1C3-2

マルチエージェントシステムにおける エージェント間の結合関係記述のためのDSL

DSL to Describe Relations among Agents on Multi-Agent Systems

境靖夫*1　楢崎修二*2
Yasuo Sakai　Shuji Narazaki

*1長崎大学大学院
Graduate School, Nagasaki University

*2長崎大学工学部
Faculty of Engineering, Nagasaki University

In multi-agent systems (MAS), the relations among agents greatly influence their performances and their functions. Therefore, on the development of MAS, we think that a domain specific language (DSL) that can concisely describe the connection relations is useful for compareing many variation of relations. Based on of the observation of some MAS as algebraic data type. We propose such a DSL for MAS. Then, we implemented it on a pure functional language Haskell. This enables separations of agent design and relation design, static type checking and automatic parallel execution.

1. はじめに

マルチエージェントシステム(MAS: Multi Agent System)において、エージェント間の情報交換形態[1]の違いによって性能、実行時の経過及び結果が変わる例は少なくない。例えば、マルチエージェントリアルタイムA*(MARTA*: Multiagent Real-Time A*)[7]による迷路探索や、研究用に開発された分散センサネットワークによる乗り物の監視システム(DVMT: Distributed vehicle monitoring testbed)[8]においても、前者ならエージェント、後者ならセンサやそれを制御するプロセッサの結合関係が性能に影響する。また、最適化問題の粒子群最適化(PSO: Particle Swarm Optimization)においては、粒子間の結合関係が収束の早さや最適解に収束する可能性の高低といった性能に影響を与える[6]。そのため、MASを構築する情報交換形態は様々なものを容易に指定できることが望ましい。

情報交換形態の指定方法は様々なものが存在する。エージェント単位で情報交換の相手となるエージェントを個別に指定する方法は、どんな情報交換形態でも表すことができる。だが、ユーザが形成したい情報交換形態との対応が書きづらいため、この方法は利便性が低い。システム側で基本となる情報交換形態を用意し、ユーザ側でその情報交換形態に含むエージェントやその数を指定する方法は、書きやすい。だが、システム側で用意した情報交換形態が必要十分な数を揃えていない場合や、組み合わせられない場合、様々なものを指定できない。

そこで本研究では、まずはMASの情報交換形態に関して考察及び分析を行い、代数的記述による一般化を試みた。その代数的記述による一般化を元に、MASにおける多様な情報交換形態を可能な限り容易に記述するためのドメイン固有言語(DSL: Domain Specific Language)[2]の設計及び実装を行った。

本稿では、2章で情報交換形態について考察し、3章では2章で発見した性質から情報交換形態の代数的記述による一般化を試みる。4章では、それまでの章を踏まえて行ったDSLの設計・実装について述べる。最後に5章において、今後の課題を含め本研究に関してまとめる。

連絡先: 境靖夫 b60630213@gmail.com

*1　本研究では、エージェント間の情報交換の関係を示すものを指して情報交換形態と呼ぶ

*2　特定分野に特化した記述言語

2. 情報交換形態に関する考察

この章では社会・地理的シミュレータに関する[1][3]などや、ロボットに関する[2][4]などを対象とし、MASの情報交換形態について分析及び考察し、その性質を示す

これらの論文において、再帰的なエージェント[3]という考え方が頻出する。例えば、[1]でのField(道路網)、Fieldを分割したSubField(部分道路網)、SubFieldに含まれるオブジェクト、それら3つの関係や、[2]におけるシステムとその内部エージェントの関係は、複数のエージェントを含むエージェントによって再帰的に構成していると言える。そこで、我々はMASを再帰的なエージェントとして構成することにする。なお、本稿では、そのような再帰的なエージェントを再帰エージェントと呼ぶ。

