CHAPTER 3

General System Problem Solving Paradigm for Qualitative Modeling

François E. Cellier

Abstract

In this chapter qualitative modeling is applied to inductively reason about the behavior of physical systems. Inductive reasoning does not dwell on the principles of "naive physics" as the commonsense reasoning does, but rather implements a sort of pattern-recognition mechanism. The basic differences between *inductitive reasoning* and *commonsense reasoning* are explained. It is shown under which conditions either of the two approaches may be more successful than the other.

1. Introduction

What is qualitative simulation as opposed to quantitative simulation? Let me begin by putting some commonly quoted myths about qualitative simulation to the sword.

> Qualitative simulation is cheaper than quantitative simulation. If quantitative simulation, e.g. in a real-time situation, cannot produce the results fast enough, qualitative simulation may be the answer to the problem.

Algorithms used for qualitative simulation are by no means faster than those used in quantitative simulation. In qualitative simulation, there are generally plenty of alternative branches to be explored, whereas quantitative simulation usually produces one individual trajectory. Thus, quantitative simulation is normally faster than qualitative simulation if applicable. Thus, if your quantitative real-time simulation executes too slowly, do not go to qualitative simulation; go to a nearby computer store and buy yourself a faster computer.

> Qualitative simulation requires a less profound understanding of the mechanisms that we wish to simulate. Therefore, if we don't fully understand the mechanisms that we wish to simulate, quantitative simulation is out of the question, whereas qualitative simulation may still work.

Wrong again! Qualitative simulation has as stringent constraints as quantitative simulation; they are just a little different. It is a convenient user interface that relieves the user from some of the intricacies of detailed understanding of the simulation mechanisms and not the modeling methodology per se. Today's languages for quantitative simulation are very user friendly, more so than today's languages for qualitative simulation. This is due to the fact that quantitative simulation languages have been around for much longer. Thus, if you do not understand what you are doing, do not go to qualitative simulation; go to an expert who does.

There are today at least three different methodologies around that are all advocated under the name "qualitative modeling" and/or "qualitative simulation." One of them uses the so-called "naive physics" approach. Its original proponents were mostly found among the social sciences, and it is in those circles where the second of the above propositions is frequently heard (e.g., [1]). Today, this school of thought is mostly found among the artificial intelligence experts (e.g., [4, 5, 7]). Since none of those above is represented in this book directly, I shall briefly introduce the formulation of Kuipers [7], which seems the most advanced among these types of qualitative simulation mechanisms.

The second approach is usually referred to as *commonsense reasoning* (this term is sometimes also used by the "naive physicists") and originates from stochastic signal processing. Whereas the former approach dwells on incomplete knowledge about system parameters, this approach dwells on the signals (trajectories) produced by these models and takes into consideration the uncertainties in the obtained data. Many of the more recent results in this arena are derived from *fuzzy logic*. The most prominent advocate for these types of qualitative models is Zadeh [13–16]. Since these results are being discussed in the chapter by George Klir, I shall refrain from discussing these results within my chapter as well.

The third and last approach is called *inductive reasoning* and originates from general system theory. One of its foremost proponents is Klir [6]. It is this approach that will be explained in most detail in this chapter.

2. The Quantitative Approach

We are going to analyze a simple linear continuous-time single-input/single-output system of the type

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b} \cdot u,$$

$$y = \mathbf{c}' \cdot \mathbf{x} + d \cdot u,$$

where the system matrices $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}'$, and d are represented in controller-canonical form, namely,

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -c & -b & -a \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot u,$$

$$y = (1 \quad 0 \quad 0) \cdot \mathbf{x} + (0) \cdot u,$$

where a, b, and c are three unknown but positive parameters.

In quantitative simulation (and in control theory) this is called a *state-space representation* of the system. Most quantitative simulation languages (but not all) will require these equations to be written individually, that is, in the form:

$$\dot{x}_1 = x_2,$$

$$\dot{x}_2 = x_3,$$

$$\dot{x}_3 = -c \cdot x_1 - b \cdot x_2 - a \cdot x_3 + u,$$

$$y = x_1,$$

which can be represented through the *block diagram* shown in Figure 3.1. We executed several quantitative simulation runs with different values for the three parameters. The results of these simulations are presented in Figure 3.2. It becomes evident that this system exhibits at least four qualitatively different modes of operation: exponential decay, damped oscillation, undamped oscillation, and excited oscillation.

