〔駒沢女子大学　研究紀要　第15号　p .87 ～ 102　2008〕

artisoc で作成する初等情報教育用電子回路シミュレーション

末　木　俊　之

Electronic Circuit Simulation with Artisoc for Beginning Class Information Education

Toshiyuki SUEKI

１．artisoc の機能

昨今いろいろな学問分野で、構成論的研究の道具としてコンピュータが使われるケースが多くなってきている。手軽に使えるシミュレーションツールも多数開発されている。

研究手法が、個々のエージェント（ノード）が局所的情報に基づき自律的行動をとる系の全体的な振る舞い・様子を観察する手法であるので、シミュレーションツールもその手法に合致するものが登場している。基本的に下記①～⑤の性質を持っている。

①個々のノードの振舞いをプログラムできて、自律的に動作させることができる。

②個々のノードが、自己の周囲の局所的情報を容易に入手できる。

③個々のノードに、周囲のノードとの繋がり・関係（ネットワーク）を設定でき、またそれを変化させることも容易である。

④個々のノードの状態を色の変化などで容易に表示できる。

⑤ノード同士の関係を、線・矢印付き線として容易に表示させることができる。

artisoc はそれらシミュレーションツール（ソフト）の１つである。『人工社会構築指南 artisoc によるマルチエージェント・シミュレーション入門』[1] のまえがきによると、『本書は、科学研究費補助金学術創成研究費による「マルチエージェント・シミュレータによる社会秩序変動の研究」（平成15-19年度）の成果の一部です。この研究費により、KK-MAS をプロトタイプにした artisoc の開発が可能になりました。』とある。著者グループの社会学系研究のために開発されたシミュレーションソフトであろう。

正確には artisoc には、artisoc academic と artisoc textbook の２種類ある。artisoc textbook は、artisoc academic の簡易版で、書籍『人工社会構築指南 artisoc によるマルチエージェント・シミュレーション入門』[1] に付録として付いている CD － ROM に記録されており、読者本人が所有または使用するコンピュータ１台のみにつきインストールして使うことができるものである。同書の説明によると、artisoc textbook には、書き込めるルールの総計が200行までであること、入出力できるテキストファイルのファイル名が「input.txt」と「outuput.txt」の２つのみであるという２つの制限がある。

今回の研究紀要で作成したシミュレーションは、artisoc textbook で作成したものである。

artisoc は、高度なシミュレーションを容易に作成することができるが、『人工社会構築指南 artisoc によるマルチエージェント・シミュレーション入門』[1] の例題として掲載されている実例を３つほど挙げ、今回の研究紀要でシミュレーションを作成する折に着目した性質６

つ（①～⑥）を列挙する。

（実例１）ランダム・ネットワークシミュレーション

（図１.）は、第27章[1)] の説明に忠実に従って実際に作成したランダム・ネットワークシミュレーションの実行例である。

20個のノード（丸い点）を円周上に配置して、ノード同士をランダムに連結させるシミュレーションである。

任意のノード２個を選択し、線でつなぐ単純なシミュレーションである。

時間の経過と､クラスター数､最大クラスターサイズの変化がグラフによって見て取れるようになっている。

①ノードを配置する平面（縦･横のサイズ等)、ノードの表示形態などは、シミュレーションモデルの定義段階の作業で決まり、artisoc エンジンが自動的に表示してくれる。プログラムには無関係である。

②シミュレーションの条件（初期条件等）を変更するためのコントロールパネルを使うことができる。図右上に表示されているコントロールパネルは、スライドバーの操作により、表示させるノード数を20 ～ 100個の任意の数に設定できる。

③グラフ表示も定義作業のみで、自動的に書かせることができる。グラフ化したいデータ選択とグラフ種類の定義を行えば自動的に表示される。シミュレーション進行に合わせたデータの時間変化を表示するグラフが表示される。

④ノード同士を結ぶ線は、arisoc エンジンが自動的に描いてくれる。ノードにエージェント集合型変数を用意し、線を引かせる対象としてそのエージェント集合型変数を定義する作業が必要なだけである。artisoc ではノードのことをエージェントと呼び、ここで言うエージェント集合型変数とは、複数のノードをまとめて保持できる変数のことである。

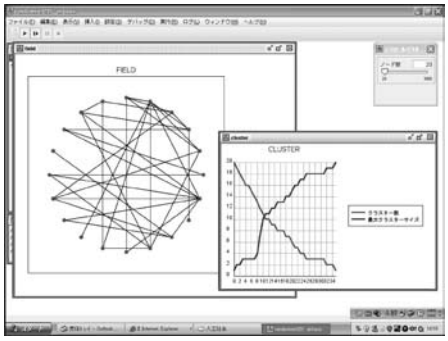
（実例２）空を飛ぶ航空機

（図２.）は、第33章[1)] の説明に従って実際に作成した単純なシミュレーションである。背景の空を飛行機が飛んでいるだけのものである。この場合にはノードは１つの飛行機であるが、ノードは小さな画像で表示されている。そして動く方向によって画像を切り換えて表示する。（図３.）がこのシミュレーションで使われている飛行機の画像群である。

