F_020

マルチエージェント・システムを用いた複雑系モデルの計算環境TEA

The Computational Environment "TEA" for Models of Complex Systems Using Multi Agent Systems

赤嶺 有平† 根路銘 もえ子‡
Yuhei Akamine Moeko Nerome

1．まえがき

代表的な複雑系モデルであるマルチエージェント・システム（MAS）及びセルオートマトン（CA）は，エージェントもしくはセル間の相互作用を定義することで様々な現象を再現できるとされており，多くの現象に対して適用が試みられている．これらのモデルは，相互作用の定義そのものよりも，シミュレーション結果が大きな意味を持つためシミュレータの役割が非常に大きい．しかも，モデルの性質上，計算量が多くなる傾向がある上に本質的に並列性を持つためシミュレータの性能が重要な意味を持つ．そのため，多くのシミュレータが開発されており[1][2]，記述の容易さ，高速性などそれぞれ特徴をもっている．

本研究では，MAS を簡潔に記述可能なプログラミング言語 ASSAM(Agent Simulation Scripting and Manipulating) の設計及び，それを用いた複雑系シミュレーションシステム TEA の開発を行った．提案システムの特徴は，モデル記述の容易さと並列演算性能の両立にある．また，多くの GUI を備え計算機やプログラミングに不慣れなユーザでも抵抗なく利用できる設計になっている．

ASSAM は，CA モデルと MAS を統一的に記述可能な言語である．ASSAM は，筆者らが CA モデルを記述するために設計した DORA 言語[3][4]を拡張したものである．ASSAM は，CA と MAS を融合するためにエージェントの移動先をセル単位に限定している．したがって，エージェントは離散空間を移動する．また，離散空間に限定することで，コンパイラによる自動並列化が行いやすくなる．

本システムは，設計用インタプリタ及び実験用コンパイラの二つのコンポーネントからなる．インタプリタは，即座に実行できるためモデルの設計段階にありがちな試行錯誤による調整(Trial and Error Adjustment)を効率的に行うことができる．一方，MAS はエージェントの数に比例して計算量が増加するため大規模なシミュレーションを行うには高速化が必須となる．提案システムは，コンパイラによる高速なシミュレータを提供する．

2．ASSAM 及び TEA の対象モデル

ASSAM は，離散空間を移動するエージェント群及びセルオートマトン・モデルの実装を目的として設計されている．離散空間とは，具体的には同サイズ，正方形のセルが敷き詰められた空間（セル空間）のことであり，エージェントはセル間を移動する．移動先をセルに限定することで，近傍エージェントの特定が容易になり，コンパイラによる最適化に貢献する．

エージェント及びセルは，内部状態を変数として保持する．外部情報は，近傍セルの状態でありエージェントはセルの内部状態に含まれる．

† 琉球大学
‡沖縄国際大学

3. ASSAM の言語使用

ASSAM の演算子及び構文は，C 言語を参考に設計されている．ただし，変数宣言は不要であり，最初に出現する代入文における右辺の型により決定する．Variant 型ではないため，一つの変数に異なる型の値を代入することはできない．ASSAM で利用できる変数の型を以下に示す．

3.1 ASSAM で利用できる変数の型

a) 整数型，実数型

C 言語の int 型，double 型に相当する．利用できる演算子も C 言語と同様である．

例）a = 1; //a は整数型として宣言され，1 が代入される
　　b = 1.0; //b は実数型として宣言され，1 が代入される

b) ベクトル型

近傍セル参照の際の位置指定等に利用されるベクトル変数の型である．同型及び他の型との間で数学的な意味を持ついくつかの演算子が定義されている．

ベクトル型は，エージェントの移動先の指定や状態を取得する近傍セルの位置を指定するときに利用される．この際，後に述べる trasform 構文で設定された行列によって座標変換が行われる．本稿では，このような座標変換を伴う操作を参照演算と呼ぶ．

例）c = [0, 1];　//c はベクトル型として宣言され，
　　//(0,1)が代入される
　　d = [1, 0] + c;//d はベクトル型として宣言され，
　　//(1,0) + (0,1)の結果が代入される
　　e = c.state1;　//以下参照

セルの状態として state1 が宣言されている場合，(0,1)に対して参照演算を行った位置にあるセルの状態 state1 の値が e に代入される．ピリオドは，ベクトル値の指定する位置にあるセルの状態にアクセスする参照演算子である．参照演算子の左辺は，ベクトル値そのものも許される．

例）f = [2, 0].state1;

セルが保持する状態は，以下の構文で宣言する．

構文）cell { *variable_type variable_identifier*, ... };

c)行列型

正方行列に限定されており，座標変換等に利用される．ベクトル型と同様に数学的な意味を持つ演算子が利用できる．

例）g = matrix(0,0,0, 0,0,0, 0,0,0);
　　//g は 3x3 行列型となり，全要素が 0 となる．

d)エージェント型

現バージョンの ASSAM においては，エージェントは C 言語における構造体と，定義・操作の両面において非常に近い仕様になっている．構造体との違いは，エージェントの実態（メモリ領域）がセルに結び付けられているという点である．エージェントは，agent キーワードを用いて以下のように宣言する．

構文）agent *agent_name* {
variable_type variable_identifier,
:
};

エージェント型は，実態に対する参照なので，代入は参照の増加を意味する．実態は，new 演算子で生成する．

構文）new *agent_name*(*initial_value1*, *initial_value2*, ...)