We can look at the system analytically. The characteristic polynomial of this system is

$$det(\lambda \cdot \mathbf{I}^{(n)} - \mathbf{A}) = \lambda^3 + a \cdot \lambda^2 + b \cdot \lambda + c = 0.0.$$

The roots of the characteristic polynomial are the eigenmodi of the system. We can analyze the stability of the system, for example, by setting up a Routh–Hurwitz scheme:

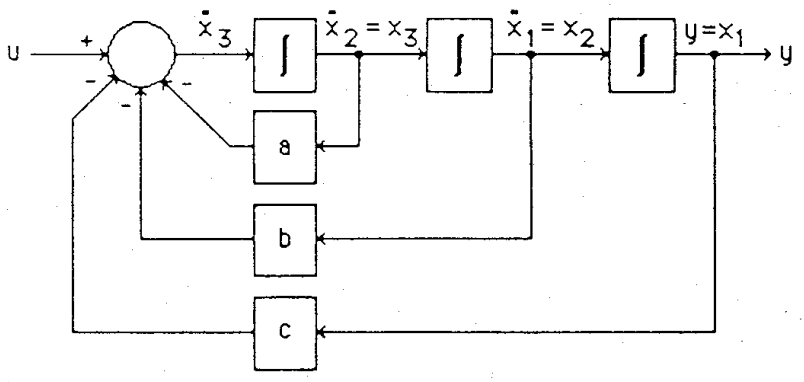

FIGURE 3.1. Block diagram for quantitative simulation.

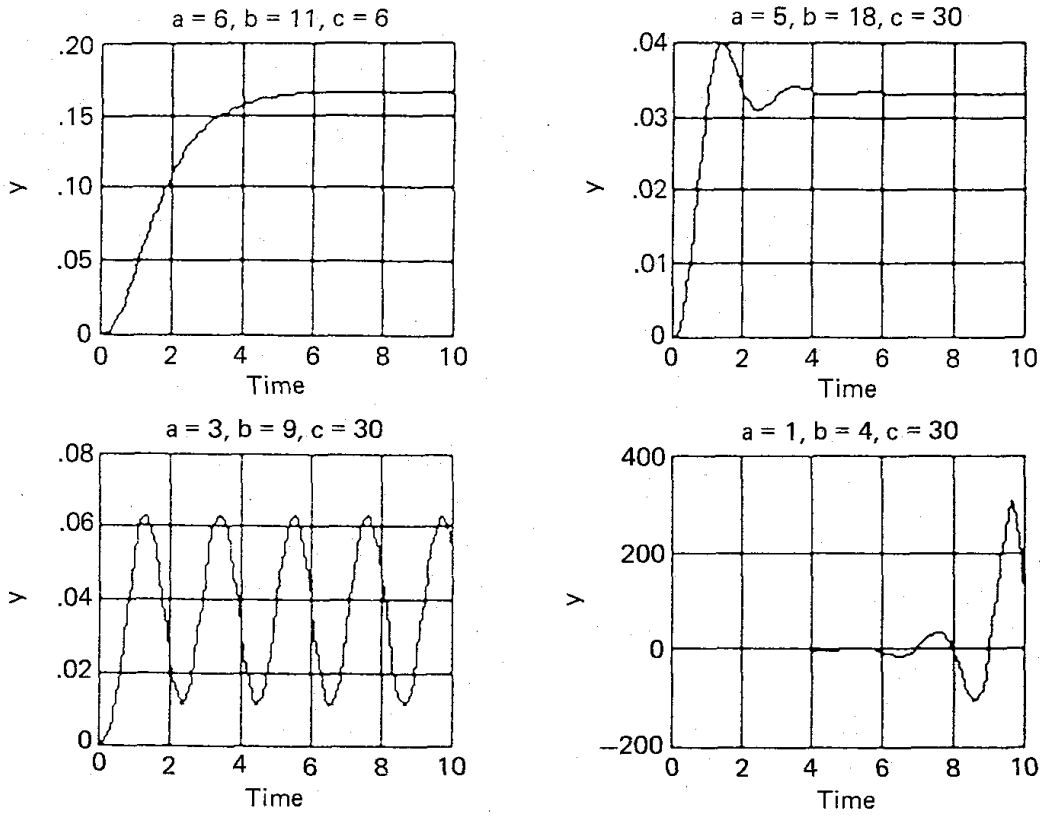

FIGURE 3.2. Results from quantitative simulation.

