INCREMENTAL MODELING IN THE FORWARD DIRECTION

Charles W. Turk
IBM Education Center
San Jose, California

Abstract-Incremental modeling in the forward direction is defined as a modeling process in which a desired conclusion is stated, then the known or stipulated facts which support the conclusion. The feasibility of implementing this process in a conversational computer environment is shown by reference to conversation with an experimental Simulator of State Described Systems.

Introduction

Mathematical analysis and mechanized systems for simulation directly implement an inductive approach to problem solving. That is, the detailed information pertaining to a problem is amassed, some of it is assumed away as being unimportant or representable by aggregative measures; the remainder is augmented by assumed values. The collection is then a set of "facts" which is manipulated in some mathematical structure to the point of delivering an answer or conclusion on the system being modeled (the "subject system"). This approach is highly efficient for one-shot modeling activities, but non-trivial problems are not likely ever to be satisfactorily analyzed in a single session.

Educators and practitioners of the arts and science of system analysis tell us that one should expect to build a model, exercise it to see the implications of the hypotheses built into the model, revise the model based on the conclusions reached to that point in the analysis, and repeat. A phrase that can be used to describe this activity is "incremental modeling"; as more is understood about the subject system, more or less of the information known to exist in that system will be included in the model as "facts" in succeeding operating sessions. Conventional modeling tools generally require that all or part of the model be restructured at each stage of this iterative process. In each instance the analyst again starts with the "facts" and builds a structure that leads to a desired conclusion (keeping intact, it is to be hoped, major sections of the model previously exercised). The analyst learns a great deal by manipulating evolving versions of a model, but the mechanical process of repeatedly rebuilding a model is a non-constructive use of the analyst's time. For complex models, the work involved may actually detract from his ultimate objective of progressively understanding the subject system better and better.

The Forward Approach

It is my thesis that less counter-productive work will be required of the analyst if he can take advantage of incremental modeling in a forward direction: from the desired conclusion to the "facts" necessary to support that conclusion, defining the appropriate model structure as he proceeds. Figure 1 illustrates a fragment of such logic. Note that this logic may be pursued in a forward direction through an indefinite number of stages. It terminates when there are no factors left undefined. They are defined by reference to known "facts" (ie., numbers that could be plugged-in for "Taxes", "Product Revenue", etc.). The conclusion can then be evaluated by starting at the last stage (ie., level of indentation) of the logical structure, where all the elements are known "facts" and proceeding in a reverse direction, applying the operative language to elements as they become known, until the conclusion is reached. This is a fully general concept, limited in practise by the operative language at our disposal. One can conceive an implementation in which the operative language includes not only arithmetic operations like "sum", but also statistical and probabalistic relationships.

Of course, a model may be designed to reach more than one conclusion, that is, define any number of subject system variables as functions of known "facts" (and other variables). The great advantage to this concept is that, in succeeding modeling sessions the known "facts" may themselves become conclusions (variables) and be defined by known "facts" at a greater level of detail. In this fashion the structure of a model is validated at each level of detail. Further, only those "facts" to which the model is sensitive, or those in which the analyst has a special interest need be expanded in succeeding sessions.

This is a useful concept when dealing with simple arithmetic relationships such as those represented in Figure 1. The far greater value of this concept comes in the notion that any operative function can be represented; and this will be a constructive representation if it is concise and intelligible to an analyst and his computing system.

An approach to mechanizing this concept of incremental modeling in the forward direction exists as a set of programmed functions in the time sharing system known as APL 360.[1] It has been named "Simulator of State Described Systems" (SSDS).[2]

SSDS acts in conversational style to assist a user in defining a model. It will then simulate the subject system by exercising the model and produce reports which tabulate and/or plot the movement with respect to simulated time of subsets of the variables defined in the model.