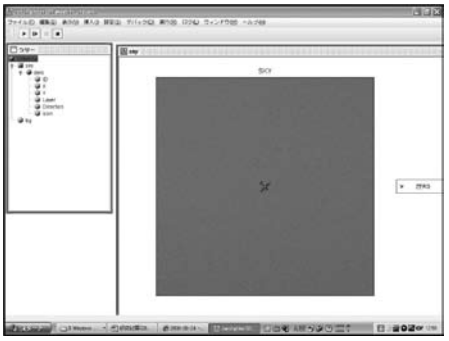
⑤ノードは、画像に置き換えて表示させることができる。

（実例３）世界地図表示（首都表示付き）

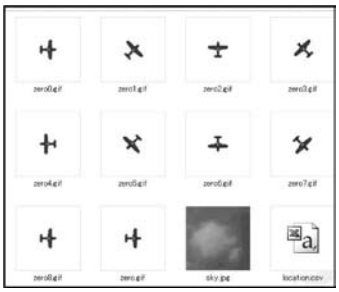
（図４.）も、第33章[1)] の説明に従って実際に作成した単純なシミュレーションである。背景の世界地図画像上に赤丸で148個の capital エー



（図１.）ランダム・ネットワークシミュレーション



（図２.）飛行機のシミュレーション



（図３.）飛行機表示用画像等

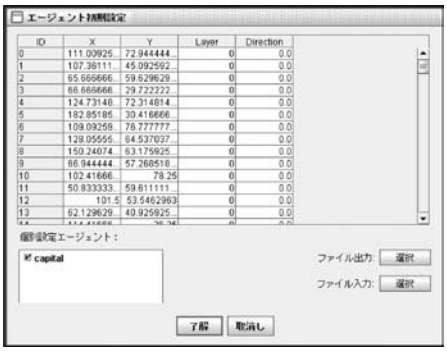
ジェント（各国の首都）が表示されている。あらかじめカンマ区切りの CSV テキストデータとして用意されている座標データを、capital エージェントの初期値データとして読み込んで使っている。

artisoc では、シミュレーションに登場する要素の全体構成は、ツリー画面として表示される。（図４.）左上に表示されているツリーを見ると、capital エージェントは５つの変数を持っていることがわかる。

（図５.）は、capital エージェントの初期設定画面を呼び出したものである。この画面に表示されるものは、各エージェントが保持する変数の内容である。X､Y という変数は､各エージェントが表示されるシミュレーション画面上の座標データである。

⑥ノード変数の内容は、カンマ区切りの CSV テキストファイルとして読み込み、書き出しが可能である。これは表計算ソフトエクセルで作成できる。

（ただし、エージェント集合型変数の読み込み、書き出しはできない。エージェント集合型変数は､エージェントの初期設定画面には現れない。）



（図５.）エージェント初期設定画面

２.　電子回路シミュレーション作成

２.１　シミュレーションの構想

artisoc は、人文科学系研究用シミュレータという位置づけであろうが、座標データ次第でノードを自由に配置でき、ノードを簡単に線で結びつけて表示させることができるというのは魅力である。ネットワーク的なものをパソコン上でシミュレーションするのに便利である。

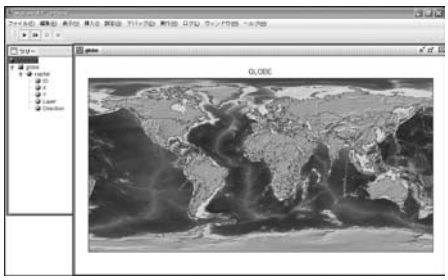
初等情報教育の分野だと、まずは簡単な電子回路動作説明用のツールとして使えそうであることに気がつく。

単純に言えば、電子回路も基本素子を電線のネットワークでつないだものであろう。

シミュレーション画面上に基本素子を配置し、連結し、特別に電流を ON ／ OFF できるノードを配置し、その状態をコントロールパネルのスライダーまたはトグルボタンで切り換えられるようにすれば仮想的な電子回路が画面上に構築されるであろう。

電子回路シミュレーションの概要は以下となる。

①画面上に配置するノードのタイプをいくつ



（図４.）世界地図表示シミュレーション

か決め、タイプごとに違った動作をさせる。

②素子・状態可変なトリガー・数値表示用ノードは、ペイントブラシソフトで作成した小さな画像で表示する。

③電線に相当するノード間をつなぐ矢印は、電流の ON ／ OFF に従い、赤／青と、矢印の色を変化させる。

矢印を引くノードを格納するためのエージェント集合型変数を用意する。しかし、artisoc にはノード間の線・矢印色を変化させる機能はないので、１つのエージェント集合型変数では不可能である。

従って、ノードには２種類のエージェント集合変数を持たせることとした。片方は赤色、もう１つは青色で矢印を引かせるために使用する。

④ノードデータ（表示座標データ等）はエクセルシート上で作成し、完成後カンマ区切りの CSV テキストファイルで保存し、エージェントの初期設定画面の機能を使い、artisoc に読み込ませる。

（図６.）は実際に作成したノード（node エージェント）のエージェントの初期設定画面の様子である。

図左には、ノード（node エージェント）の変数群がツリー画面で表示されている。

図右の表部分には、node エージェントの保持しているデータが表示されている。

IDNo は、整数型変数であり、個々の node エージェントをプログラム内で識別するための情報として使っている。

⑤問題は、個々のノードが連結する相手の情報を初期値データとしてどのように用意するか、またノードの連結関係を各ノードにどのように持たせるかであった。

電子回路の基本素子の場合、出力は１出力であるが、入力は多入力のものが多い。また素子からの出力は分岐結節点を経て分裂することもある。

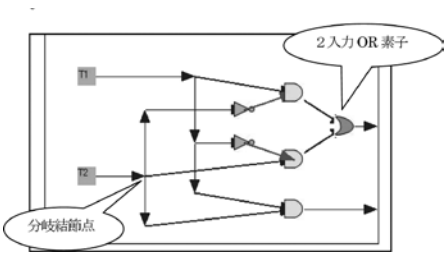
（図７.）は、半加算回路（halfadder）の様子であるが、吹き出しで説明してあるように、入力・出力が１つではないノードがいくつか存在することがわかる。

node エージェントにエージェント集合型変数 link を持たせ、これに個々の node エージェントが次に連結している node エージェントを保持させたいが、カンマ区切りの CSV テキストデータとして用意したデータを直接初期値データとして読み込ませることは不可能なので、工夫が必要であった。

node エージェントの連結データ（初期値データ）は、やはりエクセルシート上で作成し、テキストファイルとして保存したものを使いたい。

一旦 artisoc 上になんらかの方法でテキストファイルを読み込ませ、シミュレーションの動作直後にそのデータを基に、各 node エージェントのエージェント集合型変数 link に次に連結している node エージェントを書き込む処理をプログラムで実行すればよい。