s^3	1	b
s^2	a	c
s^1	$\frac{c-ab}{a}$	
s^0	c	

If all the elements in the first column of the Routh–Hurwitz scheme are positive, the system is stable, that is, all three roots are in the left half λ-plane. Otherwise, the number of sign changes in the first column of the Routh–Hurwitz scheme determines the number of unstable poles, that is, the poles in the right half λ-plane. It can be seen that, if $c > a \cdot b$, all three poles are in the left half plane. They can all be located on the negative real axis (exponential decay), or one can be located on the negative real axis while the other two form a conjugate complex pole pair (damped oscillation). If $c = a \cdot b$, one pole is still on the negative real axis, while the two dominant poles are now on the imaginary axis itself (undamped oscillation). If $c < a \cdot b$, one pole is still on the negative real axis, while the other two poles form a conjugate complex pole pair in the right half plane (excited oscillation). These are indeed

all possible cases (for positive coefficients a, b, and c), as the following analysis shows.

Consider the case of the excited oscillation with

$$\lambda_1 = -x,$$
$$\lambda_2 = y + j \cdot z,$$
$$\lambda_3 = y - j \cdot z.$$

In this case, we can write the characteristic polynomial as

$$(\lambda + x) \cdot (\lambda - y - j \cdot z) \cdot (\lambda - y + j \cdot z) = 0.0,$$

which can be rewritten as

$$\lambda^3 + (x - 2y)\lambda^2 + (y^2 + z^2 - 2xy)\lambda + x(y^2 + z^2) = 0.0,$$

which has positive coefficients iff

$$a = x - 2y > 0.0,$$
$$b = y^2 + z^2 - 2xy > 0.0,$$
$$c = x(y^2 + z^2) > 0.0,$$

whereby x, y, and z are all positive. The third condition is obviously always satisfied. The first condition can be rewritten as

$$x > 2y$$

or

$$2xy > 4y^2,$$

which we can plug into the second condition:

$$y^2 + z^2 > 2xy > 4y^2;$$

that is, in the borderline case

$$z^2 = 3y^2,$$

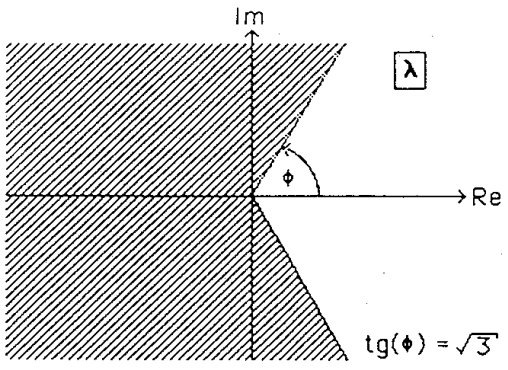

FIGURE 3.3. Domain of possible pole locations.

and therefore,

$$z = \sqrt{3} \cdot y.$$

This is shown in Figure 3.3, in which the possible pole locations for positive coefficients a, b, and c are shredded. Therefore, it was verified that this system does not exhibit any other modes of operation than the four modes shown in Figure 3.2.

3. The Naive Physics Approach

The naive physics approach has been designed to deal with exactly the type of situation that we are faced with here: a system that is structurally completely defined, but which contains a set of parameters (such as our parameters a, b, and c), the values of which are not totally determined.

Numerical mathematicians and quantitative simulationists always express their equations such that they can *integrate* signals rather than *differentiate* them (since the numerical properties of integration are much more benign than those of differentiation, the naive physicists traditionally prefer the differentiation operator, probably because naive physics evolved from the analytical physics, and not from the numerical physics).

We can easily transform our set of equations into the desired form, by solving for the x_i instead of the $\dot{x}_i$ variables

$$x_1 = -\frac{b}{c}\dot{x}_1 - \frac{a}{c}\dot{x}_2 - \frac{1}{c}\dot{x}_3 + \frac{1}{c}u,$$

$$x_2 = \dot{x}_1,$$

$$x_3 = \dot{x}_2,$$

$$y = x_1,$$

which can be represented through the block diagram shown in Figure 3.4. Kuipers [7] decomposes these equations into a set of primitive equations, for example, of the form

