MMS 24-1
(1976. 4. 23)

オンライン・シミュレータ GMSS

久保正敏・伊藤 潔・田畑孝一・大野 豊
京都大学工学部

1. はじめに

GMSS (Graphical Modeling and Simulation System) は，トップダウン的にモデリングおよびシミュレーション実行が可能なオンライン・シミュレーションシステムである。GMSSはtransactionタイプの汎用シミュレーションシステムであり，モデリングの段階および実行過程において，グラフィックスを用いた対話機能が重視されている。GMSSでは，モデリングのための言語として，text形式の言語の他に，これによく対応する図形シンボル形式の言語を用意しており，これを我々は，"pictography"と呼んでいる。この図形シンボルは，モデリングの段階のみならず，実行過程のモニタに際しても用いられ，両段階における対話機能に有効な役割を果たしている。このような対話機能の充実によって，シミュレーションにおける重要な観点であるモデルを正しく作成すること及び，モデルの正しさの検証が容易になるものと考えられる。

GMSS言語については，すでに文献[1]および本研究会[2]で発表したので，ここでは特にGMSSにおけるモデリングとシミュレーション実行の段階での対話性と，それに応ずるシミュレータの構造について述べる。

2. トップダウンモデリングとオンラインシミュレーション

一般に，シミュレーションのモデル形成から実行に至る過程は，次の4つの段階に分けられるだろう。

(1) シミュレーションの対象となるシステムの振舞を解析し，シミュレーションモデルの構想を立てる。
(2) そのモデル構想により，モデルプログラムを作成する。
(3) モデルプログラムのシミュレーションを実行する。
(4) シミュレーションの結果を解析する。

ユーザは，(1)の段階で，対象システムの振舞のうちから関心がある振舞を抽出し，これをシミュレートするために必要十分なモデルを組立てるが，組立てたモデルの振舞が対象システムのそれに合致しているかどうかは，シミュレーション実行を待たないとわからない場合が多く，モデルが予定していた振舞をしないと判断された場合には，モデルを再構成する必要が生じる。従ってシミュレーションにおいては，いかに目的とする振舞を行なう正しいモデルを作成するかが，重要な課題であるといえる。すなわち，シミュレーションシステムは，より容易に，より早く 1) ユーザの意図するモデルを正しく作成する手段 及び 2) そのモデルの振舞を確かめる手段 を備えている必要がある。我々のGMSSは，この2点の要請を満足するために，トップダウンモデリング及びオンラインシミュレーションの両概念を導入した。

トップダウンモデリング技術は，モデル作成段階でユーザの意図に沿って正しくモデルを作成することを容易とするために導入された。GMSSにおけるトップダウンモデリングは，モデルプログラムのコーディングを，トップダウン的に行なうものである。すなわち，トップダウン的にコーディングされつつあるモデルプロ

グラムは，現在コーディングが完了したレベルで実行され，テストされる。さらに，モデルプログラムは構造化プログラミング技法にのっとりGO TO文を含まず，連接，選択，繰り返しの3つの型から組みたてられる構造となっている。このようにして作成されたモデルは，モデリング段階のいかなる時点でも見やすく，誤りの発見が非常に容易となることが期待できる。

一方，オンラインシミュレーションの概念は，できるだけ容易にシミュレーションモデルの正しさを確かめ，それを修正することを可能とするために導入された。モデルが見かけ上いかに正しく作成されたとしても，そのモデルが本当に正しいかどうかはシミュレーション実行時における振舞を調べてみないとわからない。すなわち，最初作成されたモデルについてシミュレーションを実行させ，その進行過程を人間が監視し，必要に応じて進行を止め途中結果の検討，モデルの手直しが必要となり，このためにはモデル作成時のみならず実行時においても，人間と機械の対話が十分に行なえるオンラインシミュレーションが有効と考えられる。

このように，トップダウンモデリング及びオンラインシミュレーションの概念を導入することにより，GMSSは，より容易により早く正しいモデルを作成することを可能とするものと期待できる。

