Int. J. General Systems, Vol. 13, pp. 307–322
Photocopying permitted by license only

SAPS-II: A NEW IMPLEMENTATION OF THE SYSTEMS APPROACH PROBLEM SOLVER

FRANÇOIS E. CELLIER and DAVID W. YANDELL
Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona 85721, U.S.A.

(Received 21 January 1987; in final form 13 April 1987)

In this paper, we describe a reimplementation of the Systems Approach Problem Solver that was originally designed and implemented by H. J. J. Uyttenhove.[1,2] In this reimplementation, emphasis was put on a clean and flexible user interface that allows the user to conveniently combine several of the SAPS basic operations for solving more complex problems. SAPS-II[3] was implemented as a toolbox (function library) within the framework of the CTRL-C program,[4] an interactive matrix manipulation language developed originally for computer-aided control system design, and enhanced later for other purposes such as signal analysis and statistical operations.[5,6] From the point of view of the CTRL-C software, the SAPS-II library can be viewed as yet another enhancement for the purpose of systems analysis and synthesis. From the point of view of the SAPS-II software, CTRL-C can be viewed as a software shell for convenient embedding of the SAPS algorithms.

INDEX TERMS: General System Problem Solving framework, optimal mask analysis, forecasting, reconstruction analysis, computer software, data flow.

INTRODUCTION

The SAPS software was developed as an additional tool for the analysis and synthesis of *systems* using the general system solving framework proposed by G. J. Klir,[7] where a system is viewed as a potential source of data. Data from and about this system are then gathered under a given observational mode which can also be called the *experiment* that is applied to the system. The SAPS algorithms perform operations on these system observations which we call the *raw data* describing the system.

At a first glance, SAPS can be viewed as a system identification tool comparable to others such as regression analysis, nonlinear programming, etc. To some extent, this notion is correct. However, SAPS exhibits a couple of specific characteristics that make this approach different from other well established system identification mechanisms:

1) Most identification tools require an assumed model structure to start with. This structure given, they identify the (yet unknown) parameters of the structure. In this way, errors inherent in the chosen structure remain unnoticed, and the user is led to believe in the results of an identification process on the basis of a match between the observed data and the identified model. Several disasters have already taken place because of (unjustified) confidence in a chosen model structure that did not portray essential properties of the given system. Typical examples of such disasters are in ecological system modeling where e.g., a new species was introduced into a given ecosystem with the goal of controlling the population of some sort of pest. In the simulation model, this approach did work well whereas,

in practice, it failed drastically because additional interactions between the new species and other species of the existing ecosystem remained unnoticed. SAPS tries to avoid this pitfall by not presupposing any structure. The SAPS model contains no information that was not present in the observed data itself. This is one of the greatest strengths of the SAPS analysis. Unfortunately, the aforementioned pitfall cannot be totally avoided as the observational mode (experiment) must be preselected even in SAPS. As shall be demonstrated in a related paper,[8] an unlucky choice of the observational mode may easily hide some essential properties of the underlying system in the same way as the presupposed structure did.

2) Most identification procedures generate a system model that is disassociated from its own confidence information. Once a model identification has been completed, it is all too easy to forget the limitations (experimental frame) under which the model was derived, simply because the generated model will not hesitate to forecast system behavior over a ridiculously long time horizon, and display results with a double precision 14 digits out of which not a single one is zero (!) As a typical example of the shortcomings of such an identification procedure, we may mention the world model by Forrester.[9] The SAPS forecasting mechanism draws simultaneously from both the model pattern and the confidence information, and will decline a request to forecast behavior that cannot be justified on the basis of the observed data. In SAPS, time horizon and experimental frame of a model are identified together with the model pattern itself, and cannot be disassociated from the derived model.

In a SAPS analysis, the greatest care must be taken in the determination of the appropriate observational mode. To this date, SAPS does not provide the user with any information of whether or not the observational mode chosen is appropriate for his purpose. In particular, the *recoding facility* of SAPS (which is intimately linked to the determination of the observational mode) does not provide the user with any inside of whether essential data are lost in the recoding or not. More research is still needed in order to provide the user with a maximum amount of information and guidance during the recoding process. Once the raw data model is determined (that is: after the recoding step has been completed), the various SAPS analysis and synthesis procedures are performed in a fairly automatic manner, and they work in general quite well. More guidance is still needed in order to tell the user how significant his collected data are. E.g., an *optimal mask analysis* can determine a powerful forecasting model if and only if the underlying raw data are sufficiently many, and sufficiently expressive. However, the currently implemented algorithm does not tell the user whether or not this is the case.