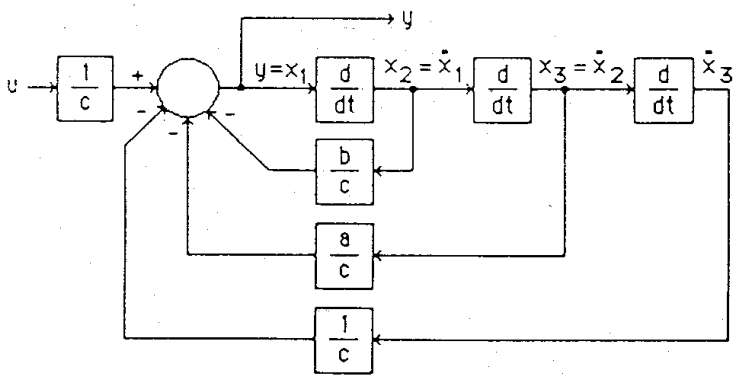

FIGURE 3.4. Block diagram using the differentiation operator.

$$H_1 = -\frac{b}{c}\dot{y},$$

$$H_2 = -\frac{a}{c}\dot{x}_2,$$

$$H_3 = -\frac{1}{c}\dot{x}_3,$$

$$H_4 = \frac{1}{c}u,$$

$$H_5 = H_1 + H_2,$$

$$H_6 = H_3 + H_4,$$

$$y = H_5 + H_6,$$

$$x_2 = \dot{y},$$

$$x_3 = \dot{x}_2.$$

At this point, we can eliminate the unknown parameters by replacing the exact first four equations by their qualitative counterparts

$$H_1 = M^-(\dot{y}),$$

$$H_2 = M^-(\dot{x}_2),$$

$$H_3 = M^-(\dot{x}_3),$$

$$H_4 = M^+(u),$$

where M^- stands for any monotonically decreasing function of the input argument and M^+ stands for any monotonically increasing function of the input argument. Using predicate logic, the nine equations can finally be represented as

$$M^-(x_2, H_1),$$

$$M^-(x_3, H_2),$$

$$M^-(\dot{x}_3, H_3),$$

$$M^+(u, H_4),$$

$$ADD(H_1, H_2, H_5),$$

$$ADD(H_3, H_4, H_6),$$

$$ADD(H_5, H_6, y),$$

$$DERIV(y, x_2),$$

$$DERIV(x_2, x_3),$$

$$DERIV(x_3, \dot{x}_3),$$

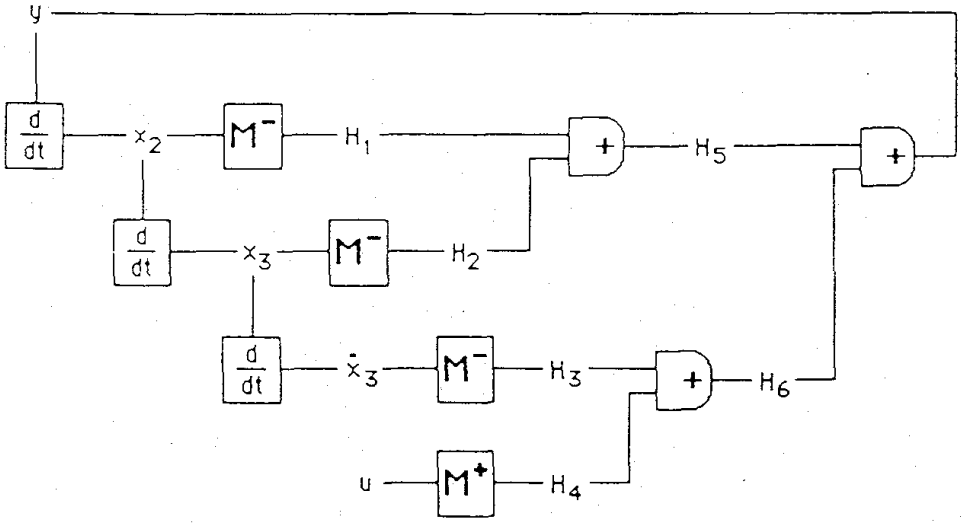

FIGURE 3.5. Kuiper's diagram.

which are ten equations (which Kuipers calls "constraints") in eleven variables (which Kuipers calls "parameters"). Kuipers [7] also uses his own form of block diagram as depicted in Figure 3.5.