3．GMSS言語

3－1．Entity

GMSSはtransactionタイプのシミュレーションシステムであり，モデルは，そこに存在するentityに対して働きかけられるoperationの集合と考えられる。モデルの状態に，関連ある変化(event)を与える一連のoperation集合はactivityと呼ばれ，activityの集合としてモデルが記述される。GMSSでは，elementary entityとしてGPSSと同様に，facility，storage，queue，semaphoreなどを取りあげた他に，abstract entityとして，その振舞は下位のレベルで記述される"actentity"を導入した。また，actentityの一種として，"comact"も導入した。これらのentityは，次のようにまとめられる。

- GMSS entity
  - elementary entity
    - transaction
    - facility, storage, queue, semaphore
  - abstract entity
    - actentity
    - comact

3－2．GMSS statement

GMSS statementは，entityに働きかけられるoperationを表現するものであり，次に示すように3つのタイプに分けられる。

- GMSS statement
  - primitive statement …… USE, IN, HOLD, PUT, GET, etc.
  - structure statement
    - selection statement …… IF, CASE, etc.
    - split statement …… SPLIT
    - repetition statement …… REPEAT, DO
  - activity statement …… ACT, USE ACT

activity statementは，actentityの起動を表現するものであり，structure statementは，transactionの流れをコントロールするものである。

3－3．Actentity，comactとモデル構造

Actentityは，システムに存在するactivityを表現し，その振舞の詳細は，モデルの下位のレベルで［ＡＣＴＩＶＩＴＹ］として定義され，記述される。図１はactentityとそれに対応する［ＡＣＴＩＶＩＴＹ］との関係を表わし，図２は，そのtree表現である。

同一のactentityが，１つの［ＡＣＴＩＶＩＴＹ］で２回以上起動されたり，いくつかの［ＡＣＴＩＶＩＴＹ］から起動されたりする場合に，このactentityの［ＡＣＴＩＶＩＴＹ］としての定義の重複を避け，一度定義されれば，それを多重に使用することを可能とするために，comact（common actentity）の概念を導入した。comactは，これを起動するいくつかの［ＡＣＴＩＶＩＴＹ］のルートの［ＡＣＴＩＶＩＴＹ］において定義される。例えば，図１において，actentity A_1'とA_2が，同一のactentityである場合，これをcomact A_3とすることによって図１は図３にかきかえられる。図４は，そのtree表現である。