Overview of "Simulator of State Described Systems"

The nomenclature of SSDS:

a. SSDS is an on-line semi-conversational system. That is, the user of SSDS is in direct contact with a computer, and SSDS communicates with the user in conversational style. The user responds to SSDS queries with a mixture of commands to control SSDS, specifications of controls on the process of simulation, and coded specifications of his model of the system to be simulated.

b. SSDS is strictly user oriented. That is, every effort has been made to free the user of SSDS from computer conventions in language, in the entry of information, and in displaying results.

c. SSDS is an interactive hypotheses tester. That is, the user is encouraged to describe some subject system in terms of the interactions of state variables. He may then obtain from simulated activity of that system a portrayal of some of the implications of what he has described.

d. SSDS is a system for incremental modeling in the forward direction. That is, the user is able to define a model by first stating a conclusion, then filling in the details necessary to support his conclusion. He can terminate the model definition process at any point, stating that certain elements of the model known to be stochastic are of fixed value or completely deterministic functions. On the basis of these fixed or deterministic elements, he can satisfy himself as to the validity of the model structure he has defined to that point. By asking that arbitrary changes be made on selected elements, he can test the sensitivity of his model to those elements. Independently and concurrently he can define and test the actions of variables left out of his main model. He can then extend his model to react to those variables, and repeat this process an indefinite number of times.

e. SSDS is a tool for heuristic problem solving. That is, the ability to define models incrementally in the forward direction gives the user a basis for the extended development of parts of a model based on experience gained in earlier analyses.

f. SSDS is interruptable. That is the user may interrupt the process of model specification or the simulation itself to go back and correct a prior entry, to display and investigate some intermediate results, or to declare a recess (i.e., put away a copy of the system and the partially specified or simulated model so that the interrupted session can continue at a later time).

g. SSDS is a secure system. That is, the user can lock-up any copy of his defined model as it is saved in the host computer's files. No person can gain access to such locked information without an appropriate "key".

h. SSDS is virtually a fail-safe system. That is, there is almost no chance that information, once input to SSDS and saved in the host computer, will be lost due to any equipment failure in the computer or communications system, if 24 hours have elapsed since the information was saved away. (It is standard practice to create an archive copy of users' private libraries daily.) As soon as information is saved it is immune from destruction except in the event of catastrophic failure of the file device in the computer system on which the information is copied. Information entered at the terminal, but not yet formally saved is immune from destruction due to any failure of the user's terminal or the communication links between the terminal and the host computer.

The Nature of SSDS Models

Any system that is to be described and simulated using SSDS must fall within the following classification:

a. The subject system and its environment (hereafter the word system is understood to stand for the combination) is always in one of its possible "states".

A state of the system is defined by all of its "state variables" having been evaluated at some specific point in time.

A state variable is any measureable attribute of the system. It can be a rate, such as "orders received per day"; or it can be a level, such as "quantity on hand", or "price-earnings ratio".

b. Each state variable of the system is evaluated at every increment of a discrete interval of time, and may be assumed to hold its value during the following interval.

The time interval at which a state variable is evaluated need not be the same as the interval at which any other state variable is evaluated.

c. The value of a state variable may be defined as a function of the present or past value of any other state variable in the system, of its own value in any previous time period, of time, or of any quantitative or quantitative constant.

d. The values to be determined at a point in time for an interrelated set of variables do not depend on a true simultaneous condition.

e. Each state variable is defined by a function which may contain an indefinite number of the members of a set of operators taken from the following classes:

- Scalar arithmetic operators
- Transcendental operators
- Logical (combination) operators
- Probability distribution functions
- Arbitrary univariate transformations
  - (Discrete or piecewise continuous operand)
  - (Deterministic or stochastic transformation)
- Arbitrary multivariate transformations
  - (Decision table logic)

These operators allow for comprehensive mathematical description of a broad class of systems. In addition, SSDS includes mechanisms for demonstrating the time dependence of a system and for maintaining files of data on a system.

A General Function Tree

This implementation of incremental modeling in the forward direction depends on a tree-like notation that, in SSDS, has been named General Function Tree (GFT). The GFT (illustrated in Figure 2) in a stylized form of nodes and connecting limbs in which SSDS operators, residing at the nodes, are assumed to operate on SSDS operands represented by the limbs extending from the nodes. Every node produces an output that is delivered to a single result-limb. The number of limbs representing input to the node, i.e., the number of opernads, may be zero or any positive integer. (This number is always dependent on the node-operator.) The operand represented by every limb which is terminated by a node is defined in content and dimension by the operator represented at the terminating node. A limb not terminated by a node is named "leaf".