4. The Constraint Propagation Approach

At this point, we are ready to perform a qualitative simulation. How this works is explained in detail in [7]. The system starts off from the initial condition (assuming that the initial condition is known) and performs one simulation step by considering all possible state transitions from the initial condition. Many (hopefully most) of these transitions will be in contradiction with one or several of the equations (constraints) and can therefore be rejected. This process of pruning is what is commonly referred to as "constraint propagation." Each state that cannot be rejected will be considered a new initial point for the next step of the simulation to be performed. If several states are possible as the outcome of one simulation step, this is considered a bifurcation point. Each branch will lead to one simulation thereafter. The simulation terminates under any of the following conditions:

1. The next system state is exactly the same as the current system state. In this case, the system has become *quiescent.*
2. The next system state is exactly the same as a previously found system state. In this case, the system has become *periodic.*
3. During the next system state, a variable becomes infinite. In this case, the system has become *divergent.*

Hopefully, there are not too many bifurcation points in the simulation since, otherwise, not much can be said about the behavior of the system.

Kuipers [7] proved that every type of behavior that is physically possible will be found by his QSIM algorithm. However, it may happen that the qualitative simulation suggests additional modes of operation that are not physically possible. This means that, without even running the example, we know for sure that QSIM will at least find four different modes of operation, namely, those that were shown in Figure 3.2.

Unfortunately, this does not tell us very much. It is almost as stating that tomorrow the weather will be the same as today, or it may change. The problem with this "simple" example is that, although it is simple from a mathematical point of view, it is not simple from a physical perspective. There are tight feedback loops in this model that create a close interaction between the various state variables, and therefore, relatively small changes in parameter values may lead to a qualitatively different pattern of overall system performance.

The "naive physics" approach works particularly well for simple physical phenomena, the basic functioning of which an untrained human observer can easily understand in qualitative terms. Feedback loops make most problems intractable for the untrained human observer and thereby defeat also the "naive physics" type of qualitative simulator.

It is this strong correlation between the types of problems that humans can analyze in qualitative terms and the types of problems that naive physics can handle that made several of the advocates of this technique proclaim that "naive physics is the way how humans go about qualitative problem solving." I personally doubt that this statement is correct and suggest that the similarity between the two types of behaviors is more of an analogy than of a homomorphism. The cause of my skepticism will be explained right away.

I have a dog who loves to play ball. I kick the ball with the side of my foot (I usually wear sandals, and a straight kick hurts my toes), and my dog runs after the ball as fast as he can. I was able to observe the following phenomenon. If I place my foot to the left of the ball, my dog will turn to the right to be able to run after the ball as soon as I hit it. He somehow *knows* that the ball will be kicked to the right. If I now change my strategy and place my foot to the right of the ball, my dog immediately swings around to be ready to run to the left. He obviously has some primitive understanding of the mechanics involved in ball kicking. However, I assure you that I never let my dog near my physics books, and thus, he had no opportunity to study Newton's laws—not even in their naive form.

I believe that my dog knows the rules of "naive physics" without knowledge of the system structure, simply as a phenomenon of *pattern recognition.* The human brain (and to some extent also the dog brain) is great at recognizing patterns that have been experienced before in a similar form. Unfortunately, our von Neumann computer architecture is very bad at reproducing this capability. It takes massive *parallel processing,* for example, in the form of a *neural network* to reproduce something faintly similar to my dog's (but not yet my own) capability of pattern recognition.

However, even if "naive physics" provides us only with an analog and not with a homomorphism to the human problem-solving approach, the technique works well when applied to the right problem. The type of problems for which the "naive physics" approach works well are characterized by the following properties:

1. The physical structure of the problem is well defined. Only some of the parameter values are incompletely known.
2. The physical structure of the problem is rather simple.
3. Subsystems are loosely coupled.
4. There are few or no feedback loops involved, and if there are feedback loops in the system, they do not dominate the system behavior.

5. An Induction-Based Approach

This time, we want to start from a completely different premise. Before, it was assumed that the structure of the system under study is known except for the numerical values of some of its parameters. This time, we want to assume that we know little to nothing about the structure of the system. The system is a "black box." All we have is a set of observations of inputs and outputs. The question now is, Can we qualitatively forecast the behavior of this system for other types on inputs for which we did not previously observe the system behavior? While the former situation could be interpreted as a *parameter estimation* problem, the latter presents itself as a *structure identification* problem.

While the goal of the previous type of qualitative simulation was to replicate the way humans reason about physical systems of which they have a basic qualitative understanding, the goal of the new type of qualitative modeling is to *learn* the behavior of an unknown system from observation. Maybe, this approach is closer to what my dog went through when he "learned" the basic properties of ball kicking.

