(n_1,n_2)} S_1 + S_2$ となる． □

この性質により経路要求を充足する移動経路は，他の充足可能な移動経路と合成しても充足されることが保証される．次にいくつかの比較例を示す．

- ホスト選択を含む移動経路
  $$(a \,\%\, b) ; (c + d) \quad \sqsupseteq_3 \quad a ; b ; (c + d)$$
  両辺の部分式は $a \,\%\, b \sqsupseteq_2 a ; b$ となる．そして左辺のタスク依頼が経路選択 $c + d$ を要求する場合，これを充足するエージェントは処理結果より経路選択を行える必要がある．
- ホスト到着順序が任意となる移動経路
  $$(a \,\%\, b \,\%\, c) ; h \quad \sqsupseteq_4 \quad c ; a ; b ; h$$
  ここで左辺はホスト a, b, c の任意の順番で一巡して，ホスト h に戻ることを要求している．右辺を $a ; b ; c ; h$ にした場合も充足順序関係となるが，$a ; b ; h$ や $a ; h ; b ; h ; c ; h$ の場合はならない．
- 経由ホストを許す移動経路
  $$((a ; b ; c) \,\&\, h^{*}) ; h \quad \sqsupseteq_6 \quad Star([a, b, c]|h)$$
  ここで $(a ; b ; c) \,\&\, h^{*}$ は移動経路 $a ; b ; c$ にホスト h への移動が 0 回以上が挿入されることに相当する．したがって，左辺が要求する移動経路はホスト a から b, c, h に移動する必要があるが，途中でホスト h に 0 回以上立ち寄ってもよいことを示している．右辺を $Travel([a, b, c]) ; h$ とした場合も充足順序関係となるが，$Star([b, a, c]|h)$ のように移動順が異なる場合は成立しない．

モバイルエージェントがタスク遂行に要するコストはネットワーク帯域やホストの計算能力，エージェントのサイズの影響も受けるが，本論文ではホスト間移動回数だけに着目した．これはモバイルエージェント性能評価に関する研究報告[6),7)] などで示されているように，一般のオフィス向け LAN 環境ではエージェント移動時のデータ転送遅延より，移動元および移動先ホストにおけるエージェントのデータ変換および逆変換を含む移動前後処理のコストが大きいからである．なお，著者は通信遅延やホスト位置などの表現性を持つプロセス代数系[13)] を定式化しており，これの成果を利用することにより，ネットワークやホスト性能を含む詳細なコスト比較が可能になる．ただし，こうした表現性の拡大とエージェント充足判定のコストはトレードオフとなる．

4. 設計・実装

モバイルエージェントの選択は，ネットワーク上の

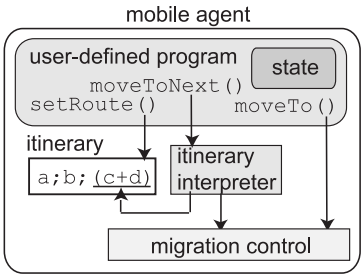

図 2 モバイルエージェントの構成
Fig. 2 The structure of a mobile agent.

1 つ以上のホストに用意される**エージェントプール**と呼ぶ機構により代行し，モバイルエージェント側の処理とプログラムサイズを最小化する．各エージェントプールはモバイルエージェントの待機場所となるとともに，外部環境からタスク依頼を受け取り，そのタスクの要求するエージェント移動経路に基づいて待機中のエージェントからそのタスク遂行に最適なエージェントを選択する．ただし，タスク依頼は多様性が高く，またアプリケーションの種類に依存する．また，本論文の目的は移動経路によるモバイルエージェントの選択手法を提示することにある．そこで，本論文ではタスク依頼中の移動経路要求が構文 $\mathcal{E}$ で記述されていること以外はタスク依頼の形式は限定しないとする☆．なお，プロトタイプシステムは Java 言語を利用したモバイルエージェントシステムである MobileSpaces[14)] 上に実装されており，モバイルエージェントの実行とホスト移動は MobileSpaces システムにより実現される．