次に、エージェント間の通信について考察する。多くのシステムにおいて、ある階層では、共通の型の情報が交換されることがある。また、それらの情報を集約したものが上位層へ送られることが多い。例えば、[2]のRescue MIKE内部のエージェント間通信は、タグとパラメータから構成されたプロポジションを介して行われる。また、[4]では、下位層で獲得された行動学習器群をより上位の層で組み合わせている。これらのことから我々は、再帰エージェントのもつ通信の機能とは、ある型の情報をエージェント間で交換すること、子エージェントの持つ情報を集約して上位層へ送ること、とする。なお、前者において受け手ごとの情報の対応を表した関数をfoldIn、後者の情報の対応を表した関数をfoldOutとする。

次に、エージェントの情報交換、行動、内部状態の更新について考察する。エージェントは、情報交換及び行動、それらによる内部状態の更新というサイクルを一般に繰り返している。例えば、[2]のRescue MIKEの内部エージェントであるMoniterエージェントは、World Modelが送る災害エリアの情報を収集し、分析して被害情報を生成しDirectorエージェントに送っている。そこで、エージェントは、情報交換、行動、内部状態の更新の3つを順に繰り返すものとする。なお、一度の繰り返し前後のエージェントの状態の対応を表す関数をnextStepとする。

以上の、再帰エージェントによる階層化、再帰エージェント内外の情報交換、エージェントと時間経過の関係、それらの分析及び考察により情報交換形態の性質を示すことができた。

*3　該当論文では明示的にエージェントと呼ばないことも多い

3. 情報交換形態の代数的記述

前章の考察を踏まえ、情報交換形態の代数的記述による一般化を試みる。

交換する情報の型をD、型Dの情報を交換する再帰エージェントとエージェントの型をRA_D、A_Dとすると、型RA_Dは式(1)のように定義できる。なお、式(1)は言語Haskellの型定義に近いが、型RA_Dに対応する型を示すだけであり、詳細は省略する。

$$\text{type } RA_D \;=\; A_D \mid S\,(RA_D) \tag{1}$$

式(1)はRA_DがA_Dもしくは$S\,(RA_D)$であることを示している。また、$S\,(RA_D)$は、要素がRA_Dである何らかの構造Sを示している。Sの構造は、対応する再帰エージェントの構造に依存し、再帰エージェントは構造によって必要な順序関係が異なる。そのため、要素間に必要な順序関係を示せる何らかの構造がSである。

前章の情報の集約に関する考察より、集約した情報をd_r、集約する複数の情報を$d_i(i = 1 \cdots n)$、2つの情報を合成する2項演算子を$\oplus$とするなら、式(2)のように情報の集約($\oplus$)を示せる。なお、d_r及びd_iの型はDである。

$$d_r \;=\; d_1 \oplus d_2 \oplus \cdots \oplus d_n \;::\; D \tag{2}$$

式(2)のように情報を集約する場合、型Dの集合は演算$\oplus$に関して半群である。情報を送らなかったエージェントからの情報d_iを単位元eと考えるなら、モノイドである[5]。

再帰エージェント内の子エージェントが受ける情報を式(3)、再帰エージェントが外へ送る情報を式(4)の型の関数によって表せる。

$$foldIn \quad :: S\,(RA_D, D) \to D \to RA_D \to D \tag{3}$$

$$foldOut \quad :: S\,(D) \to D \tag{4}$$

foldIn関数は受け手が受ける情報を表すために、親の再帰エージェントが受ける情報(第2引数)の他に、同じ親の複数の子エージェントが送る情報を必要とする。加えて、構造S上での受け手と送り手との順序関係を導くために、受け手(第3引数)と、送る情報に送り手を加えた組(RA_D, D)を要素とする$S\,(RA_D, D)$が必要である。foldOut関数は、複数の子エージェントが送る情報を引数とする。なお、上述の$S\,(D)$及び$S\,(RA_D, D)$を導くために、$S\,(RA_D)$からそれらへの写像関数が必要である。