Every leaf represents an operand that must have its content and dimension assigned by the user. This assignment is made explicitly, by entry of a scalar number, or implicitly, by entry of the name of a parameter or a state variable.

As a logical construction, all limbs are defined to be of the same "length". Therefore it is meaningful to define a "Rule of Levels" for nodes and limbs

a. Level 0 is the apex of the tree, represented by the name of the state variable defined by the tree.

b. Level 1 of the nodes is the single node that delivers the value of the state variable defined by the GFT.

c. Level J of the nodes is the set of all nodes separated from the Level 0 node by the same number of intervening limbs and nodes.

d. Level J of the limbs is the set of all limbs representing the operands of the Jth level nodes.

Alternative alignments of a GFT are shown in Figure 2. Note that these alternates represent a transposition of elements, not merely a rotation. The convention on alignment of a GFT is: Every dyadic operator (an operator that expects two operands; such as "+") with its couple of operands is read from left to right or from top to bottom. The analogous rule will hold for any operator defined to operate on more than two operands.

Figure 3 represents part of a GFT that might be used to represent the logical structure of Figure 1. Note that a clock-phase name has been added as part of the definition of "Profit After Taxes" to signify the time interval at which it is to be evaluated. This tree has five levels shown, and a single explicit leaf. The leaf-operand is "INTEREST". From this, it can be assumed that "INTEREST" is the name of a state variable, defined by its own GFT, or the name of a parameter, with a value assigned separately by the user. The names enclosed in parentheses are shown for convenience only. They are not entered into SSDS. The symbol "f" is used to represent "any SSDS functional operator".

An External View of SSDS

The activity of modeling a subject system and its environment takes place in four phases. Any of the phases may be re-entered to add to or correct previous entries, or to re-run the simulation model. Within each phase SSDS will back up for correction of entries made within a logical group of entries.

a. Phase I - Specify (or re-specify) a set of controls on the simulation that is to take place. That is:

(1) Specify a clock, to include an arbitrary number of named time intervals (clock phases) and the ratios of successive pairs of them.

(2) Specify the length of time to be simulated in terms of one of the clock phases.

(3) Set the "seed" element of the random number generator.

(4) Set the maximum number of characters to be typed on a line of output. In each case SSDS will assume an appropriate "default value" if the user chooses not to make a specification.

b. Phase II - Define (or re-define) a model of the system as a set of state variables and parameters.

A state variable may measure some attribute of a single-member entity in the system or it may measure the same attribute of a collection or set of members.
For example:
State variable "REJECTS" might be the reject-rate on a particular machine, or it could be the reject-rates of a set of machines.

A state variable that measures an attribute on a set of members need not be matched by other state variables that refer to all or part of the same set of members.
For example:
State variable "REJECTS" could refer to machines 1,2,---n, while state variable "PRODUCTIONRATE" might refer to machines 1,2,---n,p,t,---z, and at the same time a state variable "DEPRECIATEDVALUE" might refer only to machines 2,6,p, and t.

The next step in the activity of defining a model using SSDS is, reading from the GFT, to enter the name of the state variable being defined, the clock-phase on which it moves, a control on the amount of data to be kept on this variable, then the operators of the GFT, a level at a time, followed by the operands, a level at a time.

As the operators of the GFT are entered SSDS may detect the presence of some that require the entry of a table of data as auxiliary information (eg., a decision table, or table representing an arbitrary function). SSDS will call for each table by naming the operator entered by the user and identifying it by its position (as determined by the Rule of Levels).

During the process of exercising the simulation model defined by the user SSDS is guided by the positional significance of all operators and operands; again relying on the Rule of Levels. For these reasons, then, it is imperative that the user follow the convention on the alignment of operands and the Rule of Levels. A null operator is available to simplify the process of applying changes to a GFT.

When all the variables have been defined via entry of GFT's, SSDS will call for the names of parameters used in the model and of historical "real" variables. The "real" variables represent measures taken on the subject system in order that, in Phase IV, direct comparisons may be made between this data and the results of the simulation.