The current implementation of SAPS-II simply duplicates the algorithms available in the original SAPS software while providing a largely improved user interface. Several of the remarks made above suggest improvements and enhancements of the existing algorithms. These form the basis of an ongoing research effort, and will be reported at a later stage.

DATA ENTITIES IN SAPS-II

Like most of the first generation CAD tools, the original SAPS software concentrated on the *algorithms* to be implemented. These algorithms were im-

plemented as independent program modules that read input data from punched cards in a fixed format, and produce output on a listing file. The *data* fed into these algorithms that is the information about the system to be analyzed, was considered of much less concern, and little effort was spent on a convenient and flexible management of this data. In the original SAPS software, it was e.g. impossible to combine several of the available algorithms to form larger entities, or add new algorithms to the program without modifying the source code itself.

In SAPS-II, major emphasis was put on *data flow*. Here, the SAPS algorithms are viewed as *operators* mapping one data structure into another data structure. Data are maintained in a flexible data base, and can be manipulated at will.

The *raw data model* consists of one or several *trajectory vectors* where each column denotes one trajectory, i.e. the time history of one particular variable, while each row specifies one recording (that is: measurement at one time instant) of all variables contained in the trajectory vector. The time instants, at which recordings (measurements) are taken, form a *domain variable* which can also be represented as a vector, and e.g. be stored as the first column of the trajectory vector when needed. However in SAPS, these time values are rarely used, and are therefore often omitted. In data base management, this type of data structure is often referred to as a *relation*, and the domain variable is called the *key variable* of the relation. However, the data structure can also be viewed as a *matrix* of either integer or real arguments.

When a *basic behavior analysis* is performed on the raw data, another data structure results which we call a *behavior relation*. This data structure consists of an alphabetically reordered and reduced set of recordings together with a vector of probabilities denoting the frequency of occurrence of these individual recordings (states). In other words, the "BEHAVIOR" operator is a unary operator that operates on a data structure of type trajectory vector, and produces another data structure of type behavior relation. Both trajectory vectors and behavior relations can be represented by series of matrices and vectors.

Looking through the list of SAPS operators, one notices that all SAPS data can be conveniently represented by matrices and vectors some of which are of type integer (such as the recoded raw data matrix), while others are of type real (such as the behavior probability vector). A flexible user interface can be created by providing a tool for general purpose *matrix manipulations* where each SAPS algorithm is represented by an operator (that is: a function) mapping one set of such matrices into another set of matrices.

CTRL-C: A SHELL FOR IMPLEMENTING SAPS ALGORITHMS

A very convenient matrix manipulation environment (MATLAB) was created in 1980 by Cleve Moler.[10] In MATLAB, the only data structure supported is a double precision complex matrix. The basic MATLAB operators implement a large subset of the standard algorithms used in linear algebra such as linear system solutions, eigenvalue computations, etc. MATLAB offers a very convenient user interface that is so natural and intuitive that even beginners can learn how to use MATLAB within a couple of minutes just as they learn how to use a simple pocket calculator. In MATLAB, matrices are entered as follows:

$$A = [1, 2, 3; 4, 5, 6; 7, 8, 9]$$

or alternatively:

$$A=[1\ \ 2\ \ 3$$
$$4\ \ 5\ \ 6$$
$$7\ \ 8\ \ 9]$$

that is: elements in different columns are separated by either a comma or a space, while elements in different rows are separated by either a semicolon or a carriage return. Each element of a matrix can be an arbitrary expression including submatrices, thus:

$$B=[[0;\ 0;\ 0],\ \mathrm{EYE}(3);\ [2,\ 3,\ 4,\ 5]]$$

denotes the matrix:

$$B=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 2 & 3 & 4 & 5\end{bmatrix}$$

where EYE(3) denotes a unity matrix of dimension 3×3. Standard matrix operators are defined in a very natural way:

$$F=(C+D')*E$$

where C, D and E are matrices, creates a new matrix F as the product of the sum of C and the transpose of D with E. Expressions can be nested to any depth.

Stochastic variables are generated by a built in random number generator. E.g. does

$$X=\mathrm{RAND}(4,1)$$

create a random column vector with 4 rows, the elements of which are uniformly distributed between 0 and 1.

The problem $\mathbf{B}*\mathbf{y}=\mathbf{x}$ can now be solved either by inverting the matrix B:

$$Y=\mathrm{INV}(B)*X$$

or alternatively (by extending the usual definition of the division operator) through the exression:

$$Y=B\backslash X$$

(X from left divided by B) which, of course, is equivalent to:

$$Y=(X'/B')'$$

However, these expressions result in a Gaussian elimination being performed in place of the explicit computation of the inverse of B if the matrix B is square. For

B being rectangular, the either over- or underspecified set of linear equations will be solved in a least square's sense.

