Flageアーキテクチャのカーネル言語†

4N－5

粂野 文洋[1]　○佐藤 仁孝[2]　田原 康之[3]　大須賀 昭彦[4]　本位田 真一[4]

情報処理振興事業協会(IPA)
新ソフトウェア構造化モデル研究本部

1　はじめに

本稿では、環境の変化に柔軟に適応するソフトウェアを構築するために提案したFlageアーキテクチャのカーネル言語を提示する。Flageカーネル言語は並行オブジェクトモデル[1]をベースとした言語であり、エージェントと場を記述する機能を提供している。以降でエージェントおよび場の定義方法を概説し、場に対するエージェント動作を述べる。

2　エージェント

エージェントは、データを保持する属性とそれに対して操作を行なうメソッドで定義されるオブジェクトである。エージェントは互いにメッセージ通信を行ないながら並行動作するが、エージェント内は逐次的に動作する。メッセージの送信方法には、相手先の識別子を指定した送信(一対一通信)と、場への送信が存在する。場へのメッセージ送信が行なわれた場合、場に属するエージェントにそのメッセージがマルチキャストされる。一対一通信では同期および非同期型の通信を用意し、マルチキャストには非同期型の通信を用意している。エージェントは動的に生成できる。エージェントは自分自身、またはその親、祖先によってのみ消滅させることが可能である。また、エージェントは場の変化や要求の変化に動的に適応するために、自らの定義をデータとして扱い、動的に変更を行なう。この機能を実現するため、エージェントはメタレベルとベースレベルの二層構造(メタ構造)を持つ。各レベルでメソッドと属性が定義される。メッセージも２種類存在し、１つはメタレベルで、もう１つはベースレベルで処理される。エージェントは具体的には図1の形式の項で定義される。メソッド定義(m::m1;..;mn)に対し、mにマッチするメッセージを受け取ると、m1からmnが順次処理される。各miは、属性への値の代入、メッセージ送信、ローカルメソッドの呼び出しなどである。specでは副作用のない関数を定義する。現在の処理系ではprolog節で定義を行なう。

メタレベルのメソッドは、ベースレベルの定義や状態(属性値)をデータとして扱い、変更することができる。これに対して、ベースレベルのメソッドはベースレベルの状態

```
agent(aid, % aidはエージェントの識別子
 meta{   % attrは属性名と初期値の対
   attribute   attr, ..., attr
         % メソッドの定義
     method   method1 ::   method11;
                           ...
                           method1n,
              method2 ::   method21;
                           ... ,
         % 関数の定義
       spec   spec, ..., spec },
 base{ ... } )
```

図 1: エージェントの定義

のみを操作することができる。

メタレベルには、ベースレベルのメッセージをベースレベルの定義で解釈実行するインタプリタとそれに関連する手続きを定義する。具体的には、ベースレベルのインタプリタの、所定のプロトコル(メタプロトコル)を利用したカスタマイズ、および、カスタマイズしたそのインタプリタから呼び出される手続き(たとえば、ベースレベルの定義変更手続きなど)が定義される。

メタプロトコルはインタプリタ実行中におけるいくつかの特定の時点で行う処理を再定義するための言語機能である。たとえば、受け取ったメッセージが未知のものであることが分かった場合の処理の定義などをメタプロトコルを利用して再定義する。メタレベル特有の言語プリミティブとしては、ベースレベル定義の参照/変更やベースレベルのメソッドのエージェント内での模擬実行などの機能が用意される。場の出入りに関する機能もメタレベル定義で用いる機能である。また、ベースレベルの定義の変更履歴を最新の定義も含め、保持/管理することができる。

3　場

エージェントは自らの定義にしたがって稼働する実行主体であるのに対し、場はある環境下でエージェントが実行する動作定義と制約を表現した非実行主体である。これらの定義は、その環境下に存在する、すなわち、その場の中にいるエージェント共通のものであるから、個々のエージェント定義とは切り離して記述するのが自然である。場で規定する定義項目として、エージェントの動的な場への出入りも考慮し、以下のものを与える。場は動的に生成・消滅させることが可能である。

属性に対する制約　その場においてエージェントがベースレベルに持っている属性に課される制約である。制約は属

---

†Flage Kernel Language
Fumihiro Kumeno, Hirotaka Sato, Yasuyuki Tahara, Akihiko Ohsuga, Shinichi Honiden

[1](株)三菱総合研究所より出向中
[2](株)アイネスより出向中
[3]東芝より出向中
[4]現 東芝

性に関する条件式、たとえば、$A = B$や$A \leq B(A, B$は属性が引数になる関数の式)、といったブール値関数呼び出しを論理結合子で結んだものである。属性の制約には、場に入るための制約、場を出るための制約、場で常に満たしている制約の3種類が存在する。

