6-4

モデル理論アプローチによるシミュレーション

日大生産工(院)　○前川　徹　日大生産工　柴　直樹

1. はじめに

本研究では、数学である集合論表記(set notation)やオートマトンシステムを使ってシミュレーションモデルを記述することが出来るモデル理論アプローチ(Model Theory Approach: MTA)を使いシミュレーションを行う。そのために MTA の紹介や解説を行い、「宣教師と人食い人」問題の問題解決システムを例にあげて説明する。さらに MTA によるシミュレーションとして、旭による「じゃんけん集団の進化ゲーム(evogame)」[1]の紹介を行う。最終的には、他のシミュレーションの題材を捜し、MTA を使用したシミュレーションモデルとして実装することが目的ではあるが、本稿はそのための予備的な研究報告である。

本稿では、2 節で形式手法モデル理論アプローチ[2]についての説明と解説をする。3 節で「宣教師と人食い人」問題を例とした MTA による問題解決システムの説明をする。4 節で「じゃんけん集団の進化ゲーム(evogame)」を例としたシミュレーションの説明をする。5 節で結論を述べる。

2. 形式手法モデル理論アプローチ

2.1 形式手法

形式手法(Formal Method)は、モデルを記述するときに、自然言語や図、グラフなどを使うのではなく、数学の言葉を用いて記述を行う。数学の言葉を用いることにより、表現のあいまいさを排除して複数の解釈が生じないようにすることが可能となる。逆に、モデルを記述するときに、自然言語や図、グラフなどを用いる手法は非形式手法という。MTA は形式手法に属する形式的アプローチである。

2.2 システムアプローチ

システムアプローチは、対称に対してシステムモデルをもち、システム概念及び、システムモデルを使ってモデル化を行う。システムモデルの一つとしてはオートマトンシステムがあり、オートマトンシステムモデルを使用する MTA はシステムアプローチである。

2.3 モデル理論アプローチ

モデル理論アプローチ(MTA)は、モデルの記述を行うときに、数学的な言葉、特に集合論を使用する形式的アプローチである。

MTA の対となる非形式的アプローチでは、いくつか指摘されている問題点がある。問題点としては、未熟なコード化を行うことにより、結果として手戻りコストが大きくなることがあげられる。モデル記述で自然言語や模擬コードによる記述を行うと信頼性は低くなり、それを補うための見直し、検閲が必要になる。それに加えて開発者の誤解が生じる可能性などもある。また、エラーの検出や顧客からのフィードバックの遅さなどもある。

形式的アプローチでは非形式的アプローチに対して、以下の利点を持つとされている。

①仕様の矛盾・欠陥を仕様作成時に発見しやすくなる。

②形式的表現により記述の信頼性が高まる。

③手戻りの回避が可能となる。

Simulations based on the model theory approach

Toru MAEKAWA and Naoki SHIBA

④機能の追加が容易、モデル自体が文書として使える。

よって、形式的アプローチである MTA はこれらの利点を有している。

また、MTA が従来の形式的アプローチと異なるところとして、システムアプローチであるということがある。

2.4 MTA のシミュレーション・問題解決

MTA のシミュレーションでは、モデルの記述に集合論、システムモデルにはオートマトンを使用する。また、モデルで問題解決をシミュレーションすることも可能であり、これを問題解決システム(solver system)と呼ぶ。問題解決システムは問題仕様環境（問題の実質的な定義）を入力とし、解を出力する入力－出力システムである（図 1 参照）。その内部は、問題解決のプロセスをオートマトンとしてモデル化したプロセスと goal-seeker（汎用の救解プログラム）からなる。図 1 の goal-seeker は、問題を解決する活動を管理するものであり、現在の状態(c)を入力とし、各状態を評価する関数 goal(c)を使いながら、適切なアクションをアクションの集合 genA(c)から選び出力する。

goal-seeker で重要な事がある。問題解決を行っていく途中で行き詰った場合（選べるアクションが無くなった場合）ひとつ前の状態に戻り、やり残したアクションを使って、プロセスを再出発させる。これをバックトラック(back track)と呼ぶ。問題解決プロセスではバックトラックを繰り返すことにより、正しい解を発見することが可能になる。なお、MTA では標準化された goal-seeker が装備されているため、プロセス部分のみのモデルを集合論表記を用いて構築すると、MTA のセットコンパイラにより実行可能なコードに変換され、自動的に問題解決システムが生成される。