Unfortunately, the original MATLAB system does not lend itself conveniently to user extensions. For this reason, we chose for our task the program CTRL-C[4] which is a superset of MATLAB.[10] In CTRL-C, user functions may be coded that automatically extend the set of system functions, and which thereafter can be called just like any system defined CTRL-C function (such as the EYE( ), RAND( ), and INV( ) functions demonstrated above). All SAPS functions are stored in a CTRL-C function library which can be activated by the command:

DO SAPS:SAPS

Thereafter, a basic behavior analysis can e.g. be performed by writing:

$$[B, P] = \text{BEHAVIOR(RAW)}$$

where BEHAVIOR( ) is a SAPS-II function with one input argument (RAW) and two output arguments (B and P).

OPTIMAL MASK ANALYSIS IN SAPS-II

One of the standard problems in signal analysis is to determine how different signals are correlated with each other. In terms of the SAPS methodology, this problem is solved by an optimal mask analysis. A mask is a pattern of relations between different variables at different instants of time. E.g. does the following mask:

$$\text{MASK} = \begin{bmatrix} -1 & 0 & 0 \\ -2 & -3 & 0 \\ 0 & 0 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

indicate that in a three variable system the third variable (v_3) is related to the first variable (v_1) three time steps back, the same variable two time steps back, the second variable two time steps back, and the third variable one time step back. Thus, the above mask can be read as follows:

$$v_3(t) = f\{v_1(t-3\Delta t), v_1(t-2\Delta t), v_2(t-2\Delta t), v_3(t-\Delta t)\}$$

In an optimal mask analysis, all possible mask candidates are evaluated, and among those, the one is chosen which exhibits the most deterministic input/output behavior, that is: which minimizes the Shannon entropy. In SAPS-II, mask candidates are denoted as follows:

$$\text{MCAN} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

where elements marked as -1 are potential inputs, elements marked by positive integers are outputs, and elements marked by 0 are to be ignored.

The optimal mask analysis function operates on a raw data model and a mask candidate, and evaluates the optimal mask:

$$\text{MASK} = \text{OPTMASK(RAW, MCAN)}$$

As an example, let us consider the haunted house problem proposed by H. J. J. Uyttenhove.[1,2] It consists of a three variable model with the three components radio (v_1), light (v_2), and ghosts (v_3):

$v_1 = 0$: radio is off
$= 1$: radio is on
$v_2 = 0$: light is off
$= 1$: light is on
$v_3 = 0$: ghosts keep quiet
$= 1$: ghosts are laughing
$= 2$: ghosts are walking around
$= 3$: ghosts walk and laugh

Measurements were taken that provided the following raw data model:

$$\text{RAW} = \begin{bmatrix} 0 & 0 & 3 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 3 \\ 1 & 1 & 3 \\ 1 & 0 & 1 \\ 0 & 1 & 3 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

It was decided that the behavior of the ghosts is strongly correlated with the use of the lights and the radio in the house. Therefore, an optimal mask is to be determined that allows the development of a strategy that will keep the ghosts quiet during the night.

The CTRL-C macro given in Figure 1 can be used to perform an optimal mask analysis of this three variable model. This macro should hopefully be almost self explanatory. The use of SAPS-II functions is freely intermixed with that of standard CTRL-C functions. This is feasible because both operate on the same data structures. In our example, the OPTMASK( ) function is used with secondary

```
//Example SAPS Book 7.4.1
//
//
DISPLAY('The Haunted House')
DISPLAY('*****************')
DISPLAY(' ')
DO saps:saps
repo=1;
DISPLAY('Raw Data:')
LOAD <demo:haunted
raw
DISPLAY('Selection of INPUT/OUTPUT Mask Candidates')
mcan=[-1 -1 0; -1 -1 0; -1 -1 1]
[mask,hm,hr,q,mhis]=OPTMASK(raw,mcan);
DISPLAY('Optimal Mask:')
mask
DISPLAY('Hit <CR> to continue')
PAUSE
DISPLAY('Plot of Entropy and Quality versus Complexity')
PAGE
PLOT([hm,5*q])
TITLE('Optimal Mask Evaluation')
XLABEL('Complexity')
YLABEL('Entropy and Quality')
REPLOT
PAUSE
ERASE
DISPLAY('Calculate Input/Output Behavior')
io=IOMODEL(raw,mask)
DISPLAY('Input/Output Behavior')
[b,p]=BEHAVIOR(io);
DISPLAY('Input/Output Matrix')
[s,p2]=IOMATRIX(io,3);
RETURN
```