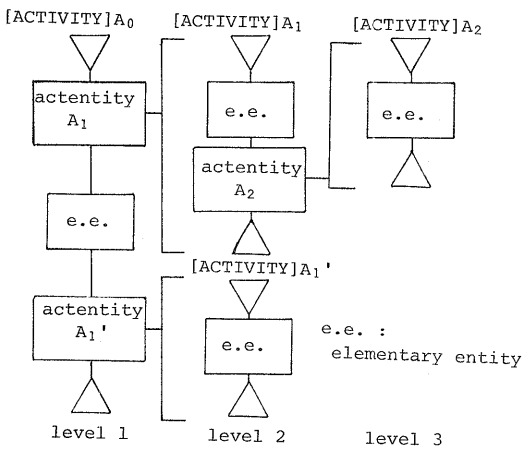

図１　[ACTIVITY]とactentityの関係

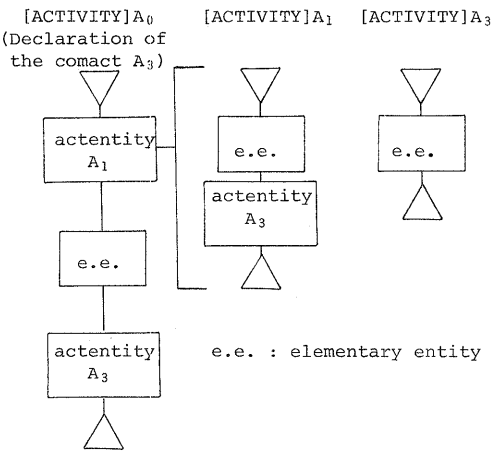

図３　Comactの例

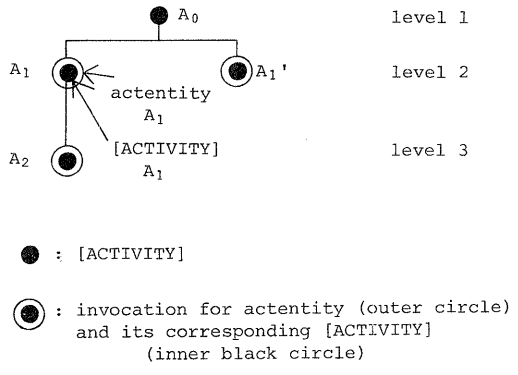

図２　図１のtree表現

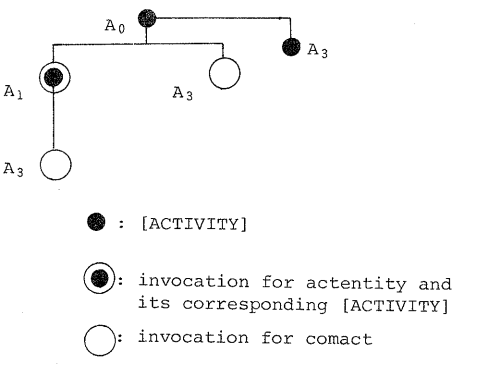

図４　図３のtree表現

3－4．Pictography

GMSSは，モデルの正しさをできるだけ容易に検証することを目ざして設計されたシミュレーションシステムである。従って，モデルを如何に表現するかが重要な問題となる。さらにオンラインシミュレーションを実現するためにも，人間とシミュレーションシステムとの間の有効な対話手段を供給する必要があり，モデルを両者の接点の一つと考えれば，モデルの表現方法が重要となる。シミュレーションモデルを図形的なシンボルで表現することが可能となり，これを用いて実行過程をモニタすることが可能となれば，モデリング段階及び，実行モニタ過程における人間とシミュレーションシステムとの対話機能は，より向上すると考えられる。

GMSSではGMSS statementに一対一に対応する図形シンボルが用意され，"pictographic" symbolと呼ばれている（図５）。これを用いて表現されたモデルは