Once all names of variables and parameters are known to SSDS, the user is given the opportunity to enter sets of initial values to be applied to any name. A scalar zero value is assumed by SSDS for any named element not explicitly initialized by the user. Re-entry of initial values for any element replaces any previous initialization of that element.

c. Phase III - Exercise the model.

Interaction with SSDS is suspended during the time it takes the host computer to "run the model". SSDS runs as a non-stop system as much as possible. That is, minor errors in model specifications will generally be overcome, and operations continued, rather than stopping and signalling the user. The user can, however, interrupt the process of simulation at any time and ask for a display of the series of values assumed by any of the state variables up to the point of interruption. He can add to or change any definitions or specifications previously made, and ask SSDS to resume the simulation.

The user can also, at a point of interruption, ask that a copy of his entire system be stored as it exists at that time. He can then end his session at the terminal, to return and resume the simulation at a later time. As another use of this ability, he can make changes in his model, resume the simulation and, if the changes lead to unsatisfactory results, return to the point at which the system was copied and resume execution of the unchanged model (or with a different set of changes).

d. Phase IV - Display results.

Any time after state variables have been defined in Phase II, the user can set up reports to be made on the results of the simulation. These reports are specified in terms of the state variables to be displayed in the form of time series, corresponding measured historical "real" data (if it has been entered in Phase II, and whether a particular series is to be printed, or plotted, or both. For any state variable that represents a measure on a set of parts of the subject system the user has the option of displaying the detail on each member of the set, or a selected subset of the members, or the average of the values taken across the entire set or any subset. Any report can always be repeated with the data on particular variables translated in time or modified by an arbitrary scale factor.

Example Conversation and Reports

Figure 4 is a reproduction of a fragment typical of an SSDS "conversation." The user entries are underlined and notes are appended in a different type style. Figure 5 is a sample output report.

Summary

The objective of this paper was to introduce the concept of incremental modeling in the forward direction. Compared to conventional systems analysis, this approach holds a potential for reducing the non-constructive time required of an analyst for the mechanics of model building in his examination of complex systems.

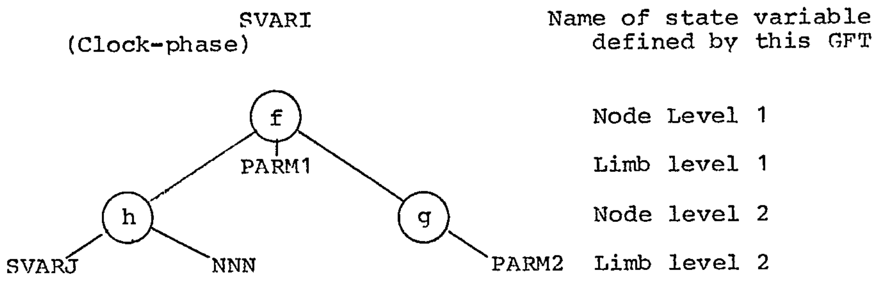
The feasibility of operating in the forward direction has been shown by the experimental implementation of a conversational system for simultion of a broad class of subject systems. This is an open-ended approach in that a conversational system could be devised to act on almost any operative language used in the basic structure illustrated in Figure 1.

References:

1. IBM Corporation, APL 360 User's Manual (GH20-0683), Armonk, New York, 1969
2. Turk, C. W., "A Simulator of State Described Systems," Unpublished Ph.D. dissertation, Stanford University, 1970

```
"Profit After Taxes" is "Profit" less "Taxes"
where  "Profit" is "Gross" less "Interest"
where     "Gross" is "Revenue" less "Costs"
where        "Revenue" is the sum of "Product Revenue"
where          "Product Revenue" is, for each product, - - -
and          "Costs" is the sum of - - -
and       "Interest" is - - -
and    "Taxes" is - - -
```

FIGURE 1
Fragment of the Logical Structure of a Budget Model



f g h are any SSDS functional operators
SVARI,SVARJ represent the names of two state variables
PARM1,PARM2 represent the names of two parameters
NNN represents an actual number
Limb level 1 contains 3 limbs, one of which is a leaf