4.1 モバイルエージェント

モバイルエージェントは下記の Java 言語によるクラス `ItineraryAgent` の派生クラスとして定義される．

```
public class ItineraryAgent
  extends MobileAgent {
  // 移動経路 (記述式) の登録
  void setRoute(Route r) throws
    IllegalSyntaxException .. { ... }
  // ホスト名を明示的に指定して移動
  void moveTo(Host h) throws
    IllegalHostException,
      NoSuchHostException  ... { ... }
  // 登録した移動経路上の次の移動先ホストに移動
  void moveToNext() throws
    MultiplePossibleHostsException,
      NoSuchHostException  ... { ... }
  // 移動経路上で次の移動先となりえるホスト名を取得
  Host[] getPossibleHosts() ... { ... }
  ...
}
```

ここでモバイルエージェントの基本構造を**図 2** に示

☆ 具体的なタスク依頼の例を 5 章に示す．

す．モバイルエージェントは構文 $\mathcal{S}$ により記述したそれ自身の移動経路をメソッド `setRoute()` の引数として下記のように呼び出すことにより，それ自身と後述のエージェントプールに登録する．

```
setRoute(new Route("a;b;(c+d)"));
```

ここで経路 `a;b;(c+d)` はホスト `a` に移動した後にホスト `b` に移動し，次に処理内容に応じてホスト `c` または `d` のどちらかに移動することを表す．そして，このエージェントは下記の 2 つの方法によりホスト間移動を行うことができる．

- 1 つ目の方法は登録した移動経路に沿って移動するものである．各エージェントは $\mathcal{S}$ 上の言語により記述された移動経路を解釈・実行する簡易インタプリタを持ち，メソッド `moveToNext()` を呼び出すごとに移動経路を逐次実行してエージェント自身を移動させる．ただし，移動経路に選択合成（`+`）による経路分岐が含まれる場合は，移動可能な経路が複数存在することとなり，例外 `MultiplePossibleHostsException` が返される．そこで，エージェントはメソッド `getPossibleHosts()` を通じて，仕様に示された移動可能なホスト名の集合を取得する．そして，エージェントの処理内容に応じてそのうちの 1 つのホストを選び，メソッド `moveTo()` をそのホストの名前とともに呼び出して移動する．たとえば上記のように移動経路 `a;b;(c+d)` を登録したモバイルエージェントの場合，**図 3** のようにメソッド `moveToNext()` を 2 回呼び出すとホスト `a` から `b` に移動するが，3 回目の移動はメソッド `moveTo()` を用い，その引数としてホスト名 `c` と `d` のどちらかを与えたときだけ移動できる．
- もう 1 つの方法は移動先ホストを明示的に指定して移動するものである．ここではクラス `ItineraryAgent` のメソッド `moveTo()` を移動先ホスト名とともに呼び出す．ただし，メソッド `setRoute()` で登録した移動経路に反する場合は例外 `IllegalHostException` が返される．たとえば移動経路 `a;b;(c+d)` を登録したモバイルエージェントは，ホスト `a` の次に `b`，そして `c` または `d` を引数として同メソッドを順次呼び出していくことにより移動できるが，それ以外の経路は強制されて移動できない．

ところで，現在の実装ではエージェントプールはそのエージェントプール内で待機しているエージェントを選択候補とし，移動中のエージェントは候補としない☆．これは移動中エージェントは何らかの処理を行うために移動していることが多く，別タスクを割り当てられるとは限らないからである．さらに移動中エージェントの位置捕捉とメッセージの転送に要する実現コストは小さくない．そこで，各モバイルエージェントはタスクを終了すると所定のエージェントプールに帰還して，次のタスク割当てを待つ．そして，移動経路の登録は各エージェントがエージェントプールに戻るたびに自動的に行われる．なお，エージェントはメソッド `setRoute()` により移動中に移動経路を再設定することが許されるが，エージェントプールへの帰還後に登録する移動経路に反映され，現在移動中の経路の変更はできない．