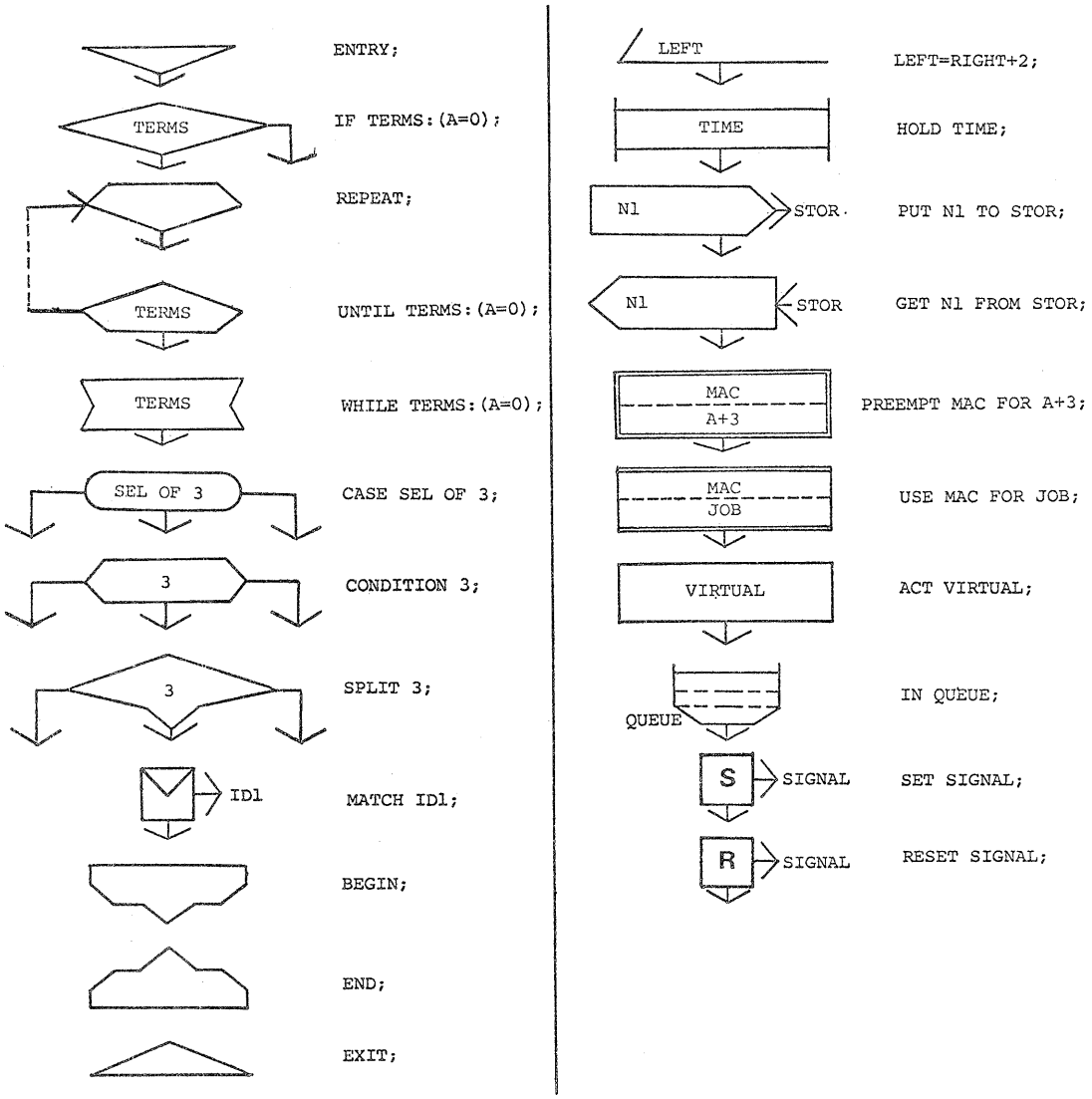

図5　GMSSのpictographic symbol

"pictograph" と呼ばれる。すなわちGMSSは2つのタイプのプログラミング言語 — textualな言語と pictographic symbol を用いる言語であり，後者は "pictography" と呼ばれる。GMSSのユーザは，この2者のいずれを用いてモデルを作成してもよいが，モデルの構造は，いずれの形式でもグラフィックディスプレイ上に同時に表示され，pictographを一見することにより，モデルの誤りを発見することは非常に容易となる。さらに，このpictographを用いて実行過程をモニタできる。

4. GMSSのシステム構成と制御の流れ

4－1. GMSSシステム構成

GMSSのシステム機器構成は，図6に示すように，ホストコンピュータをHITAC10IIとし，これにグラフィックディスプレイ及びこの制御のためのミニコンPDS-1D，ハードコピーをとるためのシリアルプリンタ，ファイルとしてディスクなどを接続したものである。

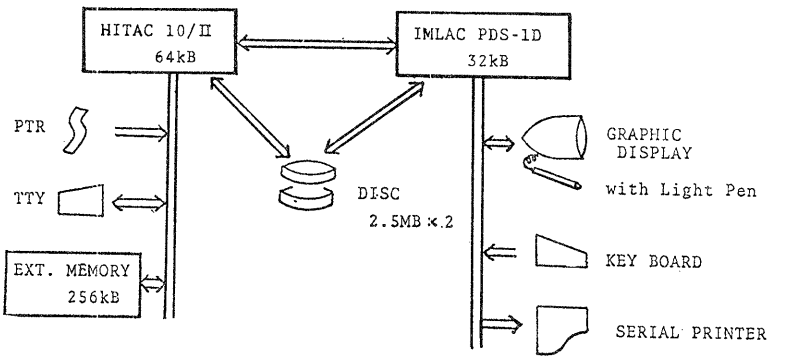

図6 GMSSのシステム機器構成

HITAC10IIには，GMSSモデルプログラムの translator，simulatorが常駐する。これらはFORTRANでインプリメントされている。PDS-1Dにはキイボード，グラフィックディスプレイが付属しており，各[ACTIVITY]を記述するための editor，[ACTIVITY]をリンクしてモデルプログラムを作成するための linker，実行時のモニタリングを制御する monitor などの他に，ユーザとの対話を実現するコマンド解析ルーチン，ディスプレイコントロールルーチンなどが常駐する。これらはアセンブリ言語でインプリメントされている。

4－2. システムの制御の流れ

GMSSシステムの制御の流れは図7に示すように

(a) ユーザによるモデルプログラム作成
(b) モデルプログラムのトランスレート
(c) ユーザによる実行コマンドの指示
(d) シミュレーション実行と結果の出力

の4段階に分かれている。(d)の段階でユーザはシミュレーションを実行し，同時にその過程をモニタすることができる。必要に応じて，実行の途中で(a)，(b)，(c)の段階に戻ることが可能である。