（図６.）node エージェント初期設定画面



（図７.）半加算回路シミュレーション

すると次には、node エージェントとは別に、テキストファイルを読み込んで保持する実体を考える必要があるが、２方法ほど考えられる。

（１）グローバル変数（ノードに所属しない変数）として２次元配列変数数を１つ用意し、そこに全ての連結情報を入れる方法。

例えば、link（50,2）のようなグローバル配列変数数を用意すれば、50件の連結情報が保持できる。

例えば、link（0,0）＝－1、link（0,1）=1という情報が１組の連結情報で、IDNo が－１の node エージェントが、次に IDNo が１の node エージェントと連結しているという情報になる。

ただし、この方法だとカンマ区切りの CSV テキストデータとして情報を作成するのが多少困難である。artisoc の場合（図８.）のように１件の連結情報が２行に渡るので、手作業で作成するには不向きであろう。

（２）連結情報の数分、特別なエージェントを生成し、１つのエージェントに１件の連結情報を持たせる方法。

この場合には、単に初期値データを保持するために使われるだけのエージェントが多数生成されることになる。かなり冗長な話ではあるが、その代わり連結情報をカンマ区切りの CSV テキストデータとして用意するのが楽である。１組の連結データが１レコードのデータとなり、分かりやすい。

結局、今回はこの方式で作成してみた。（図９.）は、連結情報を保持する link エージェントに読み込ませるデータのエクセルシート上での様子である。

	A	B	C	D
1	Dim:1	Dim:2	link	
2	0	0	−1	
3	0	1	1	
4	1	0	−2	
5	1	1	5	
6	2	0	1	
7	2	1	2	
8	3	0	2	
9	3	1	4	

（図８.）連結情報 CSV ファイル1

	A	B	C	D	E	F	G	H
1	ID	X	Y	Layer	Direction	FromNode	ToNode	
2	0	0	0	0	0	−1	274	
3	1	0	0	0	0	−2	275	
4	2	0	0	0	0	−3	276	
5	3	0	0	0	0	−4	277	
6	4	0	0	0	0	−5	278	
7	5	0	0	0	0	−6	279	
8	6	0	0	0	0	−7	280	
9	7	0	0	0	0	−8	281	
10	8	0	0	0	0	−9	282	
11	9	0	0	0	0	−10	283	
12	10	0	0	0	0	−11	284	

（図９.）連結情報 CSV ファイル2

FromNode が連結元ノードの IDNo、ToNode が連結先ノードの IDNo である。

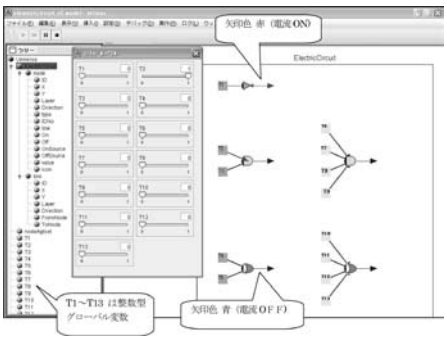
例えば２行目の例だと、IDNo が－１の node エージェントが、IDNo が274の node エージェントと連結していることがすぐ分かる。

２.２　基本素子シミュレーション

（図10.）は、NOT 回路、２入力 AND 回路、４入力 AND 回路、２入力 OR 回路、４入力 OR 回路の５つを単に画面上に配置した単純なシミュレーションである。

回路を表示するフィールドには、Electric Circuit という名前をつけている。左下隅の座標が (0, 0)、右上隅の座標が (550, 600) のフィールドである。

総 node エージェント数36個、総 link エージェント（node エージェントの連結情報を保持するエージェント）数31個のシミュレーションである。



（図10.）基本素子シミュレーション画面

四角い画像のノードとして配置されているもの（T1 ～ T13）は、回路への入力の起点となるノードで、それぞれコントロールパネルのT1 ～ T13のスライダーと連動し、スライダーの操作により電流をON／OFFできるようになっている。図左下の吹き出しで説明されているT1 ～ T13はノードに属さない整数型のグローバル変数で、この変数値がコントロールパネルと連動し変化する。そして、そのグローバル変数値をT1 ～ T13の四角い画像として表示されるnodeオブジェクトの整数型value変数と連動させている。

コントロールパネルのT1 ～ T13のスライダーが0の場合は、それに連動するノードへの電流はOFF、1の場合は、それに連動するノードへの電流はONとなる。

白黒印刷では分かりにくいが、図の吹き出しで説明の通り、電流がONの箇所（矢印）は赤色である。（白黒印刷では少し薄い灰色）電流がOFFの箇所（矢印）は青色である。（白黒印刷では少し濃い灰色）