以上より、情報交換形態を構成する再帰エージェント、その構造に対応する構造S、交換する情報Dを表し、対象MAS群の情報交換の構造を表すことができた。

4. DSLの実装

前章の代数的記述を基にしたDSLの実装について述べる。本DSLは、純関数型言語Haskell上のEmbedded DSL(EDSL)として、つまりはHaskell上の一部として実装する。

まずは、本章での実装を明確にするために、ユーザ及び各設計者の役割について述べる。DSL提供者は、前章の代数的記述に基づいたDSL実装部を提供する。この上に、ネットワーク設計者は、問題に依存しない再帰エージェントをfoldIn、foldOutをネットワークにあうように設計し、ライブラリとして提供する。一方、特定の問題を考えるプログラマは、その問題固有のエージェントを提供する。先の問題固有のエージェントと、問題非依存のネットワークライブラリを用い、DSLユーザはMASを構成できる。

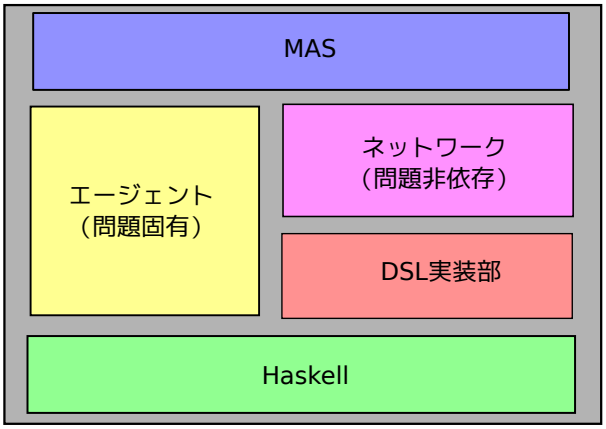

Figure 1: 記述部分の階層

上記の関係を表したものが、Figure 1である。これにより、ユーザ及び各設計者は各自の記述に専念できる。

これ以降、本章ではDSL実現部に関して主に扱う。なお、問題固有のエージェントについては詳細を省く。前章より、DSLでは、再帰エージェント自体とその情報交換を示す関数を記述するための機能を提供する[4]。そのために、型クラスRAを提供する。型クラスとは、型の性質を示すものであり、特定の型クラスに属する型をその型クラスのインスタンスと呼ぶ。型クラスRAは、前章で定めた再帰エージェントの性質を示している。また、前章では、型RA_Dの値が個々の再帰エージェントに対応していたが、本実装では、共通の性質(情報交換の規則)を持つ再帰エージェントを一つの型に対応付けし、異なる型の再帰エージェントは同じ型クラスRAに属するものとする。

型クラスRAの定義を以下に示す。

```
1 class (Monoid d) => RA ra d where
2   data S ra :: * -> *
3   foldIn :: (Functor (S ra))
4     => (S ra) (RAW d,d) -> d -> RAW d -> d
5   foldOut :: (Functor (S ra))
6     => (S ra) d -> d
```

この型クラスはインスタンスに対して型の指定や関数の定義を要求しており、行ごとに何を要求しているかを以下に示す。

1行目　再帰エージェントの型raと、交換する情報の型であり型クラスMonoidのインスタンスでもあるdの指定

2行目　再帰エージェントraで用いる構造の指定
（(S ra)が前章のSに相当）

3,4行目　指定した型に沿った関数foldInの定義
（(S ra)は型クラスFunctorのインスタンス）

5行目以降　同様に関数の定義

なお、Monoidはモノイドであること、Functorは順序関係を保った写像が可能なコンテナであることを示す。また、RAW dは実装の都合上定義した型であり、種類ごとに異なる型である再帰エージェントを一つの型RAW dに見せかけている。Haskellにおいては、データ構造に含む要素を基本的に同じ型に限定してい