5. モデル作成段階における対話性

モデルを作成するための図7の(a)の段階は，[ACTIVITY]の定義と，[ACTIVITY]のリンクによるモデル完成の2つのフェーズに分けられる。最初のフェーズでは，[ACTIVITY]毎に定義が完了し，システムのファイルに登録される。この場合，

同じ名前を持つが，定義内容が少しずつ異なる[ACTIVITY]をいくつ定義，登録してもかまわない。これらの同じ名前を持つ[ACTIVITY]の"属"のうちから1つを選び出して他の[ACTIVITY]とリンクしてモデルを作成することにより，モデル内のある[ACTIVITY]の定義内容の差異によって生じるシミュレーション実行のバリエーションが，容易に実現できる。

[ACTIVITY]を新しく定義する場合を例にとり，対話機能を説明する。この場合のディスプレイ画面は，図8のような構成となっている。Pictograph display area には，モザイク式手法[3]を用いて[ACTIVITY]のpictographが表示され，text display area には対応するtextが表示される。ユーザはpictographic symbol，textのいずれを用いて[ACTIVITY]の記述を行なってもよく，記述方法をそれぞれpictographicモード，textモードと呼んでいる。

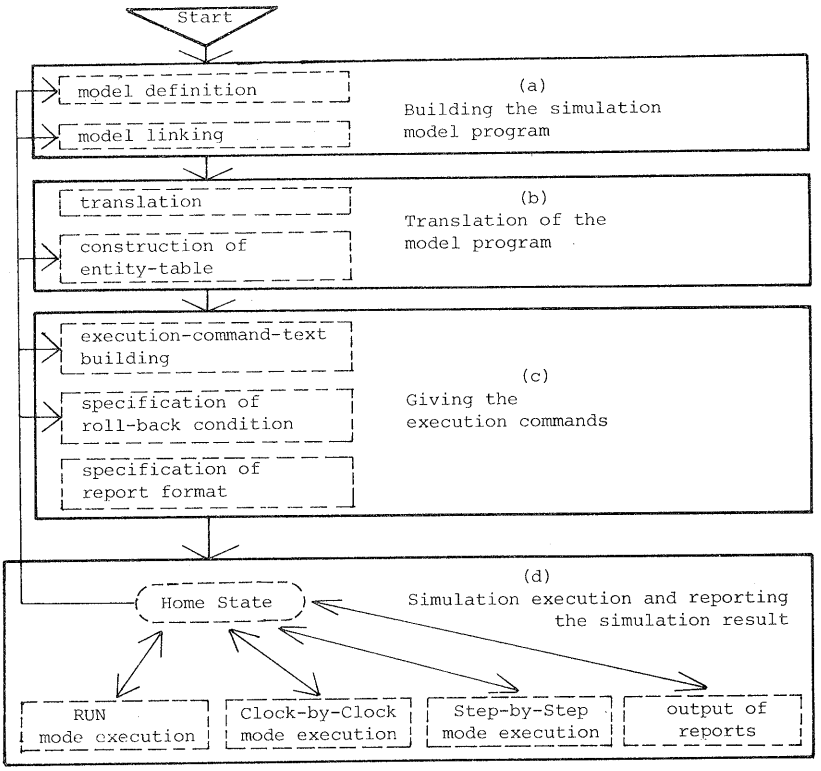

図7 GMSSの制御の流れ

Pictographicモードにおいては，まずmenu display area に表示されているpictographic symbolをライトペンでヒットする。次いでこのsymbolに対応するactivity statementに必要なパラメータをparameter display areaに打ちこみ，続いてpictograph display area上のworking box内にこのsymbolを挿入する。同時にtext display areaに対応するtextが自動的に挿入される。Pictographic symbolの削除も，working boxを用いて行なわれる。1つのpictographic symbolの挿入，削除のたびに，シンタックスチェックルーチンが起動され，現在作成中の[ACTIVITY]の構文，構造上の誤り