Figure 1 Optimal mask analysis of the haunted house example.

output parameters providing additional information about the history of mask evaluations. The IOMODEL( ) function is another SAPS-II function that evaluates an input/output model out of the raw data model and the previously evaluated optimal mask. The IOMATRIX( ) function generates the state transition matrix for the previously computed input/output model. Its second parameter (3) indicates that the first three columns of the input/output model denote inputs while the remaining column denotes an output. REPO is a global (SAPS-II) variable that determines the amount of intermediary output to be displayed. The resulting optimal mask in this case is:

$$\text{MASK} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & -3 & 1 \end{bmatrix}$$

Evidently, this example is somewhat artificial, but it is also sufficiently simple to allow a demonstration of the properties of the SAPS-II software without obfuscating the desired message through details of an involved and more realistic problem.

In this example, the mode of observation was chosen by deciding that there existed a correlation between the behavior of the ghosts on the one hand, and the

setting of radio and light on the other. The SAPS software was not able to help us in this matter. However, there was no need for any recoding of data as all three variables assumed discrete values from the beginning.

RECONSTRUCTION ANALYSIS IN SAPS-II

Once an input/output model has been determined by an optimal mask analysis, the SAPS methodology allows to identify *structure* by grouping subsets of variables together that seem to be more closely related to each other than to the remaining variables. Each potential structure is analysed by decomposing it into its subsystems, recombining the subsystems to a whole, and evaluating the resulting reconstruction error. In an optimal structure analysis, all possible structures are evaluated in this manner, and the most refined structure is chosen that exhibits an acceptably small reconstruction error.

SAPS-II distinguishes between three different types of structure representation. A *causal structure* lists the variables that form the subsystems as a row vector whereby subsequent subsystems are separated by a zero. Below this row vector, masks are coded that show the causal relation between the variables of the subsystems. For example, the structure specified in Figure 2 can be coded as:

$$\text{ISTC} = \begin{bmatrix} 1 & 4 & 6 & 0 & 2 & 3 & 4 & 5 & 0 & 5 & 6 & 7 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & -1 & 0 & -2 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & -2 & 0 & 1 \end{bmatrix}$$

By eliminating the mask, we obtain a *composite structure*. Obviously, the composite structure contains less information than the causal structure. The direction information is lost, and also the timing information is gone. In SAPS-II, the composite structure is represented by the first row of the causal structure:

$$\text{ISTR} = [1\,4\,6\,0\,2\,3\,4\,5\,0\,5\,6\,7]$$

which can mean e.g. the structure shown in Figure 3, but also the very different structure exhibited by Figure 4. It can be easily verified that both systems contain the same three subsystems, and are thus identified by the same composite structure. The composite structure no longer contains information about which variables are inputs and which are outputs.

The third structure representation in SAPS-II is the *binary structure*. A binary structure is an ordered list of all binary relations between variables belonging to the same subsystem. The operator:

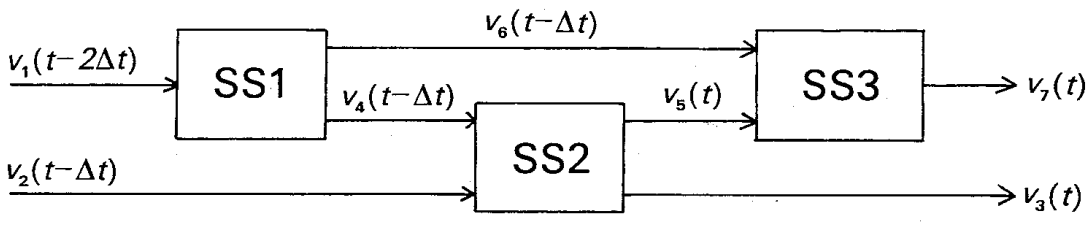


Figure 2 Graphical representation of a causal structure.

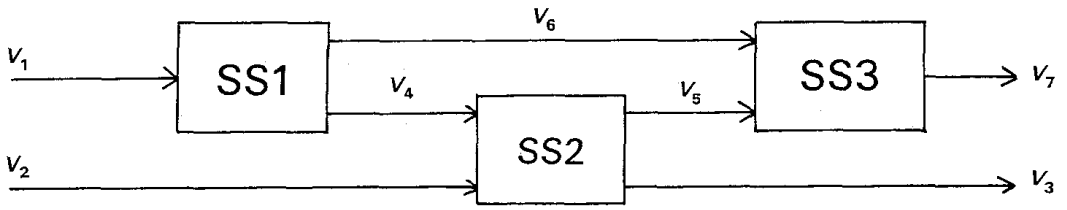


Figure 3 Graphical representation of the same structure without causality information.