以下、２．３節以降のシミュレーションも同じ機能を有する。

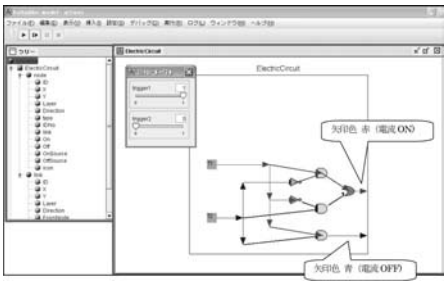
２．３　半加算回路（half adder）シミュレーション

（図11.）は、桁上がりのない２進数１桁のみの足し算を実行する半加算回路シミュレーションである。

T1の電流がON、T2の電流がOFF。出力は２つで、図中の吹き出しで示されているように上方出力の電流がON、下方出力の電流がOFFの状況図である。

このシミュレーションも、回路を表示するフィールド名はElectricCircuitで、左下隅の座標が（0，0）、右上隅の座標が（200，200）のフィールド上に構築した。

総nodeエージェント数18個、総linkエージェント数20個のシミュレーションである。



（図11.）半加算回路シミュレーション画面

２．４　全加算回路（full adder）シミュレーション

（図12.）は、桁上がりを考慮した２進数１桁の足し算を実行する全加算回路シミュレーションである。

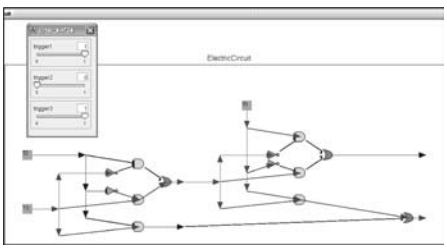
T1（下位桁からの桁上がりに相当する入力）の電流がON、T2の電流がOFF、T3の電流がON。出力は2つで、上方出力の電流がOFF、下方出力（上位桁への桁上がりに相当する）の電流がONの状況図である。

このシミュレーションも、回路を表示するフィールド名はElectricCircuitで、左下隅の座標が（0，0）で右上隅の座標が（500，200）のフィールド上に構築した。

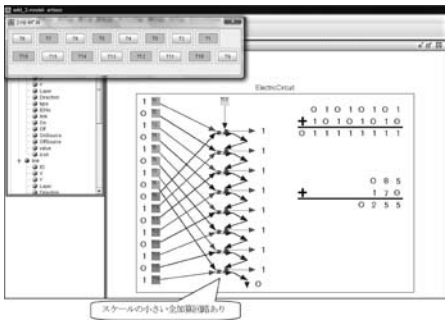
総nodeエージェント数35個、総linkエージェント数41個のシミュレーションである。

２．５　８ビット加算回路シミュレーション

（図13.）は、８ビットの２つの２進数の足し算を実行するシミュレーションである。



（図12.）全加算回路シミュレーション画面



（図13.）８ビット加算回路シミュレーション画面

２．４節までのシミュレーションと基本的な機能は同じである。

図の吹き出しで説明してあるように、２．４節で説明した全加算回路８個を、座標のスケール10分の１ほどに縮めて配置してある。

２．４節までの回路との違いは、以下の（１）～（５）である。

（１）８つの全加算回路の座標スケールを縮めて配置しているので、NOT・OR・AND 素子画像表示をやめた。矢印のみの表示である。

（２）コントロールパネル上のトグルボタンで、トリガーノードの電流 ON ／ OFF を切り替えている。

コントロールパネルの T1 ～ T16が銘記されているボタンがそれである。濃い灰色で表示されるボタンは、押されている状態（電流 ON）、薄い灰色で表示されるボタンは、押されていない状態（電流 OFF）である。

（３）node オブジェクトの値（整数型 value 変数）と連動して画像を切り換えるタイプのノードを追加。

矢印の色（赤／青）で電流 ON ／ OFF を判別するのでは分かりにくいこと、電子回路の説明に加え２進数の説明、さらには２進数の足し算の説明にも使えるツールを目指し、いくつかのノードの value 変数の値（０／１）に連動して、画像（０と表示する画像／１と表示する画像）表示を切り換えるノードを新たに追加した。

入力側トリガーの左、出力矢印の右、さらに画面右上に２進数の足し算を筆算風に表示している０／１画像がそれである。一見むずかしそうに見えるが、単にノードの value 変数値と連動して画像を切り換えるタイプのノードがそれらの座標位置にあるだけのことである。

（４）変化しない、単に固定画像を表示するだけのタイプのノードを追加。

画面右の足し算の記号（＋）、筆算風に足し算表示させるための横線は、単なるペイントブラシで作成した白黒画像である。画像がそれらの座標位置にあるだけのことである。

（５）特別な計算により10進数表示のための０～９の数字画像を表示するノードの追加。

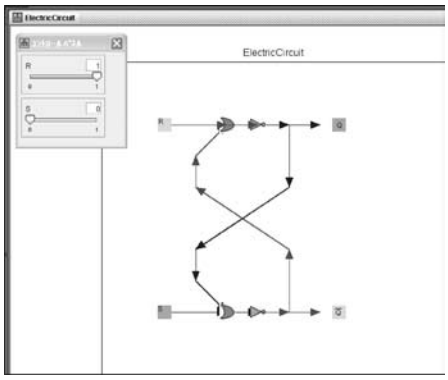
２進数の説明に加えて、２進数と10進数の比較説明にも使えるツールを目指して追加した機能である。

ノードのタイプとしては、（３）と同じであるが、この部分は少し泥臭いプログラムを加え、２つの入力側８桁２進数と出力側９桁２進数を10進数に変換した結果によりノードの value 変数の値（０～９）を更新し、その値に連動した０～９画像を表示させ、10進数の足し算を筆算風に表示している。