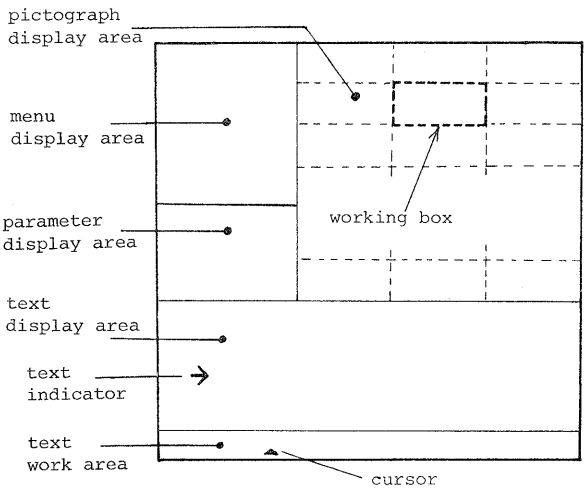

図8 モデル作成時の画面の構成

が可能な限りチェックされ，エラーがtextの頭に表示される。

Textモードにおいては，text work area 上に1行のstatementを打ちこみ，次いでこの1行をtext display areaのtext indicatorによって指示される位置に挿入するが，その効果は全くpictographicモードの場合と同じである。

このように，pictographによる見やすいモデル構造とシンタックスチェック機能により，モデル作成時に生じるエラーをインタラクティブにできるだけ少なくし，モデル作成段階の完結度を高めることができる。

6. 実行コマンドの指示

図7の(c)の段階は，ユーザがシミュレーション実行に関する指示を与える段階であり，execution-command-textの作成，roll-back 条件の指示，レポート仕様の指示，の3つのフェーズに分かれる。

Execution-command-textは，transactionの発生を指定するGENERATE文，消滅を指定するTERMINATE文，実行を終えるSTOP文，どの[ACTIVITY]を実行するかを指示するEXEC文，統計をとるためのTABULATE文，TABLEの定義などから成る。このように，実行コマンドとモデルを分離することにより，モデル部分を変更することなく，実行コマンドの組みあわせを変更するだけで種々のシミュレーションを行なうことができる。

Roll-backは，あらかじめ指定したroll-back条件を満足した時点のモデルの全状態をファイルに保存し，roll-backの指示によりその状態から実行を再開するものであり，時間逆行を可能としている。Roll-back条件は，1) execution-command-text中のTERMINATE文で消滅するtransactionの個数 2) シミュレーションクロックタイム 3) ある変数名とそのとるべき値，の3つのタイプで指定され，それぞれが満足された時点のモデル状態がファイルに保存される。

レポート仕様の指示は，変数の値，あるいはentityの統計量の時間を軸とする線グラフを得たい場合に与えるものであり，この指示がなかった統計量は，表の形で出力される。

いずれのフェーズの指示も，5.のtextモードと同様の方法で，グラフィックディスプレイを用いて行なわれる。

7. 実行時の対話性

GMSSでは，実行過程のオンラインモニタリングを重視しており，実行形態を3種用意して多様なユーザの要求に応えうるようになっていて，それぞれを次のように名づけている。

Step－by－Step モード
Clock－by－Clock モード
RUN モード

実行時にはこの3つのモードに従ってユーザの指示を受けつけるためのHome－Stateを置き，このHome Stateでは現在のモデルの状態を静止画面として表示し，ユーザによるモニタを許している。

7－1. Step-by-Step モード

この実行モードでは，transactionが1statementを実行するたびに実行を一時停止し，モデルの状態を画面に表示する。Simulatorは，ある[ACTIVITY]内で動きうるtransaction

の処理を終了してから他の[ACTIVITY]に制御を移し，モデル構造にみあってトップダウン的にレベルごとの実行を行なう。従ってこのモードでは，[ACTIVITY]単位に実行過程をモニタすることになり，ユーザはあらかじめモニタしたい[ACTIVITY]の名前を指示する必要がある。指示された[ACTIVITY]に制御が移れば，その構造と各種変数の現在値を画面に表示する。画面の構成は図9に示すようにモデル作成時とほぼ同じで，モデル構造とともに現在実行中のstatementに対する指標，そのstatementで使用する変数の現在値が表示されるので，実行過程を逐一ミクロ的にモニタでき，モデルの振舞の正しさを検証することが可能となる。