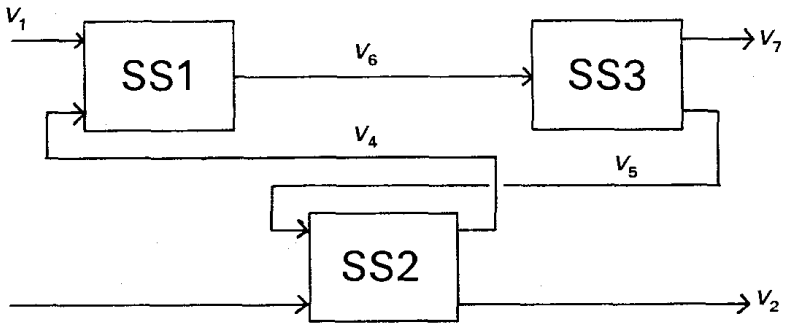


Figure 4 Graphical representation of a different structure being indistinguishable from the previous one if the causality information is not provided.

$$\text{ISTB} = \text{BINARY(ISTR)}$$

generates the binary structure out of the composite structure. For our example, the resulting binary structure is:

$$\text{ISTB} = \begin{bmatrix} 1 & 4 \\ 1 & 6 \\ 2 & 3 \\ 2 & 4 \\ 2 & 5 \\ 3 & 4 \\ 3 & 5 \\ 4 & 5 \\ 4 & 6 \\ 5 & 6 \\ 5 & 7 \\ 6 & 7 \end{bmatrix}$$

Again, the binary structure contains less information than the composite structure. For instance, the system in Figure 5, with the composite structure:

$$\text{ISTR} = [1460234505670456],$$

possesses the same binary structure as our previous system, even though it contains an additional subsystem (SS4), and thus a different composite structure.

In SAPS-II, the command:

$$\text{ISTR} = \text{COMPOSE(ISTB)}$$

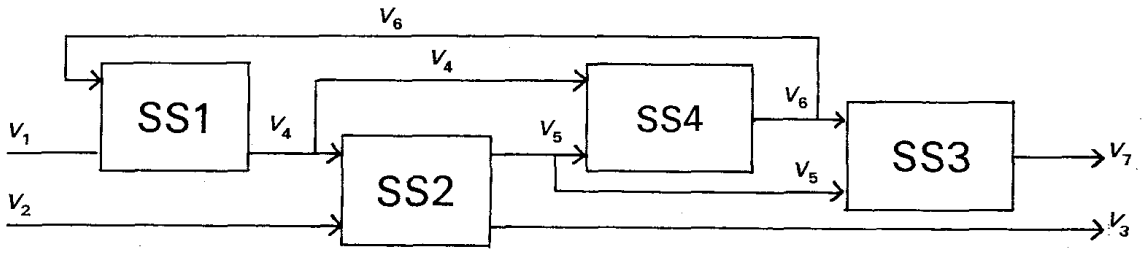


Figure 5 Graphical representation of yet another structure which is indistinguishable from the previous ones if only its binary representation is given.

will always produce the minimal composite structure among all possible composite structures.

Causal structures are currently planned, but they have not yet been implemented.

The evaluation of the reconstruction error of a proposed system bases on its composite structure and its behavior relation:

$$\text{ERR} = \text{STRUCTURE}(\text{ISTR}, B, P)$$

computes the reconstruction error of the system represented by its behavior relation ($[B, P]$) and the suggested composite structure (ISTR).

However, optimal structure analysis bases on the binary structure representation. Three different structure optimization algorithms have currently been implemented. The *structure refinement* algorithm starts with a totally interlinked composite structure in which all possible binary relations are present. This structure, of course, shows no reconstruction error at all as there is nothing to be reconstructed. Binary relations are cancelled one at a time, and at each level (that is: number of binary relations), the structure is chosen to continue which exhibits the smallest reconstruction error. The iteration goes on until the reconstruction error becomes too large.

The *structure aggregation* algorithm starts with a system in which each variable forms a substructure of its own that is not linked to any other variable. No binary relations are thus initially present, and the reconstruction error of this structure is very large. Binary relations are added one at a time, and at each level the structure is chosen to continue that shows the largest reduction of the reconstruction error. The iteration goes on until the reconstruction error becomes sufficiently small.