例）agent ex_agent { int h; };
//内部状態として整数値を一つ持つエージェント
//ex_agent を宣言
i = new ex_agent(1);
//ex_agent の実態を生成，i はエージェント型として宣
//言され生成された実態への参照が代入される

e)配列型

同型な複数の値を保持できる．配列のサイズは自動で伸縮する．配列変数の識別子は$から始まる．

例）$array = { 0, 1, 2, 3};
$array_array = { $array, {1, 2, 3, 4} } ;

また，*$agent_name* には存在する全てのエージェント，$cell には全てのセルが保持されている．

3.2 ASSAM の構文

基本的な構文は C 言語に準ずるため，C 言語と異なる ASSAM 独自の構文のみ以下に示す．

a) *agent_type_variable -> reference_type_variable*
agent_type_variable が保持するエージェントの実態を *reference_type_variable* が示すセルに移動する．

b) for_each(*iterator* of *array*) *statement* ;
配列変数 *array* 内の全ての値を一つずつ取り出して，*statetment* を実行する．取り出された値は変数 *iterator* に代入される．

c) transform(*matrix_type_value*) *statements* ;
statements は，一つ以上の命令文である．*statements* に含まれる全ての参照演算に対して，行列値 *matrix_type_value* が適用される．

4．ASSAM による記述例

本節では，ASSAM による記述例として，学術的な意味はもたないが，単純な規則で行動するエージェントと単純な規則で状態遷移するセルを記述している．

エージェント ex は，内部状態として進行方向 dir を保持し，以下の規則で行動を決定する．

- 進行方向に壁があるか他のエージェントが存在すれば，進行方向を反時計回りに 90 度回転する
- 進行方向になにも何もなければ，1 セル分前進

また，セルは，4 近傍全てにエージェントが存在し自身にエージェントが存在しない場合，壁に遷移する．以下にソースコードを示す．

```
agent ex { ref dir };
cell { int type };
static front = [0,-1];
static left  = [-1,0];

for_each(an_agent of $ex) { //$ex はすべての ex エージェント
  transform(
  local(posof(an_agent),front, an_agent.dir)) {
  //local は第 1 引数を原点として第 3 引数のベクトルが第 2 引数のベクト//
  ルの向きになるような変換行列を生成する
    if(exist(front.ex) || front.type != 0 ) {
      an_agent.dir  =
              transform(left)  -  posof(an_agent);
    }else{
      an_agent -> front;
    }
  }
}
$neighbors = {[-1,0], [1,0], [0,-1], [0,1]};
for_each(a_cell of $cell) {//$cell はすべてのセル
  transform(local(a_cell)) {
    sum = 0;
    for_each(neig of $neighbors) {
     sum += exist(neig.ex);
    }
    if(sum == 4 && !exist([0,0].ex))
      [0,0].type = 1;
  }
}
if(time() % 5 == 0) //5 時間ステップに 1 回エージェントを生成
```

5．おわりに

本稿では，複雑系を対象としたマルチエージェント・システム計算環境 TEA の特徴，及びプログラミング言語 ASSAM の言語仕様について述べた．ASSAM は，複雑系モデルの記述を目的として設計されているため，ある集合の各要素に同様な操作を繰り返し行うような処理を簡潔に記述可能である．このような記述は，分散・並列化に適しており，今後はこれらを自動並列化するコンパイラの開発を行っていく．

参考文献

[1]North, M.J., N.T. Collier, and J.R. Vos, "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit," ACM Transactions on Modeling and Computer Simulation, Vol. 16, Issue 1, pp. 1-25, 2006
[2] 山影 進, 服部正太, コンピュータのなかの人工社会－マルチエージェント シミュレーション モデルと複雑系－, 共立出版(株)
[3] 赤嶺有平, 遠藤聡志, 山田孝治, 拡張 SIMD 命令を用いたセルオートマトンシミュレータ用並列化コンパイラ, 情報処理学会論文誌, Vol. 45, No. 10, pp.2443-2453,2004
[4]Yuhei Akamine, Satoshi Endo and Koji Yamada, The Development of a Computational Environment for Cellular Automata, IEICE Transactions on Information and Systems,Vol.E88-D No.9, pp.2105-2112,2005