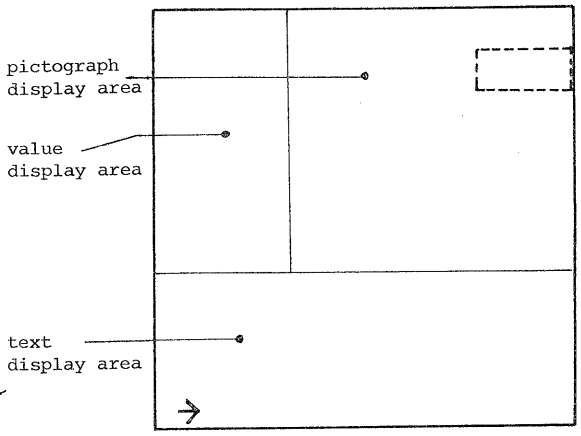

図9 Step-by-Step モード時の画面の構成

7-2. Clock-by-Clock モード

このモードでは，シミュレーションクロックが更新されるたびに実行を停止し，モデルの状態を画面に表示する。この場合の画面構成もStep-by-Stepモードとほぼ同じで，pictographic symbolの左方にそれが関係するentityの現在統計量が数値で表示される。このモードによって，モデルの振舞を統計量の観点からモニタし，モデルの正しさとともにモデルに含まれる各種の特徴的な振舞を検証できる。

7-3. RUN モード

このモードでスタートすると，simulatorはSTOP文によって指定された条件を満足するまで連続実行を行なう。ディスプレイ画面には，あらかじめ指定されたentityの現在統計量が時々刻々表示されるので，ユーザがこれを見て異常を発見すれば，割込みをかけて実行を中断することもできる。この場合にも制御はHome Stateに戻り，他のモードによる実行，roll-back，モデル作成段階への戻り等の指示が受けつけられる。

8. GMSS simulator

GMSS simulatorは，GMSS言語で記述されたモデルプログラムをインタプリティブ方式で実行するもので，図10に示すように，translator，interpreter，transaction scheduler，transaction generator/terminator，executerのモジュールから成っている。TranslatorはGMSSモデルプログラムをinternal-form-textに変換し，同時にentity table，transaction-scheduling-listのkernel1を作る。Transaction-scheduling-listの構造は図11に示すように，そのkernel1と，kernel elementに付属するblock chainあるいはready chainから成る。kernel1は各[ACTIVITY]ごとに作られ，kernel elementのチェインにより構成されている。kernel elementは，GMSS statementの順に，transactionがブロックされる可能性のあるstatementや，ready chainを持つstatementごとに設けられる。kernel elementは，それぞれステータスを持ち，それにつながるblock chain，ready chainにスキャンされるべきtransactionがあるかどうかを表示する。

Transaction schedulerは，GMSSモデルとは別に与えられるexecution-command-textの指示に従い，transactionを発生する。全体を管理するexecuterは，発生したtransactionをschedulerの管轄のもとにtransaction-scheduling-listに待ち状態として登録する。Schedulerは，transaction-scheduling-listをスキャンして実行待ちのtransactionのなかから1つを選択し，executerを介してinterpreterに渡す。Interpreterはこのtransactionに対応する実行指標ポインタに基づき，あるいはそれを更新しながらinternal-form-text上を進行する。それにつれて関連する種々のentityの属性値が変化してゆく。

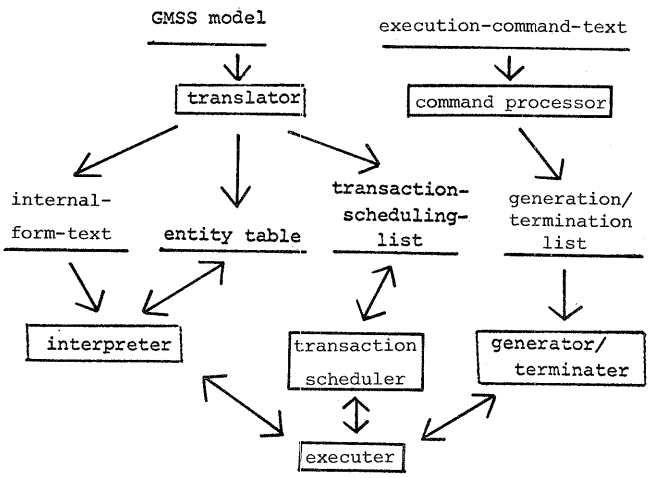

図 10　Simulatorの構成

Transactionがtext中のあるstatementでブロックされると進行をやめて，transaction-scheduling-listに，ブロック状態として登録される。