The *single refinement* algorithm starts similar to the structure refinement algorithm. However, instead of cancelling one binary relation only, all binary relations that exhibit a sufficiently small reconstruction error are cancelled at once, and only one step of refinement is performed.

All three algorithms are suboptimal algorithms, since neither of them investigates all possible structures. Therefore in a sufficiently complex system, the three algorithms may well suggest three different structures, and it often pays off to try them all. The single refinement algorithm is much cheaper than the other two, and yet it performs amazingly well. Thus, in a real time (on-line) structure identification, this will probably be the best algorithm to use.

In SAPS-II, optimal structure analysis is performed by the following function:

$$\text{ISTR} = \text{OPTSTRUC}(B, P, \text{ERRMAX}, \text{GROUP}, \text{ALGOR})$$

where $[B, P]$ is the behavior relation of the system to be analyzed, and ERRMAX

is the largest reconstruction error to be tolerated. The GROUP parameter allows to aid the optimal structure algorithm by providing *a priori* knowledge about the structure to be analyzed. E.g. would the grouping information:

GROUP=[1 2 3 1 0 4]

tell the optimization algorithm that, in a six variable system, the first and the fourth variable appear always together whereas the fifth variable is certainly disassociated. Thus, the six variable system is effectively reduced to a four variable system. If no *a priori* knowledge about the structure exists, the grouping information must be coded as:

GROUP=[1 2 3 4 5 6]

which, in CTRL-C, can be simplified to:

GROUP=1:6

The ALGOR parameter finally tells the analysis which of the three algorithms to use. Possible values are:

ALGOR='REFINE'

ALGOR='AGGREGATE'

ALGOR='SINGLEREF'.

ISTR is the resulting composite structure of the system.

One more possibility was built into SAPS-II. After an optimization has taken place, the resulting structure can be postoptimized by applying the single refinement algorithm once more to each of its substructures. This algorithm is executed by the command:

ISTP=SINGLEREF(ISTR, *B*, *P*, ERRMAX)

As an example, let us consider the open heart surgery problem proposed by H. J. J. Uyttenhove.[1,2] This problem consists of a six variable model with the following variables:

VAR_1—Systolic Blood Pressure
1= 0.0 to 75.0
2= 75.0 to 100.0
3=100.0 to 150.0
4=150.0 to 180.0
5=180.0 to highest

VAR_2—Mean Blood Pressure
1= 0.0 to 50.0
2= 50.0 to 65.0
3= 65.0 to 100.0
4=100.0 to 110.0
5=110.0 to highest

VAR_3—Central Venous Pressure
2= 0.0 to 4.0
3= 4.0 to 20.0
4=20.0 to highest

VAR_4—Cardiac Output
1=0.0 to 2.0
2=2.0 to 3.0
3=3.0 to 7.0
4=7.0 to highest

VAR_5—Heart Rate
1= 0.0 to 50.0
2= 50.0 to 60.0
3= 60.0 to 100.0
4=100.0 to 110.0
5=110.0 to highest

VAR_6—Left Atrial Pressure
1= 0.0 to 1.0
2= 1.0 to 4.0
3= 4.0 to 20.0
4=20.0 to highest

As can be seen, all variables in this example are of type real. Thus, the first step in the SAPS analysis must consist of a recoding step. In this example, the following methodology was used: Each variable is digitized using five different levels where:

1: =much too low
2: =too low
3: =normal
4: =too high
5: =much too high

Expert knowledge was used to determine the breakpoints between the five different levels for each variable. An inappropriate choice of these breakpoints may easily jeopardize the success of the entire analysis. However, SAPS currently does not support the user in this decision yet.

In SAPS-II, the recoding of the measurement data can be performed by the command sequence shown in Figure 6. Measurements are assumed to be stored in the matrix MEAS. One column after the other is recoded into a vector *R* which is then concatenated from the right to the previous vectors. The raw data model is finally contained in the matrix RAW. The second parameter of the RECODE( ) function tells the system which type of recoding is to be performed. In our example, the 'DOMAIN' algorithm was used. Several other types of recoding are available in SAPS-II as well.