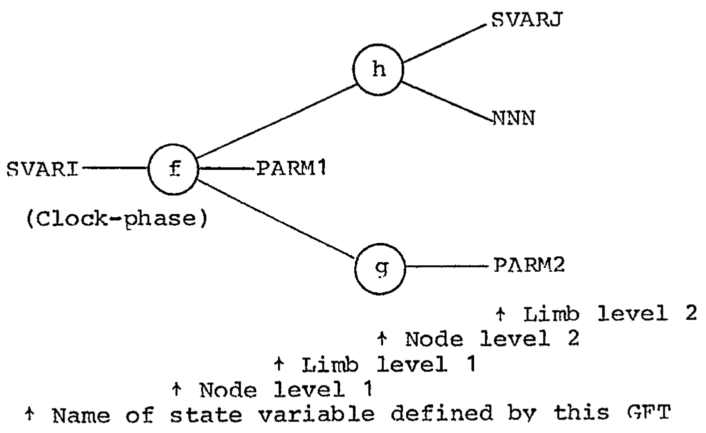


FIGURE 2
Alternative Alignments of a General Function Tree

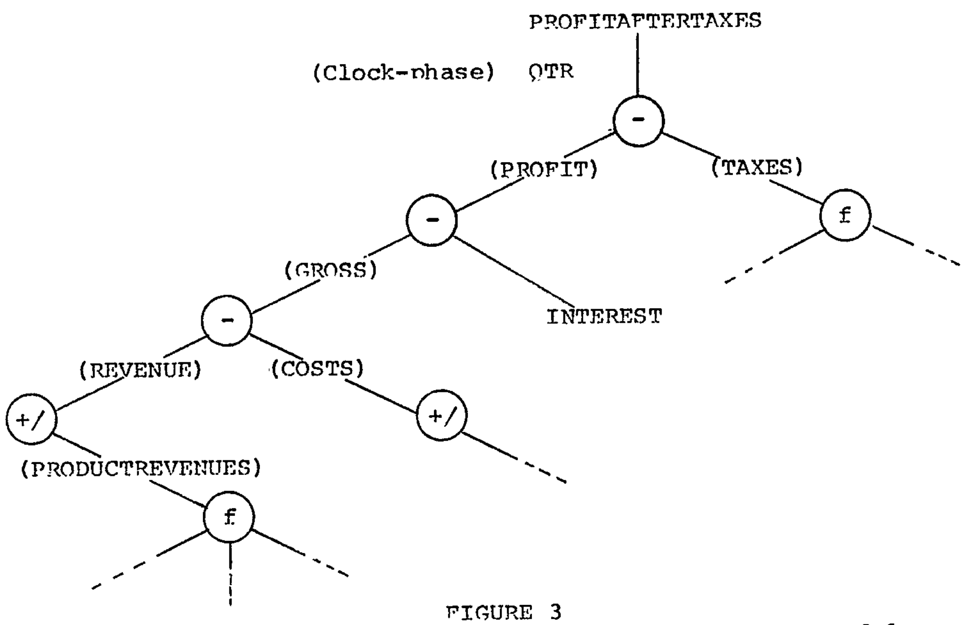


FIGURE 3
Fragment of a General Function Tree for a Budget Model

```
)391768:LOCKWORD                        Initial sign-in.
OPR:  UP UNTIL ABOUT 0830 PDT MONDAY

013) 14.06.41  05/28/70 CWTURK          Time and date of
                                          sign-in.


A P L \ 3 6 0


      )LOAD 111 SSDS

SAVED  10.10.13 05/28/70                Time SSDS was last
                                          saved-away.

      START NEW MODEL

05/28/70            14:07:15            Time now.


      ∘ ∘ ∘  SYSTEM PARAMETERS SPECIFICATION ∘ ∘ ∘

THE LINE-LENGTH IS SET TO 130 CHARACTER SPACES.
ACCEPT THAT WITH NULL C/R, OR ENTER NEW VALUE  (30≤N≤130)

62

THE RANDOM NUMBER GENERATOR IS 'SEEDED' WITH 16807
ACCEPT WITH NULL C/R  OR ENTER YOUR OWN
                                        There is no visible
                                          indicator of a null
                                          C/R.

      ∘ ∘ ∘  MODEL DEFINITION PHASE  ∘ ∘ ∘

WHAT ARE THE NAME OF THE PHASES OF YOUR POLYPHASE CLOCK?
WEEK HOORS
]BACK                                   When ']BACK' is ambiguous
 LINE TO BE CHANGED:                      SSDS will ask you what
                                          you want to change.

 WEEK HOORS
     5  /1                              Slashes eliminate characters.
                                        Digits cause spaces ahead
 WEEK      HO RS                          of their relative position.
     DAYS   U


 OK?                                    A chance to re-correct.
 YES
```