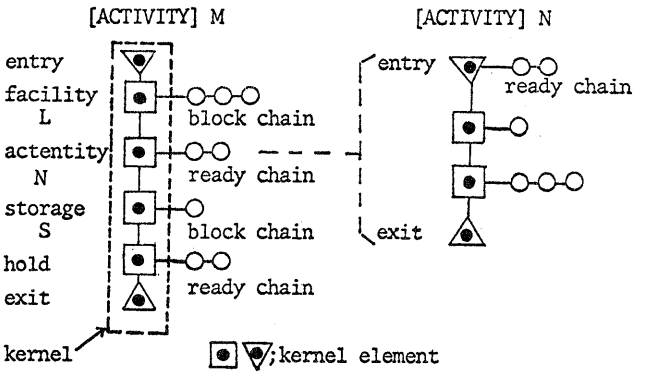

図 11　Transaction-scheduling-listの例

そして新たに，実行を待っているtransactionが1つ選択されることによって，次々とシミュレーションが進行する。Internal-form-textの終了に到達したtransactionは，executerからtransaction terminatorに渡され，そこで消滅する。

9. むすび

GMSSシミュレーションシステムでは，モデル記述の対話的な作成のみならず，そのシミュレーションの実行の直接的な制御，監視が可能である。いずれの過程でも，グラフィック端末による入力，監視ができるので，システムとのマンマシンインタラクションの機能は向上し，システムの取扱いはより容易となっている。モデル記述，その実行の対話性，即時性により，対象システムのモデル化の完了に要する時間は大巾に短縮されるものと期待できる。

また，実行過程の監視により，対象システムに特徴的な振舞を様々な角度からとらえることによって，システムの解析がより細部にわたり詳細に行なうことが可能となり，システム解析手段としてのシミュレーションの役割は大きくなることが期待できよう。

終りに，本システムのハードウェア開発に尽力いただいた阿草清滋助手に深謝いたします。
なお，この研究は文部省科学研究費の援助を受けている。

参考：[ACTIVITY] 定義を行なうフェーズでのキイボードコマンド一覧

（キイボードによるコマンドは，"K:" の後にキイの名前をつけて表わす。
AとB 2つのキイを同時に押す場合は，"K:A * B" と表わす。）

K:CTRL * T ····· コントロールは，textモードに移る。
K:CTRL * P ····· コントロールは，pictographicモードに移る。
K:CTRL * R ····· 作成中の[ACTIVITY]は，システムに登録される。

Text work area，parameter display area の操作に関するもの

K:DEL ····· cursor上の文字が削除される。cursorは動かない。
K:Left Arrow ····· cursorが左へ1文字分動く。
K:Right Arrow ····· cursorが右へ1文字分動く。

Statementの挿入，削除に関するもの。

K:CTRL * I ·····	pictographicモード	ヒットしたpictographic symbolが，working box位置に挿入される。
	textモード	text work areaの1行が，text indicator位置に挿入される。
K:CTRL * DEL ·····	pictographicモード	working box位置のsymbolを削除する。
	textモード	text indicator位置の1行を削除する。

Text indicator，working boxの移動に関するもの。両者は連動している。

K:CTRL * Up Arrow ···· working boxあるいはtext indicatorを1つ分上方へ動かす。
K:CTRL * Down Arrow ··· working boxあるいはtext indicatorを1つ分下方へ動かす。
K:CTRL * Left Arrow ···· working boxを左方へ動かす。
K:CTRL * Right Arrow ··· working boxを右方へ動かす。

文献

[1] TABATA，WADA，OHNO "Top-Down Modeling and Simulation with Graphics"，2nd. USA-JAPAN Computer-Conference，pp410～415，1975 8月
[2] 田畑，大野 "対話型シミュレーションシステム"，情報処理学会マンマシンシステム研究会資料20，1975 5月
[3] 酒井，田畑，大野 "モザイク的グラフィックスとそのGPSSプログラミング"への応用" 情報処理，Vol.17(1976)に掲載予定．
[4] 久保，伊藤，田畑，大野 "トップダウン的モデル形成とシミュレーションシステム－グラフィックスの有用性について" 情報処理学会第16回大会，No.133，1975 11月
[5] 伊藤，久保，田畑，大野 "トップダウン的モデル形成とシミュレーションシステム－GMSSシミュレータ" 情報処理学会第16回大会，No.210，1975 11月