（図14.）は、ノード表示用に作成した画像の例である。

このシミュレーションも、回路を表示する

（図14.）ノード表示用画像例



（図15.）RS フリップフロップ回路シミュレーション画面

フィールド名は ElectricCircuit で、左下隅の座標が（0, 0)、右上隅の座標が（700, 500）のフィールド上に構築した。

総 node エージェント数371個、総 link エージェント数412個のシミュレーションである。

２．６　RSフリップフロップ回路シミュレーション

（図15.）は順序回路の１つ、RS フリップフロップ回路のシミュレーションである。

これも、２．２～２．４節までのシミュレーションと同じである。

このシミュレーションも、回路を表示するフィールド名は ElectricCircuit で、左下隅の座標が（0, 0)、右上隅の座標が（550, 500）のフィールド上に構築した。

総 node エージェント数20個、総 link エージェント数18個のシミュレーションである。

２．７　８ビットメモリ回路シミュレーション

２．６節の RS フリップフロップ回路を８個使えば、８ビットの情報を記憶するメモリー回路ができるはずである。実用的なメモリー回路はもっと複雑であろうし、適当な回路図資料も無かったので、勝手に８ビット情報を記憶できそうな回路を構成してみた。単なる説明用の仮想メモリーシミュレーション回路である。

これも、２．５節の８ビット加算回路と機能はほぼ同じであるが、加算回路ほど複雑では無かったので、NOT・OR・AND 基本素子画像を表示させるスペース上の余裕があった。一見２．５節の８ビット加算回路よりも複雑そうに見えるが、それは基本素子画像が表示されているからである。基本素子画像が表示されていると、見た目は電子回路らしく立派に見える。

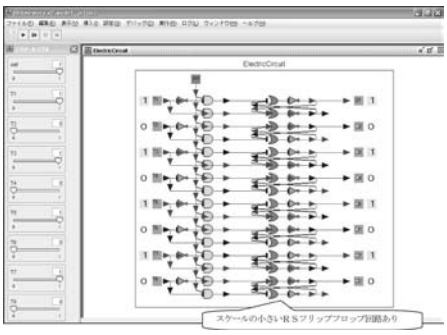
このシミュレーションも、回路を表示するフィールド名は ElectricCircuit で、左下隅の座標が（0, 0)、右上隅の座標が（600, 550）のサイズのフィールド上に構築した。

総 node エージェント数226個、総 link エージェント数241個のシミュレーションである。

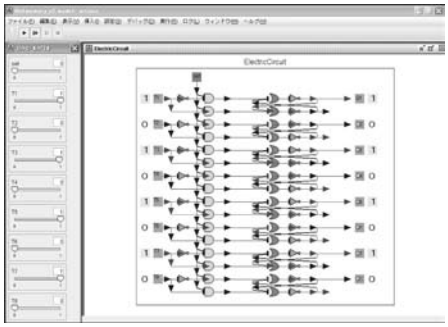
コントロールパネルのスライダーで、set、T1 ～ T8のトリガー node エージェントの電流 ON ／ OFF を切り換える。

（１）（図16.）は、set トリガー node エージェントの電流が ON の状態である。この状態で T1 ～ T8トリガー node エージェントの電流 ON ／ OFF を切り換えると、T1 ～ T8トリガー node と同じ状態が出力側の Q1 ～ Q8 node エージェントに再現される。

（２）次に、（図17.）のようにコントロールパネルのスライダーで、set トリガー node エージェントの電流を OFF にすると、出力側の Q1 ～ Q8 node エージェントの状態が固定されることになる。つまり、T1 ～ T8トリガー



（図16.）8ビットメモリ回路シミュレーション画面

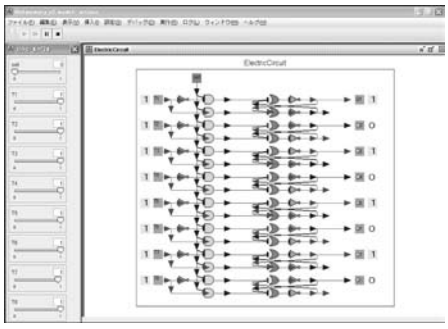


（図17.）8ビットメモリ回路シミュレーション画面2

node エージェントの状態が記憶（メモリー）されたことになる。

（3）次に、T1 ～ T8のトリガー node エージェントの電流 ON ／ OFF を変化させても、Q1 ～ Q8 node エージェントの状態は変化しない。つまり、set エージェントの電流が OFF の状態では、T1 ～ T8のトリガー node エージェントの以前の電流 ON ／ OFF 状態が確かに保持されている（メモリーされている）ことが確認できる。

仮に、学生各自が授業で使用する各パソコンで artisoc が使える環境になっていれば、一種の遊び的な演習であろうが、「各自のパソコンでこのシミュレーションを使い、“10010101”という情報が完全に記憶された状態にせよ。そしてその状態を PrintScreen でコピーし、パワーポイントのシート上に画像として貼り付けて保存し提出せよ。」というような課題も可能である。



（図18.）8ビットメモリ回路シミュレーション画面3

（ただしそれには、学生各自に『人工社会構築指南 artisoc によるマルチエージェント・シミュレーション入門』[1] を購入してセットアップして使わせるか、artisoc academic を購入し、演習室のパソコン全てに artisoc がセットアップされて使える状態になっている必要がある。）

3．電子回路のエージェント

（1）node エージェント

エージェントを定義すると、自動的に ID、X、Y、Direction、icon などの変数が用意される。特にシミュレーションでは使用していない変数は、（未使用）と記述する。（表1．参照）