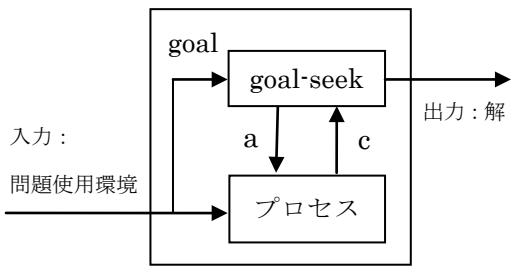

図 1 solver system

3. MTA の問題解決システム例

3.1 「宣教師と人食い人」問題

川の左岸に、宣教師 3 人と人食い人 3 人の合計 6 人がいる。ボートを使って 6 人全員を以下のルールに基づいて右岸に渡らせる。

・6 人は全員ボートの操縦が可能。

・ボートは最大 2 人乗りで、無人で動かすことは不可能。

・どちらかの岸で宣教師の数＜人食い人の数となると宣教師は人食い人に食べられてしまう。

この問いをシミュレーションで解決する。

3.2 シミュレーション方法

問題のモデルを集合論表記で記述した missionary.set というモデルを作成し、MTA の solver system を使って問題解決を行った（モデルのソースは付録 1 を参照されたい）。

3.3 シミュレーション結果

図 2-1 では今回のシミュレーション実行時において、状態の評価値（付録 1 における goal(c)）の変化を、問題解決システムがグラフとして出力したものである。モデルに記述された終了条件を満たすために、プロセスはバックトラックにより実行されていく（グラフの縦軸は、左岸に居る人の数）。

図 2-2 はモデルのシステムによるシミュレーション結果の出力である。今回のモデルでは、システム全体の状況を 3 つ組(m,e,b)で表わしている。ここで m,c,はそれぞれ左岸に居る宣教師と人食い人の数、b は舟が左岸に居るか(“l”)、右岸に居るか(“r”)を表わす。これを終了条件が満たされるまで行った結果（[1,1](右へ),[1,0] (左へ),[0,2] (右へ),[0,1] (左へ),[2,0] (右へ),[1,1] (左へ),[2,0] (右へ),[0,1] (左

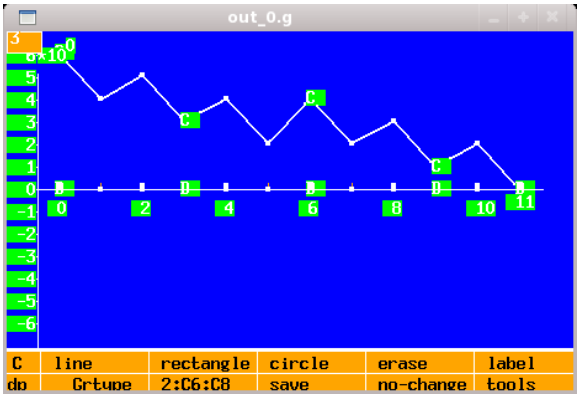

図 2-1 missionary.set の goal(c)状態変化

```
"MODEL:""out_0.g"" GO"
"*****************************"
"[state C   ]="[1,1,"l"]
"[As        ]="[[1,0],[1,1],[0,1]]
"repeated state"
"constraint violated"
"[goal(As)  ]="[1.0000e+12,0,1.0000e+12]
"[selected A]="[1,1]
"[next C    ]="[0,0,"r"]
"[RA        ]="nil
"MODEL:""out_0.g"" GO"
"Solf="[[1,1],[1,0],[0,2],[0,1],[2,0],[1,1],[2,0],[0,1
],[0,2],[1,0],[1,1]]
"Yf="[1,1,"l"]
WARNING:predicate 'postprocess()' is not defined!!!
missionary.p" ends"
setcompiler.p" ends"
"normal reset state"
"May I help you?"
X>
```

図 2-2 missionary.set の結果

へ),[0,2] (右へ),[1,0] (左へ),[1,1] (右へ)) が表示され、この図のシステムの出力では終了条件を満たすオートマトンの入力の系列が Solver によって求まっているのが分かる。

4. MTA のシミュレーション

4.1 じゃんけん集団の進化ゲーム[1]

競合関係が三すくみの関係にある 3 種類の生物の人口分布の変化を動的に計算する進化ゲーム(Evolutionary Game)のシミュレーションを行う。ここで、三すくみの関係とは、生物 1 は生物 2 より強い、生物 2 は生物 3 より強い、生物 3 は生物 1 より強い、ということを言う。そのため、生物 1,2,3 のそれぞれをじゃんけんでのグー、チョキ、パーとして考える。ここでは、人口分布のバランスが崩れた場合にその後の人口分布がどのように変化していくかをシミュレートする。

4.2 シミュレーション方法

競合している生物 1,2,3 が混在する状態を考える。現在の人口分布を c = (c1, c2, c3) ,(c1+c2+c3=1)とする。各個体は 1 対 1 でランダム対戦を行ない、適応度 w1, w2,w3 が定まる。次世代の人口分布 cc = (cc1, cc2, cc3)は, 各生物の適応度が高い方がより多く子孫を残せるように決まる。旭によるモデル evogame5.set では初期値 c0:=[0.5,0.3,0.2]、期間 times()=1000 となっている(本稿では times()=3000 で実行した)。なお、モデルの集合論記述 evogame5.set は紙面の制限より省略する。詳しくは文献[1]を参照されたい。