FIGURE 4a
An SSDS Conversation

```
CONTINUE  ENTERING

HOW MANY HOURS PER DAYS?
6

HOW MANY DAYS PER WEEK?
5

ENTER INCREMENT BY WHICH 'HOURS' SHOULD BE STEPPED, IF NOT 1
3

CLOCK PHASE ON WHICH SIMULATION WILL BE CUT-OFF?
WEEK

VALUE OF 'WEEK' AT WHICH SIMULATION SHOULD BE ENDED?
30


NOW, TO DEFINE THE STATE VARIABLES OF YOUR MODEL ° ° °

NAME OF STATE VARIABLE
LONGVARIABLENAME

CLOCK PHASE ON WHICH 'LONGVARIABLENAME' VARIES?
WEEKS

NAME MUST MATCH ONE OF:    WEEK DAYS HOURS
CLOCK PHASE ON WHICH 'LONGVARIABLENAME' VARIES?

WEEK

HOW MANY VALUES OF   'LONGVARIABLENAME'   SHOULD BE RETAINED?
        ('ALL' IS A VALID ENTRY)
ALL


ENTER OPERATOR ARRAY

+                                           Level 1 node.

÷ FD                                        A level of operators.
                                            "FD" is Decision Table Opr.
                                            Null C/R
ENTER CORRESPONDING OPERAND ARRAY

° °                                         Null characters to indicate
                                              structure of GFT.
PARM1  SVAR1                                Leaf-operands. "FD" takes
                                              none.
```

FIGURE 4b
SSDS Conversation (Continued)

```
                   ARBITRARY REPORT NAME
07/06/70                16:35:21

ANY NUMBER OF HEADER LINES

AS OF THE END OF
30 WEEKS
 0 DAYS
 0 HOURS

SVAR4 SVAR5
 WEEK  WEEK       0      20      40      60
__*_____o___      ↓__|___↓___|___↓___|___↓________
            |WEEK|
 42.25 53.50| 30 |                *   o
 40.50 51.25| 29 |               *    o
 38.75 49.00| 28 |               *   o
 37.00 46.75| 27 |              *   o
 35.25 44.50| 26 |             *   o
 33.50 42.25| 25 |            *   o
 31.75 40.00| 24 |            *  o
 30.00 37.75| 23 |           *  o
 28.25 35.50| 22 |          *  o
 26.50 33.25| 21 |          * o
 24.75 31.00| 20 |         * o
 23.00 28.75| 19 |        *  o
 21.25 26.50| 18 |        * o
 19.50 24.25| 17 |       * o
 17.75 22.67| 16 |      * o
 16.00 21.08| 15 |     * o
 14.93 19.50| 14 |     * o
 13.93 17.92| 13 |     *o
 13.03 16.33| 12 |    * o
 12.29 14.75| 11 |    *o
 11.76 13.17| 10 |    o
 11.48 11.58|  9 |    o
 11.50 10.00|  8 |   o*
 11.87  8.42|  7 |  o *
 12.64  6.83|  6 |  o *
 13.86  5.25|  5 | o   *
 15.57  3.67|  4 |o    *
 17.82  2.08|  3 |o     *
 20.67  2.37|  2 |o      *
 24.15  2.45|  1 |o         *

----------------------------------------------------
                  ↑  |   ↑   |   ↑   |   ↑
                  0      20      40      60


ANOTHER?
NO
```

This report, by being vertical, can cover an indefinite number of time periods.

FIGURE 5
Sample SSDS Output