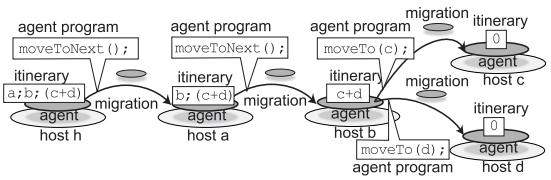

図 3 移動経路 `a;b;(c+d)` を持つモバイルエージェントの移動
Fig. 3 The movement of a mobile agent whose itinerary is specified as `a;b;(c+d)`.

4.2 エージェントプール

エージェントプールは MobileSpaces の階層化機構を利用し，モバイルエージェントを内包するエージェントとして実現される☆☆．そして，ネットワーク上の 1 つ以上のホストに配置されて，待機エージェントの格納用プレースとなる（**図 4**）．

外部から与えられるタスク依頼はエージェントプールが受け取る．このとき，前述のように移動経路に関する要求は構文 $\mathcal{E}$ に従って記述されるとし，エージェントプールで待機しているモバイルエージェントはその移動経路を $\mathcal{S}$ により記述したものを保持していることから，定義 3.4 の充足順序関係を利用して，タスク依頼の移動経路要求を満足し，さらに移動回数が最小となるモバイルエージェントを探す．

具体的には比較対象となる 2 つの記述式に対して定義 3.3 の状態遷移先を節点，ラベル付き遷移を名前付き枝とする木構造を生成する．そして，定義 3.4 に従って，その 2 つの木構造において互いに対応する節点どうしが同じ名前を持つ枝を持つことをすべての節点に関して調べることにより実現している．そしてその充足性の判定は 2 つの木構造のそれぞれの根を起点に部分木を持たない節点に至るまで次の動作を実行す

☆ したがってホストを巡回し続けるエージェントは本論文の対象外となる．
☆☆ ただし，エージェントプール自体は移動性を持たないとする．

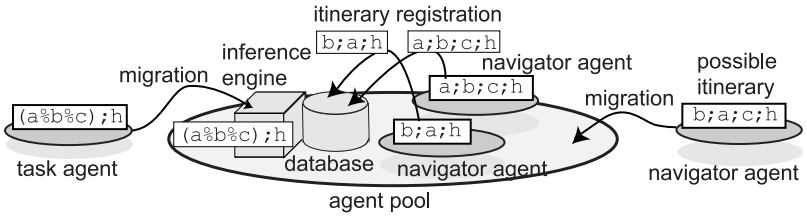

図 4 エージェントプール
Fig. 4 Agent pool.

ることにより行う.

(1) まず比較対象の節点が互いに同じ名前の枝を持つか調べる.
(2) 同等の枝を持つ場合は枝の先にある節点について (1) を繰り返す.
(3) 逆に持たない場合は 1 つ前の節点に後戻りして, 残りの節点に関して (1) を繰り返す.

そして, タスク依頼の移動経路要求と充足順序関係を満足するエージェントが複数存在する場合は, 定義 3.4 の $\sqsupseteq_n$ の n が最小のもの, つまり移動回数が最小なものを選ぶ☆. なお, 判定コストを低減するため, エージェントプールはエージェントから移動経路を示す記述式を受け取った時点で木構造を生成する. したがってエージェント選択時は木構造どうしの比較だけが行われることになる. また, 移動経路要求に含まれる閉包 E^* は定義 3.3 の合同等価性による展開を遅延することにより, 比較対象の木構造は有限として扱える.

補　　足

本論文で示した手法は各エージェント内の補助プログラムとエージェントプールとして実装されている. また, エージェントプール以外のホストには特別な機構は不要であり, さらにエージェントプールに関してもいくつかの既存モバイルエージェントシステムが提供するプレース[20] またはステーショナリエージェントとして実現できると予想される. したがって, 広範なモバイルエージェントシステム上で利用可能であることに注意されたい.