The recoded raw data model was then stored as a data file by use of the CTRL-C function:

SAVE RAW>DEMO:HEART

```
from=[ 0.0   75.0  100.0  150.0  180.0
       75.0  100.0  150.0  180.0  999.9];
to=1:5;
raw=RECODE(meas(:,1),'DOMAIN',from,to);
from=[ 0.0  50.0   65.0  100.0  110.0
       50.0  65.0  100.0  110.0  999.9];
r=RECODE(meas):,2),'DOMAIN',from,to);
raw=[raw,r];
from=[0.0   4.0   20.0
       4.0  20.0  999.9];
to=2:4;
r=RECODE(meas(:,3),'DOMAIN',from,to);
raw=[raw,r];
from=[0.0  2.0  3.0    7.0
       2.0  3.0  7.0  999.9];
to=1:4;
r=RECODE(meas(:,4),'DOMAIN',from,to);
raw=[raw,r];
from=[ 0.0  50.0   60.0  100.0  110.0
       50.0  60.0  100.0  110.0  999.9];
to=1:5;
r=RECODE(meas(:,5),'DOMAIN',from,to);
raw=[raw,r];
from=[0.0  1.0   4.0   20.0
       1.0  4.0  20.0  999.9];
to=1:4;
r=RECODE(meas(:,6),'DOMAIN',from,to);
raw=[raw,r];
```

Figure 6 Recoding the data for the open heart surgery example.

The CTRL-C macro given in Figure 7 performs an optimal structure analysis on the previously saved raw data model.

The analysis starts by applying the structure refinement algorithm to the behavior relation of the system. No *a priori* knowledge about the structure is yet available. The resulting composite structure is:

$$IS1=(1,2,4)(1,3,4)(2,4,5)(2,4,6)$$

A postoptimization of this structure leads to:

$$ISR1=(1,2,4)(1,3,4)(2,4,6)(5)$$

Obviously, the fifth variable (heart rate) is only weakly related to the other variables in the system.

Next, the structure aggregation algorithm is applied. This algorithm suggests the composite structure:

$$IS2=(2,4,6)(1,2,3,4)(5)$$

Postoptimization leads to:

$$ISR2=(3,4)(1,2,4)(2,4,6)(5)$$

```
// SAPS Book Example 8.3.1.2
//
//
DO saps:saps
DISPLAY('Surgery patient structure')
DISPLAY('************************')
DISPLAY(' ')
repo = 1;
LOAD<demo:heart
DISPLAY('Basic behavior:Probability and States')
[b, p] = BEHAVIOR(raw);
DISPLAY('Selecting a grouping mask')
igr = 1:6
DISPLAY('Selecting the maximum tolerated error')
ermax = .016
DISPLAY('Calculate the optimal structure by refinement')
isl = OPTSTRUC(b,p,ermax,igr,'refine')
DISPLAY('Try to refine this result once')
isrl = SINGLEREF(isl,b,p,ermax)
DISPLAY('Calculate the optimal structure by aggregation')
is2 = OPTSTRUC(b,p,ermax,igr,'aggregate')
DISPLAY('Try to refine this result once')
isr2 = SINGLEREF(is2,b,p,ermax)
DISPLAY('Calculate the suboptimal structure by single-step method')
is3 = OPSTRUC(b,p,ermax,igr,'singleref')
DISPLAY('Try to refine this result once more')
isr3 = SINGLEREF(is3,b,p,ermax)
DISPLAY('Results are different, but indicate that (1,4) are always')
DISPLAY('together, while (5) seems pretty unimportant')
DISPLAY('Select grouping mask accordingly, and retry')
igr = [1 2 3 1 0 4]
DISPLAY('Calculate the optimal structure by refinement')
isl = OPSTRUC(b,p,ermax,igr,'refine')
DISPLAY('Calculate the optimal structure by aggregation')
is2 = OPTSTRUC(b,p,ermax,igr,'aggregate')
DISPLAY('Calculate the suboptimal structure by single-step method')
is3 = OPTSTRUC(b,p,ermax,igr,'singleref')
DISPLAY('Now, the results are the same')
DISPLAY("Try a final refinement step')
is = SINGLEREF(is3,b,p,ermax)
DISPLAY('Comparing the results, we may come to the conclusion')
DISPLAY('that the correct structure is indeed: [1,2,4][1,3,4][1,4,6][5]
RETURN
```

Figure 7 Reconstruction analysis of the open heart surgery example.

Finally, we try the single refinement algorithm. This time, the following composite structure is found:

$$IS3 = (3,4)(4,6)(1,2,4)(5)$$

and postoptimization does not change the structure.