X、Y、type、IDNo、icon 変数値は、エクセルシート上で作成し、カンマ区切りの CSV テキストファイル化したものを初期値として読み込ませている。

（2）link エージェント

node エージェントの連結情報を保持するための使われるエージェント。エージェントとしての表示・動作は無い。（表2．参照）

FromNode、ToNode 変数値は、エクセルシート上で作成し、カンマ区切りの CSV テキストファイル化したものを初期値として読み込ませている。

4．電子回路のスクリプト

4．1　スクリプトの構成

artisoc のスクリプトは、独立した5種類のセクションに分かれている。（図19．参照）

artisoc のシミュレーションを実行すると、まず Univ_Init セクションに書かれたスクリプトが実行され、その後ステップ処理が繰り返される。

変数名	型	用途
ID	整数型	aritisoc エンジンが自動的に各エージェントに設定する値（未使用）
X	実数型	エージェントの X 座標値
Y	実数型	エージェントの Y 座標値
Layer	整数型	ノードを表示するレイアー（未使用）
Direction	実数型	エージェントの向いている方向（未使用）
type	整数型	node エージェントのタイプ ０：結節点ノード 配線が曲がる点、配線が分岐する点などに利用。 特に表示はしない。見えない点。 １：NOT 回路素子 入力信号 ON ／ OFF 値を反転して、次に連結するノードに伝える。 ２：AND 回路素子 全ての入力信号が ON の時に、次に連結するノードへの信号を ON とする動作を行う。 ３：OR 回路素子 入力信号値が１つでも ON であれば、次に連結するノードへの信号を ON とする動作を行う。 －１：トリガーノード コントロールパネルのスライダーまたはトグルボタンと連動して value 値を変動させるノード。 シミュレーションへの入力信号となるノード。 IDNo 値＋“.jpg”名称の画像を表示する。 －２：Value 値表示ノード value 値に従って、対応する数値画像を表示する。 －３：画像表示ノード icon 変数に格納されている名称の画像を常に表示するのみ。
IDNo	整数型	プログラム上ノードを識別するための数値
link	エージェント集合型	node エージェントが連結している node エージェントを格納する
On	エージェント集合型	自分が起点で赤い矢印を引く node エージェントを格納する変数
Off	エージェント集合型	自分が起点で青い矢印を引く node エージェントを格納する変数
OnSource	エージェント集合型	自分に電流 ON 信号を送ってくる node エージェントを格納する変数（冗長ではあるが、プログラムを簡単にするために用意）
OffSource	エージェント集合型	自分に電流 OFF 信号を送ってくる node エージェントを格納する変数。（冗長ではあるが、プログラムを簡単にするために用意）
value	整数型	自分の状態を格納する変数、または次のノードに伝えるべき ON ／ OFF 信号を格納している変数 ０：電流 OFF、１：電流 ON、または２～９の整数値
icon	文字列型	表示画像名称（この変数に画像名称が格納されると artisoc が自動的に node エージェント位置に画像表示を行う）

（表１.）node エージェント

変数名	型	用途
ID	整数型	aritisoc エンジンが自動的に各エージェントに設定する値（未使用）
X	実数型	エージェントの X 座標値（未使用）
Y	実数型	エージェントの Y 座標値（未使用）
Layer	整数型	ノードを表示するレイアー（未使用）
Direction	実数型	エージェントの向いている方向（未使用）
FromNode	整数型	連結元 node エージェントの IDNo を保持する
ToNode	整数型	連結先 node エージェントの IDNo を保持する

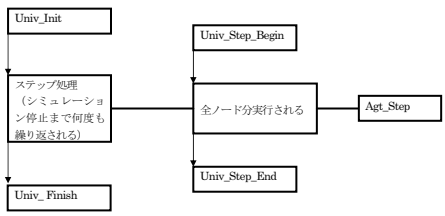
（表２.）link エージェント

各ステップ処理においては、まず Univ_Step_Begin セクションのスクリプトが実行された後、各ノードの振舞いが記述された Agt_Step セクションのスクリプトが全ノード分実行される。そしてステップ処理の最後に Univ_Step_End セクションのスクリプトが実行される。

シミュレーションの最後には Univ_Finish セクションのスクリプトが実行される。

（図19.）がスクリプトの構成概略図である。上から下に向かう順序で実行されるが、四角の要素の右にある処理部分は、繰り返し実行される部分という意味で図示してある。

今回の電子回路シミュレーションで、スクリプトを書き込んだセクションは、Univ_Init、Univ_Step_Begin、node エージェントの Agt_Int および Agt_Step セクションの４箇所である。link エージェントは初期値データを保持する用途で使用しただけなので Agt_Int に記述はない。また、これらセクションに書き込んだスクリプトとは別に３つのサブルーチンを独立したテキストファイルとして作成して使用した。



（図19.）スクリプトの構成外略図

４.２　Univ_Init セクション

全ての link エージェントを読み込み、node エージェントの連結関係データ（FromNode、ToNode ペア）を取り出し、連結元 node エージェントのエージェント集合型変数 link に連結先 node エージェントを追加する。

４.３　Univ_Step_Begin セクション

トリガーノードタイプの node エージェントの value 変数に、対応するグローバル変数の値を代入する。グローバル変数の値は、コントロールパネルのスライダーまたはトグルボタンで変えることができる。

例えば（表４.）３行目の場合は、IDNo = −9の node エージェントの value 変数に、グローバル変数 set の値を代入していることになる。

この箇所は、シミュレーションによって変わる箇所である。(使用するトリガーが変わるため)