5. 応 用 例

ここでは具体的なタスク割当てに本論文で提案したモバイルエージェント選択手法が利用できることを示す. 典型的なモバイルエージェントの応用事例であるネットワーク管理や遠隔情報検索を行うモバイルエージェントは所定のホストに到着すると, それ自身のアプリケーション依存処理プログラムを呼び出し, 処理が終了すると次のホストに移動する. このとき各ホストで実行される処理プログラムはホストによらず 1 種類となることが多い. たとえばネットワーク管理をするエージェントは監視・制御対象となるホストに移動するたびに同じ監視・制御用プログラムを実行することがある. そこで, ここでは各ホストで実行される処理プログラムが 1 種類となる場合を扱う. そしてモバイルエージェントを各ホストで実行される処理に相当するプログラム (タスクエージェントと呼ぶ) を運ぶ実体 (移送エージェントと呼ぶ) の 2 つから構成する.

- **タスクエージェント** は検索やモニタリングなどのアプリケーション処理に相当するプログラムである.
- **移送エージェント** はそれ自身の移動経路に従ってタスクエージェントを所定のホストに運ぶモバイルエージェントであり, 各サブネットワークや移動経路ごとに用意される.

タスクエージェントと移送エージェントとともに前述の MobileSpaces システムのモバイルエージェントとして実装した☆☆. MobileSpaces はその階層化機構により, モバイルエージェントが他のエージェントを内包することができ, さらにエージェントが他のホストに移動する際, それが内包するエージェントも一緒に移動することになる. ここでは移送エージェントは 1 つ以上のタスクエージェントを内包するモバイルエージェントとして実装される. 一方, タスクエージェントはそれ自身の移動経路を持たない受動的なモバイルエージェントとなる. このタスクエージェントは下記のようなクラス `TaskAgent` のサブクラスとして定義される.

```
public class TaskAgent extends ItineraryAgent {
  // 移動経路要求の登録
  void setRoute(Route r)
    throws IllegalSyntaxException ... { ... }
  // 現ホスト上の処理終了を移送エージェントに通知
```

☆ 現在の実装における充足判定アルゴリズムは必ずしも最適化されたものではないが, 本論文に示された記述例はいずれも 10 ms 以下で判定可能である.

☆☆ 本章のタスク割当て機構は MobileSpaces システムに依存するが, 本論文で示したエージェント選択手法とエージェントプール機構自体は依存しない.

```
  void done() throws
    MultiplePossibleHostsException,
      TerminalRouteException ... { ... }
  ...
  // 移動直後に呼び出される抽象メソッドであり,
  // これに各ホスト上の処理を実装する
  void arrived(Host here);
  ...
}
```

たとえば，データベース検索処理に相当するタスクエージェントの場合，抽象メソッド `arrived()` にホスト到着した際の処理としてデータベース検索プログラムを定義する．また，タスクが移送エージェントに要求する移動経路はメソッド `setRoute()` を呼び出すことにより指定する．なお，`h^*`は構文 $\mathcal{E}$ の h^* に相当する．

```
setRoute(new Route("h;((a%b%c%d)&h^*);h"));
```

実際には移動経路ごとに様々な移送エージェントをエージェントプールに待機させることになるが，ここではその中でも次の 2 つの移送エージェントを考える．1 つ目の移送エージェント `CarrierAgent1` である．