*4　正確には、再帰エージェント同士を組み合わせる機能も必要

るため、本実装においては一つのデータ構造に違う型の要素を混在させるために、存在型を使ってRAW dを定義することが必要となる。

次に、2章で取り上げたnextStepの実装について述べる。実は、nextStepはエージェントの役割や問題に応じて2種類必要である。一つは、問題固有のエージェントごとに存在し、内部状態の更新を必要とするものである。もう一方は、問題に依存しない再帰エージェントにおいて共通で統一的な記述が行えるものである。

以下に、それらの関数の型を示す。

```
7   nextStep :: ra -> d -> ra
```

次に、処理の順序を、問題固有のエージェントのnextStepについて示す。

1. 自身の内部状態を更新する
2. 自身が次ステップで送る情報を生成する
3. 次ステップで送る情報を保持し、更新した自身を返す。

次に、処理の順序を、再帰エージェントのnextStepについて示す。

1. 関数foldInにより子エージェントの受ける情報を個別生成
2. 子エージェントと、受ける情報を引数とし再帰呼び出し
3. 関数foldOutによって自身の送る情報を生成
4. 次ステップで送る情報を保持し、更新した自身を返す。

このnextStepを繰り返すことにより、エージェントの時間経過を表すことができる。

また、上述のnextStepの子エージェントごとの呼び出しや、情報の集約について、並列処理による処理時間の短縮を考える。その場合、型クラスRAのインスタンス定義において、構造($S\ ra$)や型クラスRAの定義上の各関数を並列処理に対応させることにより、再帰エージェントから情報交換形態を構成するユーザは並列処理を意識することなく並列処理に対応した再帰エージェントを用いることができる。

以上より、前章の代数的記述に則った実装が可能であり、実際に情報交換を行う実装が可能であることを示すことができた。

5. まとめ

本研究では、対象MAS群の構造に対して分析及び考察を行い、それに基づきその構造に関して代数的記述を行い、DSLを実装した。実装したDSLは、再帰エージェントによる階層ごとの結合関係の組み立て、モジュール化による高い再利用性、ベース言語であるHaskellの静的型検査を利用した実行を介さない個別の型検査、再帰エージェントの型ごとの並列処理への対応の可能性、などの利点を挙げられるものとなった。

しかしながら、本稿では環境（交換情報）変化や時間経過による対象MAS群の構造の変化に関しては触れておらず、それに関しての考察と実装が今後の課題である。

References

[1] 尾崎 敦夫, 古市 昌一, 阿部 一裕, 中島 克人, 田中 秀俊. 大規模並列交通シミュレータの実現と負荷分散方式の評価. 情報処理学会誌, 40(6):2810–2818, June 1999.

[2] 久保 長徳, 森下 卓哉, 下羅 弘樹, 河原林 友美, 小高 知宏, 小倉 久和, FRANK Ian, 田中 久美子, 田所 諭, 松原 仁. Rescue MIKE：災害シミュレーション実況システム：version 0 の設計と実装. 人工知能学会論文誌, 21(4): 388–397, November 2006.

[3] 鳥居 大祐,石田 亨. 社会・環境シミュレーションにおけるインタラクション層の構築. 人工知能学会論文誌, 21(1):28–35, 2006.

[4] 高橋 泰岳. 階層型学習機構における状態行動空間の構成. 日本ロボット学会誌, 21(2):164–171, March 2003.

[5] 堀田 良之. 数学シリーズ 代数入門-群と加群-, chapter 1, pages 6–7. 裳華房, September 1987.

[6] Rui Mendes and James Kennedy and Jose Neves. The Fully Informed Particle Swarm: Simpler, Maybe Better. IEEE Transactions of Evolutionary Computation, 8(3):204–210, June 2004.

[7] Kevin Knight. Are many reactive agents better than a few deliberative ones? In Proceedings of the 13th international joint conference on Artifical intelligence, volume 1, pages 432–437, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[8] Victor R. Lesser and Daniel D Corkill. The distributed vehicle monitoring testbed : a tool for investigating distributed problem solving networks. AI Magazine, 4(3):15–33, 1983.