場固有のメソッド　場固有のメソッドはエージェントのベースレベルメソッドの定義を場固有のものに固定する。その場でエージェントのベースレベルに付加するメソッドの他に、そのメソッドが新たに呼び出す関数と参照・書き込みを行う属性の定義を記述できる。また、各メソッドの定義に対し、場を出た後もその場固有の定義をエージェントが保持するかしないかを場で指定することができる。メソッドを保持すると指定した場合、そのメソッドが参照している属性や呼び出している関数も同時に保持される。ただし、場固有の属性(後述)にアクセスしているメソッドは場外で保持することはできない。また、関数や属性に対しても保持する/しないの指定ができる。

エージェントがmethodAというメソッドを固有に持っていたとしても、エージェントが入った場で、methodAの場固有の定義が規定されている場合、その定義がエージェントに取り込まれ、利用される。ただし、元の定義も保存されており、場を出る時に保持されないメソッドとしてmethodAが指定されている場合、場を出た時点でmethodAの元の定義が復活する。場を出る時に保持されるメソッドとしてmethodAが指定された場合には、場固有のメソッド定義がエージェントの新たな定義となる。ただし、元の定義は、実行時に参照されることはないが、そのまま保持される。

場固有の属性　場におけるエージェントの共有知識を保持するための属性定義である。各属性に対するアクセス手続きも指定できる。指定がない場合は組み込みのアクセス手続きが呼ばれる。いくつかの組み込みの属性が存在し、場に出入りするエージェントの情報、場の定義自身、その場に関連した場の名前のリスト、場の制約に適応するメソッドの格納などに利用される。

また、マシン・ノードを表現した場(ノード)を組み込みの場として導入している。エージェントは、必ずいずれかのノードに属しており、ノードを除くすべての場は1つのノード内に含まれていなければならない。エージェントおよび場で構成されるアプリケーションの概念図を図2に示す。

4　場に対するエージェントの振舞い

エージェントが場に対して行なう手続きは、エージェントのメタレベルが組み込み機能として行なう基本手続きである。エージェントは、場に入るためのプリミティブを起動することにより、場の属性から、場の制約と場固有のメソッドのインタフェース(メソッド名、引数、各引数のデータ構造)などの情報を参照できる。エージェントが場に入る際には、自分の状態(属性値)が場の制約を満たすかどうかのチェック(条件チェック)をエージェント自身で実行する。その結果、もし制約を満たしていない場合、エージェントは制約を満たすよう自らを変更する適応を(必要に応じて対話的に)行なう。適応方法を実現するメソッドは、メタプロトコルを通して定義される。また、制約の条件式で参照されている属性を持たないエージェントは場に入ることはできない。条件チェックで制約を満たす場合、エージェントは場に入ることができる。場に入ることで、場のメソッドを自動的に獲得し、場のすべての属性にアクセスできるようになる。場の中で常に満たしている制約が定義されている場合、その条件式で参照されている属性の値が書き換えられた直後に、条件チェックが起動される。条件チェックの結果、制約を満たしていない場合、実行は中断され、後の振舞はメタレベルの定義に委ねられる。定義はメタプロトコルを通して行なわれる。エージェントが場を出る場合も、場に入る場合と同様の手続きで動作する。以上の手続きにより、エージェントが場の出入りを通して常に場の制約を満たしていることが保障される。したがって、これらの手続きをメタレベル定義によってカスタマイズできない。メタで定義できる手続きは、場の制約をエージェントが満たさない場合の適応の処理のみである。

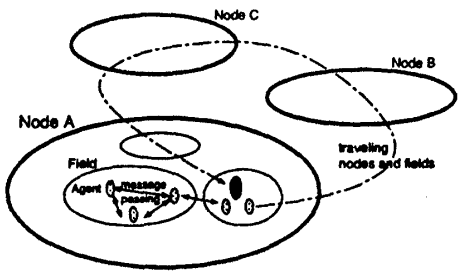

図 2: エージェントと場によるシステム構成

5　おわりに

Flageアーキテクチャのカーネル言語の機能概要を説明した。現在カーネル言語の処理系はprologで実装されている。またマシンノード間の通信(エージェントの移動など)には、TCP/IPに基づく通信関数を利用している。

謝辞　本研究は、産業科学技術研究開発制度「新ソフトウェア構造化モデルの研究開発」の一環として情報処理振興事業協会（IPA）が新エネルギー・産業技術総合開発機構から委託をうけて実施したものである。

参考文献

[1] Yonezawa, A et al.:*ABCL An Object-Oritented Concurrent System*, MIT Press,1989.

[2] Yaoqing, G et al.: "A Survey of Implementations of Concurrent, Parallel, and Distributed Smalltalk", ACM SIGPLAN NOTICES,Vol.28,No.9,1993.

[3] 粂野 文洋 他: "Field Oriented Language, Flage", ソフトウェア工学の基礎ワークショップ(FOSE'94), 日本ソフトウェア科学会,1994.