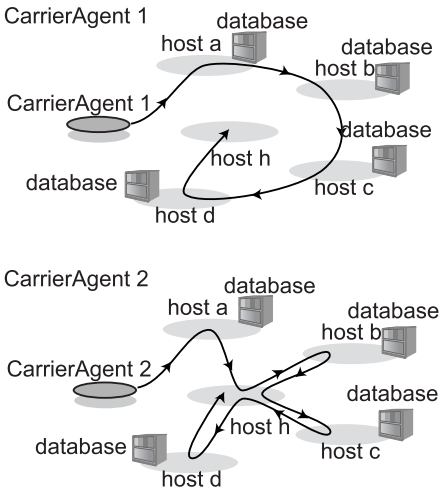

図 5 移送エージェント（CarrierAgent1 と CarrierAgent2）の移動経路
Fig. 5 The itineraries of two carrier agents (CarrierAgent1 and CarrierAgent2).

```
public class CarrierAgent1
  extends MobileAgent {
  public CarriarAgent1() {
    // 移動経路を設定
    setRoute(new Route("h;a;b;c;d;h"));
  }
  // タスクから呼び出される終了通知メソッド
  public void done() throws
    MultiplePossibleHostsException .. {
    moveToNext();
  }
  ...
}
```

もう 1 つの移送エージェント `CarrierAgent2` である．

```
public class CarrierAgent2
  extends MobileAgent {
  public CarriarAgent1() {
    setRoute(new Route("h;a;h;b;h;c;h;d;h"));
  }
  // タスクから呼び出される終了通知メソッド
  public void done() throws
    MultiplePossibleHostsException .. {
    moveToNext();
  }
  ...
}
```

上記の移送エージェントが持つ移動経路を**図 5** に示す．また，**図 6** のようにエージェントプールはタスクエージェントが要求する移動経路 `h;((a%b%c%d)&h^*);h` を満足する移送エージェントを 4 章に示した充足順序関係に対応する判定手法を利用して調べる．待機中の移送エージェント `CarrierAgent1` と `CarrierAgent2` の両方が要求経路を充足するが，ここでは移動回数が少ない `CarrierAgent1` の方が選ばれる．そしてエージェントプールはタスクエージェントを移送エージェント `CarrierAgent1` に移動させる．一方，移送エージェントはタスクエージェントを受け取るとそれ自身の移動経路に従って移動を開始する．そして，移送エージェントはホストに到着するたびに，タスク側エージェントのメソッド `arrived()` を呼び出す．そこで，タスクエージェントは各ホストで実行すべき処理プログラムを実行し，それが終わるとメソッド `done()` を呼び出して，移送エージェントに次のホストへの移動を

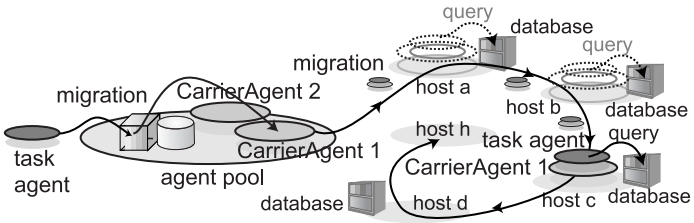

図 6 移送エージェント（CarrierAgent1）の選択
Fig. 6 Selection of a carrier agnet (CarrierAgent1).

促す．なお，このようにタスク処理とエージェント移動を分離することにより，前者はネットワーク構造に非依存となり，後者はアプリケーションに非依存となり，両者を適宜組み合わせることによりモバイルエージェントの再利用性を向上させることができる．

6. 関連研究

ソフトウェアエージェント研究では，契約ネットプロトコル[18]やKQML[4]をはじめとしてエージェントへのタスク割当て手法が数多く研究されているが，モバイルエージェントを対象としたものはきわめて少ない．その中でも，Plangent[12]はモバイルエージェントにプランニング機能を提供することから，モバイルエージェントはユーザ要求を満足する移動経路や処理内容に関するプランを動的に生成し，そのプランに従って移動・処理を行うことができる．ただし，Plangentのプラニング機構そのものは人工知能研究におけるプラニング手法を踏襲したものであり，エージェント移動経路に特化したものではなく，また複雑かつ効率的な経路は扱うのは困難と思われる．一方，本論文は事前に与えられた移動経路の比較・選択手法を明らかにするものである．このほか，モバイルエージェントの移動パターンに関する研究事例がいくつかあるが[1),19)]，モバイルエージェントのソフトウェア開発方法論を提案するものであり，移動経路によるエージェント選択は対象としていない．