4.3 シミュレーション結果

図 3 の三角形では、左下がグー、右下がチョキ、頂点がパーであり、グー、チョキ、パーのそれぞれが、人口の全体を占めている点を表わす。シミュレーションでは最初、3 種のなかで一番多いグーが増加していくが、餌となる（勝つことが出来る）チョキの数が減少していくため、子孫が残せなくなり減少していく。一方、パーは餌となるグーが多くいる為、増加していく（餌となるグーはさらに減少する）そのため、今度はパーが 3 種の中で一番多くなっていく。このパーの状況は先ほどのグーと同じ状況なので、今度はパーが減少してチョキが増加していく。この流れを繰り返していくことにより、図 3 の状況が得られた。よって、3 種はバラン

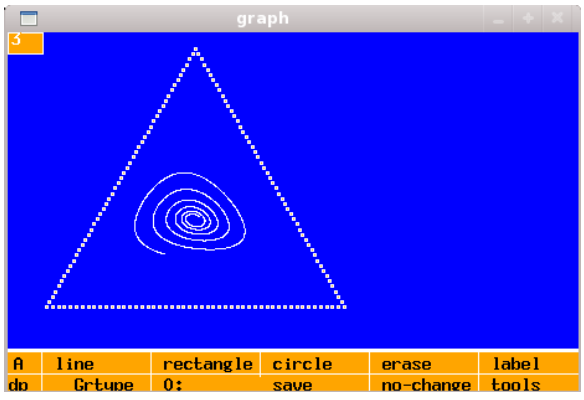

図 3 evogame5.set 実行結果

スのとれた人口分布（(多型集団の）進化的安定状態（Evolutionarily Stable State: ESS)）へ向かっていくことが分かる。

5. おわりに

MTA を使用したシミュレーションでは、形式手法である集合論記述やシステムアプローチにおける代表的なシステムモデルであるオートマトンなど、数学的な記法を使用する。これらによりモデルを記述する場合に数学の言葉で表わすことができるようになり、あいまいさの除去や複数の解釈など非形式手法で問題となることを排除することが出来ることが分かり、実際にその表記を行うことが出来た。

今後の展望としては、今回得たことを踏まえて、数学的な記述が更に有効活用できるような複雑なモデルを記述し、このアプローチの有用性の追求をしていくということがある。

参考文献

[1] 旭　貴朗　モデル記述言語 CAST によるシミュレーション（モデル理論アプローチの応用）
(http://www2.toyo.ac.jp/~asahi/research/simulation/index.html)
最終アクセス 09/10/29

[2] 高原　康彦　他著「形式手法 モデル理論アプローチ」日科技連　2007


```
/*missionary.set*/
delta(c,a)=c2 <->
        (project(c,3)="l") ->
                (X1 := project(c,1) , Y1 := project(a,1) , A1 := X1-Y1 ,
                X2 := project(c,2) , Y2 := project(a,2) , A2 := X2-Y2 ,
                c2 := [A1,A2,"r"]),
        (project(c,3)="r") ->
                (X1 := project(c,1) , Y1 := project(a,1) , A1 := X1+Y1 ,
                X2 := project(c,2) , Y2 := project(a,2) , A2 := X2+Y2 ,
                c2 := [A1,A2,"l"]),
        constraint(c2);
genA(c)=As <->
        (project(c,3)="l") ->
                (As := defSet(p(z,x,[c]),member(x,[[2,0],[1,0],[1,1],[0,1],[0,2]]))),
        (project(c,3)="r") ->
                (As := defSet(p(z,x,[c]),member(x,[[2,0],[1,0],[1,1],[0,1],[0,2]])));
p(z,x,[c]) <->
        (project(c,3)="l") ->
                (project(x,1) <= project(c,1) , project(x,2)<= project(c,2), z:=x) ,
        (project(c,3)="r") ->
                (project(x,1) <= 3-project(c,1) , project(x,2)<= 3-project(c,2), z:=x) ;
constraint(c) <->
        (project(c,1) <> 0) -> (project(c,1) >= project(c,2)) ,
        (3-project(c,1) <> 0) -> (3-project(c,1) >= 3-project(c,2));
initialstate( )=c <-> c := [3,3,"l"];
finalstate(c) <-> c=[0,0,"r"];
st(c) <-> finalstate(c);
goal(c) = r <-> r := project(c,1)+project(c,2);
```

付録 1 「宣教師と人食い人」問題のモデル記述 missionary.set