４.４　Agt_Int および Agt_Step セクション

Agt_Init では、NOT・AND・OR・トリガータイプ node エージェントの場合に固定の画像を座標位置に表示させている。

Agt_Step では、value 変数値によって表示画像を切り換えるタイプの node エージェントでの画像切り替え、結節点・NOT・AND・OR タイプの node エージェントでの、入力値に応

```
include "SearchAgt.inc"
include "SetTrigger.inc"
include "SetOnOff.inc"

Univ_Init{
Dim link As Agtset
Dim one As Agt
Dim FromAgt As Agt
Dim ToAgt As Agt

MakeAgtset（link, Universe.ElectricCircuit.link）　//　link エージェントのみ集める
For each one in link
    if one.FromNode <> 0 then
        SearchAgt（one.FromNode）　//　リンク元ノードを検索する
        if CountAgtset（Universe.nodeAgtset）> 0 then
            FromAgt= GetAgt（Universe.nodeAgtset, 0）

            SearchAgt（one.ToNode）　//　リンク先ノードを検索する
             if CountAgtset（Universe.nodeAgtset）> 0 then
                ToAgt= GetAgt（Universe.nodeAgtset, 0）
                AddAgt（FromAgt.link, ToAgt）　//　リンク元ノードの link 変数にリンク先ノードを追加する
            end if
        end if
    end if
next one
}
```

（表３.）Univ_Init セクション

```
Univ_Step_Begin{

setTrigger（- 9, Universe.set）
setTrigger（- 1, Universe.T1）
      ・
      ・
}
```

（表４.）Univ_Step_Begin セクション

```
include "SearchAgt.inc"
include "SetTrigger.inc"
include "SetOnOff.inc"

Agt_Init{
if My.type == 1 then
    My.icon = "NOT.jpg"
elseif My.type == 2 then
    My.icon = "AND.jpg"
elseif My.type == 3 then
    My.icon = "OR.jpg"
elseif My.type == - 1 then　//　トリガーの場合
    My.icon= Cstr（My.IDNo）& ".JPG"
end if
}

Agt_Step{
Dim one As Agt
Dim dest As Agtset
Dim total As Integer
Dim On As Integer
Dim Off As Integer
Dim i As Integer
```

```
Dim num As Integer

if My.type == - 1 then    //   トリガーノードは何もせず

elseif My.type == - 2     then //   Value ノードは value に基いて表示する
    My.icon= Cstr（My.value）& ".JPG"
else

    On = CountAgtset（My.OnSource）
    Off = CountAgtset（My.OffSource）
    total= On + Off

    For each one in My.link
        if My.type == 0 then    //    結節点ノードの場合
            if On > 0 then
                SetOnoff（1 , My, one）
            else
                SetOnoff（0 , My, one）
            end if

        elseif My.type == 1 then    //   NOT 回路の場合
            if On > 0 then
                SetOnoff（0 , My, one）
            else
                SetOnoff（1 , My, one）
            end if

        elseif My.type == 2   then //   AND 回路の場合
            if Off > 0 then
                SetOnoff（0 , My, one）
            else
                SetOnoff（1 , My, one）
            end if

        elseif My.type == 3 then   //   OR 回路の場合
            if On > 0 then
                SetOnoff（1 , My, one）
            else
                SetOnoff（0 , My, one）
            end if

        end if
     next one

 end if
}
```

（表5.）Agt_Int および Agt_Step セクション

じて、連結する node エージェントへの電流 ON ／ OFF 信号の伝達および、電流 ON ／ OFF に対応して矢印色を赤色／青色に切り換える処理を行っている。

4.5　サブルーチン

（1）サブルーチン SearchAgt

同名の SearchAgt.txt テキストファイルとして保存して使用。

インクルード宣言（include "SearchAgt.inc"）により artisoc により自動的に読み込まれて使われる。

IDNo をキーに全 node エージェントから目的の node エージェントを探す。

（2）サブルーチン SetOnOff

同名の SetOnOff.txt テキストファイルとし

```
1  //   IDNo を持つエージェントを全エージェントから検索する
2  Sub SearchAgt（IDNo As Integer）{
3
4  Dim allAgent As Agtset
5  Dim dest As Agtset
6  dim one As Agt
7
8  ClearAgtset（Universe.nodeAgtset）
9
10 MakeAgtset（allAgent, Universe.ElectricCircuit.node）
11
12 for each one in allAgent
13    if one.IDNo == IDNo then
14       AddAgt（Universe.nodeAgtset,  one）
15       Break    // もう調べる必要なし（エージェントが見つかった）
16    end if
17 next one
18 }
```

（表６.）サブルーチン SearchAgt

```
1  //   連結するエージェントへの電流を On ／ Off にする
2  Sub SetOnOff（OnOff As Integer, Myself as Agt, one As Agt）{
3
4  if OnOff ==  0  then    //   電流を Off にする
5     if one.type == - 2  then    //   2進数表示ノードの場合
6        one.value= 0
7     else
8        AddAgt（one.OffSource, Myself）
9        RemoveAgt（one.OnSource, Myself）
10       AddAgt（Myself.Off, one）
11       RemoveAgt（Myself.On, one）
12    end if
13
14 else                        //   電流を On にする
15    if one.type == - 2  then    //   2進数値表示ノードの場合
16       one.value= 1
17    else
18       AddAgt（one.OnSource, Myself）
19       RemoveAgt（one.OffSource, Myself）
20       AddAgt（Myself.On, one）
21       RemoveAgt（Myself.Off, one）
22    end if
23 end if
24 }
```

（表７.）サブルーチン SetOnOff

て保存して使用。

インクルード宣言（include " SetOnOff.inc"）により artisoc により自動的に読み込まれて使われる。

連結する node エージェントへの電流 ON ／ OFF 信号の伝達および、電流 ON ／ OFF に対応して矢印色を赤色／青色に切り換える処理を行っている。node エージェントの On・Off・OnSource・OffSource 変数を操作している。