モバイルエージェントの形式化としてはMobile UNITY[9]やMobile Ambient[3]，Join Calculus[5]などがある．このうちMobile UNITYは分散処理の仕様記述手法であるUNITYに移動に関する表現性を付加したものであり，移動先における処理の記述などには有用であるが，移動経路自体の抽出や形式化は困難である．Mobile AmbientとJoin Calculusは本論文の言語と同様にプロセス代数（またはプロセス計算）をもとにした体系であるが，エージェント間通信などの相互作用を含めたモバイルエージェント計算全体を形式化することを目的としている．特にMobile Ambientが扱えるモバイルエージェントは入れ子状に構造化された移動経路を持つものに限定されてしまう．一方，本論文の言語はモバイルエージェントの移動経路に特化したものであり，移動先ホストをその移動順序に並べた列として直接的に記述することができる．さらに処理内容に依存した移動経路や，移動順序・回数の任意性に関する表現性も有している．なお，移動経路によるモバイルエージェントの順序関係を定式化した研究事例は著者の知る限りない．

ところで，著者は本論文の5章と同様に処理内容と経路制御部分を分離し，それぞれをモバイルエージェントとして実現する方法を提案している[15),16)]．ただし，これはネットワーク管理システム向けモバイルエージェントの再利用性を向上させるものであり，またモバイルエージェントの選択・比較手法自体はいっさい扱っていない．このほかにもモバイルエージェント向けの経路制御プロトコルを設計・実装を行っている[17)]．これはアクティブネットワーク技術を利用して，エージェント転送プロトコルをネットワーク環境やアプリケーション要求の変化に応じて動的に変更することが目的であり，移動経路に特化したものではない．

7. まとめ

本論文ではモバイルエージェントへのタスク割当てを念頭に，移動経路によるモバイルエージェントの選択手法を提案した．これはエージェントの移動経路に関する仕様記述言語とその記述式間の順序関係から定式化される．これらによりタスク依頼が要求する移動経路とエージェントの移動可能経路が一致する場合だけでなく，要求されたホストに移動可能であれば到達順序・回数が任意となる場合も柔軟に判定することができる．そして，この手法のプロトタイプシステムをJava言語によるモバイルエージェント上に実装した．このシステムはモバイルエージェントの待機場所を提供するとともに，外部環境からタスク依頼を受け取ると，この順序関係を利用して待機中エージェントからタスク依頼の移動経路要求を満足するエージェントを選択することができる．

最後に今後の課題を述べる．本論文で提案された言語は移動経路の記述に限定しているが，今後は具体的な処理内容に関する表現性を拡張することを検討している．また，現在の実装における充足順序関係の判定アルゴリズムは必ずしも最適化されたものでない．その効率化をはかるとともに充足順序関係に対応する公理系の導入を考えている．ところで，この言語は移動経路に関するポリシー記述が当初の研究目標であった．今後はルール記述性などを拡張して，モバイルエージェント向けのポリシー記述言語・推論系として発展させる予定である．

謝辞 有意義なコメントをいただいた査読者の方々に感謝する．

参考文献

1) Aridor, Y. and Lange, D.B.: Agent Design Patterns: Elements of Agent Application De-