Analyzing the different suggested structures, it becomes obvious that the heart rate does not need to be considered at all. It also shows that there exists a very strong link among variables one and four. Thus, those two variables can easily be grouped together. The analysis is now repeated applying the appropriate grouping information. This time, all three algorithms suggest the same composite structure:

$$ISTR = (1,2,4)(1,3,4)(1,4,6)(5)$$

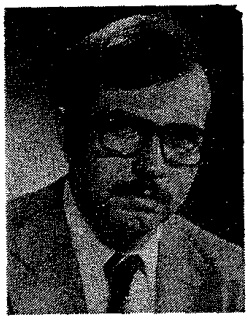
which seems to be a good working hypothesis for continuation of the system analysis.

CONCLUSIONS

SAPS-II is a new tool for the analysis and synthesis of raw data models using the methodology of general system theory. As compared to the original SAPS system, SAPS-II exhibits much more flexibility, and allows the user to easily combine different algorithms into new hierarchically higher components. All features of the original SAPS system have been reproduced in SAPS-II with the exception of the *meta analysis* module for the identification of variable structure systems. Implementation of this algorithm would be straight forward, but we decided against an implementation of the proposed algorithm, as we have not yet seen examples of meta analysis that proved the proposed methodology to be sound. More research is still needed to come up with an appealing and robust mechanism for the identification of variable structure systems.

REFERENCES

1. H. J. J. Uyttenhove, *Systems Approach Problem Solver* (*SAPS*): An Introduction and Guide. Computing and Systems Consultants, Inc., P.O. Box 1551, Binghamton, NY 13902, 1979.
2. H. J. J. Uyttenhove, "SAPS—A software system for inductive modelling." In: *Simulation and Model-Based Methodologies: An integrative View,* edited by T. I. Ören, B. P. Zeigler and M. S. Elzas, Springer-Verlag, Series F: *Computer & System Sciences,* **10,** 1984, pp. 427–449.
3. D. W. Yandell, *SAPS-II, Raw Data Analysis in CTRL-C, User's Manual and Progress Report.* Senior Project, Department of Electrical and Computer Engineering, The University of Arizona, Report: CERL 85-07, 1985.
4. Systems Control Technology, Inc., *CTRL-C A Language for the Computer-Aided Design of Multivariable Control Systems.* Systems Control Technology, Inc., Palo Alto, CA 94304, 1984.
5. A. Alali, *Random Number Generation and Distribution Fitting with an Interface to the CTRL-C System.* Senior Project, Department of Electrical and Computer Engineering, The University of Arizona, Report: CERL 86-03, 1986.
6. Aptech Systems, Inc., *GAUSS, Programming Language Manual.* Aptech Systems, Inc., P.O. Box 6487, Kent, WA 98064, 1986.
7. G. J. Klir, *Architecture of Systems Problem Solving,* Plenum Press, New York, 1985.
8. F. E. Cellier, "Qualitative simulation of technical systems using the general system problem solving framework." *Inter. J. General Systems,* **13,** 4, 1987, pp. 333–344.
9. J. W. Forrester, *World Dynamics.* Wright-Allen Press, Cambridge Mass., 1971.
10. C. Moler, *MATLAB User's Guide.* Department of Computer Science, University of New Mexico, Albuquerque, NM, 1980.



Dr François E. Cellier received his B.S. in Electrical Engineering from the Swiss Federal Institute of Technology (ETH) Zürich in 1972, his M.S. in Automatic Control in 1973, and his Ph.D. in Technical Sciences in 1979, all from the same university. Following his Ph.D., Dr. Cellier worked as a Lecturer at ETH Zürich. He joined the University of Arizona in 1984 as an Associate Professor. Dr Cellier's main scientific interests concern modelling and simulation methodology, and the design of advanced software systems for simulation, computer-aided modelling, and computer-aided design. He has designed and implemented the GASP-V simulation package, and he was the designer of the COSY simulation language which has meanwhile become a standard by the British Ministry of Defence. Dr Cellier has authored or co-authored more than thirty technical publications, and he has edited two books. He served as a chairman of the National Organizing Committee (NOC) of the Simulation '75 conference, and as a chairman of the International Program Committee (IPC) of the Simulation '77 and Simulation '80 conferences, and he has also participated in several other NOC's and IPC's. He is associate editor of several simulation related journals, and he served as vice-chairman of two committees on standardization of simulation and modelling software. Memberships include SCS and IMACS.

David W. Yandell is currently a Member of the Technical Staff at Hughes Aircraft Company in Tucson, Arizona. After six years of service in the U.S. Navy working with nuclear propulsion systems in submarines, he graduated in 1985 with a bachelor's degree in electrical engineering from the University of Airzona. He has completed some graduate level courses at the University of Arizona. Mr Yandell is also a member of Tau Beta Pi.