２進数表示ノードの場合には、単に value 変数値を更新するだけである。

（３）サブルーチン SetTrigger

同名の SetTrigger .txt テキストファイルとして保存して使用。

インクルード宣言（include " SetTrigger.inc"）により artisoc により自動的に読み込まれて使われる。

トリガーノードタイプの node エージェント

```
1   //    トリガーエージェントの電流を On ／ Off する
2   Sub SetTrigger（IDNo As Integer, OnOff As Integer）{
3
4   Dim trigger As Agt
5   Dim one As Agt
6   DIm linkNode As Agtset
7
8   SearchAgt（IDNo）
9
10  if CountAgtset（Universe.nodeAgtset）＞ 0 then
11
12      trigger= GetAgt（Universe.nodeAgtset, 0）
13      linkNode= trigger.link
14
15      For each one in linkNode
16
17          if OnOff == 0 then
18              AddAgt（one.OffSource, trigger）
19              RemoveAgt（one.OnSource, trigger）
20          else
21              AddAgt（one.OnSource, trigger）
22              RemoveAgt（one.OffSource, trigger）
23          end if
24      next one
25  end if
26  }
```

（表８.）サブルーチン SetTrigger

の value 変数に、対応するグローバル変数の値を代入する。

以上が今回作成した電子回路シミュレーションのスクリプトの概要である。２.５節に記述した８ビット加算回路シミュレーションの場合には、10進数の足し算を筆算風に表示させる機能を実現させるためのスクリプトを追加してある。

５．結び

今回作成した電子回路シミュレーションは、ルールの総計が200行までという制限のある artisoc textbook で十分作成可能であった。電子回路を構成する基本素子 NOT・AND・OR 素子を実現するためのエージェントのルール自体簡単なので当然であろう。

エクセルシート上で、node エージェントデータと link エージェントデータを作成し、artisoc に読み込ませるだけで、また新しい電子回路シミュレーションを作ることができる。

今回作成した、半加算回路、全加算回路などは動作（真理値表）が知られているものであるが、初期値データとして読み込ませる node エージェントデータと link エージェントデータ次第で、動作が未知の電子回路シミュレーションを作成することができる。

「ある回路図を元に真理値表をもとめよ。」という演習は、参考にした情報処理の教科書[2] にもある。基本素子５つくらいが組み合わされた回路であるが、シミュレーションを使えば、かなり複雑な回路でも可能である。

学生各自が授業で使用するパソコンで artisoc が使える環境になっていれば、そのシミュレーションを自分で動かして、真理値表を作成するという演習も可能であろう。

『人工社会構築指南 artisoc によるマルチエージェント・シミュレーション入門』[1] に掲載されていたシミュレーション例は全て、シミュレーション実行中にノード同士の連結関係をつけていくものであった。

キストファイルデータとして事前に用意した表示座標データを含むエージェント（ノード）の初期情報を、シミュレーション実行前に読み込みことにより適切な座標位置にノードを配置してシミュレーションを開始することはできたが、問題はノード同士の連結関係の情報である。

電子回路シミュレーションを実現するには、ノードの座標データだけでは不十分で、シミュレーション開始時にノード同士の連結関係も決まっている必要がある。

ノード連結関係データの扱いとして、今回はかなり冗長ではあるが、エクセルシート上で簡単にデータが準備できるやり方を取った。連結元ノードIDと連結先ノードIDの１組のデータペアで１つの連結関係を表現するのは常識的な方法で、かつデータ構築作業もわかりやすいものであろう。

データ構築作業は簡単だが、そのデータを読み込ませるために、連結データに１対１で対応するlinkエージェントをartisoc上に生成し、シミュレーションのスタート時（Univ_Initセクション）にてlinkエージェントから情報を取り出し、それを元に最終的にnodeエージェントに連結先のノードを接続するというかなり冗長な方法を取った。

冗長なやり方ではあろうが、実際に作成してみると多くのコーディング量が必要なわけでもなく、簡単に実現できた。仮に多数発生させたlinkエージェントによってシミュレーションの実行速度が低下するようならUniv_Initセクションで使い終わった後にlinkエージェントを全て抹消すれば良いであろう。

今回シミュレーションの初期状態からノードの連結関係が決められているタイプのシミュレーションにトライすることにより、今後のartisocの利用へのアイデアがいろいろ浮かんできた。

今後の方向性としては、以下の（１）、（２）が考えられる。

（１）初等情報教育用のネットワーク（インターネット）の説明ツールとして何か作成できるのではないか。

（２）ノード同士が連結関係にある系について、複雑ネットワークの科学等の研究成果を実際に学生にシミュレーションして見せるツールを作れないか。

その場合、今回の逆の手法が使えないであろうか。つまり、あるシミュレーションを実行することによって生み出されたノードとノード同士の連結関係をテキストファイルとして書き出し、そのデータを基に別のシミュレーションを実行するという手法である。

今回の逆で、シミュレーション停止時に実行されるUniv_Finishセクションにて、ノードの連結関係を（連結元ノードID－連結先ノードID）の単純なペア情報として書き出すスクリプトを入れて置けば、そのシミュレーションが生成したデータを基に別のシミュレーションが実行できるという理屈である。

artisocは、最初に読み込むノードの初期値データさえ利用できれば、いろいろなシミュレーションが容易に作成できそうなツールであると実感した次第である。

[参考文献]

1）『人工社会構築指南 artisocによるマルチエージェント・シミュレーション入門』山影進著、2007年有限会社書籍工房早山

2）『コンピュータサイエンスとリテラシー』楊国林／篠政行／市瀬紀彦著、2005年弓箭書院

3）『情報学入門　－大学で学ぶ情報科学・情報活用・情報社会－』大内東・岡部成玄・栗原正仁編著、2006年株式会社コロナ社