sign, *Proc. 2nd International Conference on Autonomous Agents* (*Agents '98*), pp.108–115, ACM Press (1998).
2) Beaten, J.C.M. and Bergstra, J.A.: *Process Algebra*, Cambridge University Press (1990).
3) Cardelli, L. and Gordon, A.D.: Mobile Ambients, *Foundations of Software Science and Computational Structures*, LNCS, Vol.1378, pp.140–155 (1998).
4) Finin, T., Labrou, Y. and Mayfield, J.: KQML as An Agent Communication Language, *Software Agents*, MIT Press (1997).
5) Fournet, C., Gonthier, G., Levy, J., Marnaget, L. and Remy, D.: A Calculus of Mobile Agents, *Proc. CONCUR'96*, LNCS, Vol.1119, pp.406–421, Springer (1996).
6) Gray, R.S., et al.: Mobile-Agent versus Client/Server Performance: Scalability in an Information-Retrieval Task, *Proc. International Conference Mobile Agents* (*MA'2001*), LNCS, Vol.2240, pp.198-212, Springer (Dec. 2001).
7) Hagimont, D. and Ismail, L.: A Performance Evaluation of the Mobile Agent Paradigm, *Proc. International Conference on Object-oriented Programming, Systems, Languages, and Applications* (*OOPSLA'99*), pp.306–313, ACM Press (Oct. 1999).
8) Karmouch, A.: Mobile Software Agents for Telecommunications, *IEEE Communication Magazine*, Vol.36, No.7 (1998).
9) McCann, P.J. and Roman, G.-C.: Compositional Programming Abstractions for Mobile Computing, *IEEE Trans. Softw. Eng.*, Vol.24, No.2 (1998).
10) Milner, R.: *Communication and Concurrency*, Prentice Hall (1989).
11) Milner, R.: *Communicating and mobile systems: the π-calculus*, Cambridge University Press (1999).
12) Ohsuga, A., Nagai, Y., Irie, Y., Hattori, M. and Honiden, S.: PLANGENT: An Approach to Making Mobile Agents Intelligent, *IEEE Internet Computing*, Vol.1, No.4, pp.55–57 (1997).
13) Satoh, I. and Tokoro, M.: Time and Asynchrony in Interactions among Distributed Real-Time Objects, *Proc. European Conference on Object-Oriented Programming* (*ECOOP'95*), LNCS, Vol.952, pp.331–350, Springer (July 1995).
14) Satoh, I.: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, *Proc. International Conference on Distributed Computing Systems* (*ICDCS'2000*), pp.161–168, IEEE Computer Society (Apr. 2000).
15) Satoh, I.: A Framework for Building Reusable Mobile Agents for Network Management, *Proc. Network Operations and Managements Symposium* (*NOMS'2002*), pp.51–64, IEEE Communication Society (Apr. 2002).
16) Satoh, I.: Reusable Mobile Agents for Managing Networks, *Journal of Information Processing Society of Japan*, Vol.43, No.11, pp.3448–3457 (2002).
17) Satoh, I.: Network Protocols for Mobile Agents by Mobile Agents, *Journal of Information Processing Society of Japan*, Vol.44, No.3, pp.760–770 (2002).
18) Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, *IEEE Trans. Computers*, pp.1104–1113 (1980).
19) Tahara, Y., Ohsuga, A. and Honiden, S.: Agent System Development Method Based on Agent Patterns, *Proc. International Conference on Software Engineering* (*ICSE'99*), pp.356–367, IEEE Computer Society (1999).
20) White, J.E.: *Telescript Technology: Mobile Agents*, General Magic (1995).

(平成 14 年 9 月 30 日受付)
(平成 15 年 2 月 4 日採録)

佐藤 一郎（正会員）

1991 年慶應義塾大学理工学部電気工学科卒業．1996 年同大学院理工学研究科計算機科学専攻後期博士課程修了，博士（工学）．1996 年お茶の水女子大学理学部情報科学科助手，1998 年同学科助教授．2001 年より国立情報学研究所助教授．このほか，1994 年〜1995 年 Rank Xerox Grenoble 研究所客員研究員．1999 年〜2001 年科学技術振興事業団さきがけ研究 21（「情報と知」領域）研究員．1996 年度情報処理学会論文賞，1999 年度同学会山下奨励賞，1998 年日本ソフトウェア科学会高橋奨励賞ほか受賞．分散システムの理論，プログラミング言語，ミドルウェア等の研究に従事．日本ソフトウェア科学会，電子情報通信学会，IEEE，ACM 各会員．