6. Quantitative Simulation for the Purpose of Fact Gathering

Let us analyze the simple linear continuous-time single-input/multi-output system

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -3 & -4 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot u,$$

$$\mathbf{y} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \cdot u,$$

which is one particular system in the class of systems that we discussed before except for the fact that we now measure multiple outputs. The trick is that we are going to use this structure exclusively to obtain trajectory behavior using *quantitative simulation*, as if we were recording (measuring) the input/output behavior of a physical black-box system. Thereafter, we shall immediately forget all that we know about the system structure. From then on, we shall work with the recorded trajectories exclusively and see whether we can create a *qualitative model* that is able to predict the behavior of this unknown system qualitatively for an arbitrary input sequence.

We shall discretize the time axis by adequately *sampling* the three-state variables over the simulation domain. We shall then *recode* the continuous variables into a set of distinct levels using the interval mapping technique; that is, if the continuous variable is within a certain interval, it will be mapped into a discrete value. If we use, for example, five levels for a particular variable, say, a temperature reading, we can map the continuous domain of recorded temperatures into the set of integers $\{1, 2, 3, 4, 5\}$ that can be interpreted as "very cold," "cold," "moderate," "hot," and "very hot." These are what Kuipers [7] calls *landmarks*.

Selection of an adequate sampling rate and an appropriate set of landmarks is crucial to the success of the endeavor, and we shall discuss how good values for these quantities can be determined.

Thereafter, we shall discuss how we can use the recoded measurement data to come up with a qualitative model with *optimized forecasting power*.

In order to retrieve as much information from the system as possible, we decided to excite the system with a stochastic binary input signal; that is, the input is either "high" or "low," and the transitions between the two possible states are chosen at random [2].

7. The General System Problem Solving (GSPS) Framework

General System Problem Solving (GSPS) [6] is a methodological framework arising from General Systems Theory that allows the user to define and analyze types of systems problems. In this methodology, systems are defined through a hierarchically arranged set of epistemological subsystems. Forecasting and reconstruction analysis capabilities are two examples of the capabilities of the GSPS methodological tools. An on-line monitoring system can be implemented in the GSPS framework by using its inductive inference capability to imitate the human learning process.

SAPS-II [3] is software coded at the University of Arizona that implements the basic concepts of the GSPS framework. SAPS-II has been implemented as an application function library to the control systems design software CTRL-C [9]. In terms of common artificial intelligence terminology, we can say that SAPS-II employs CTRL-C as an artificial intelligence shell.

GSPS or analysis through General Systems Theory starts by defining a region in the universe where the system and the observer coexist and interact. A system in this context can be interpreted as a set of relations between some objects that belong to that region of the universe and in which the observer is interested.

Therefore, the first step to problem solving, or analysis, is the definition of the system: What is it that is of interest to us concerning the problem under study? A set of variables to represent the system has to be chosen, and this set is to be classified into *input variables* and *output variables*, which is a natural classification of the variables: Input variables depend on the environment and control the output variables.

8. The Epistemological Hierarchy

The GSPS framework is a hierarchically arranged set of epistemological subsystems. Starting at level zero, the amount of knowledge in the systems increases as we climb up the epistemological ladder. The lower level subsystems are contained in the ones that are at higher epistemological levels.

At the lowest epistemological level, we find the *source system*, which represents the system as it is recognized by the observer. The amount of information present at this level represents the basic description of the problem in which the observer is interested: which variables are relevant to the problem, what causal relationships are present among them (which are inputs and which are outputs to the system), and which are the states these variables can possibly assume along their time-history. The number of states, or levels, that each variable can potentially assume is essentially problem dependent. It should be kept as low as possible without unacceptable loss of information.

The next epistemological level in the hierarchy is represented by the *data system*. It includes the source system and, additionally, the time-history of all the variables of interest.

Yet one epistemological level higher, we find the *behavior system*, which holds, besides the knowledge inherent to both, source and data systems, a set of time-invariant relationships among the chosen variables for a given set of initial or boundary conditions. Behavior systems can be considered basic cells for yet higher epistemological levels, the so-called *structure systems*, which we shall not, however, discuss in this treatise.

The time-invariant relationships among the variables are *translation rules* mapping these variables into their common spaces. They can be used to generate new states of the variables within the time span defined in the Data Model, providing in this way an *inductive inference* capability in the methodology. Due to this characteristic, *behavior systems* are sometimes also referred to as *generative systems*.

9. The Concept of a Mask

A Data Model in the GSPS framework is an $n_{rec} \times n_{var}$ matrix, where n_{rec} is the number of recordings (data points) collected in the time span covered by the Data Model, and n_{var} is the number of variables present in the model. This is a matrix representation of the time-history of the system, and the convention is that time increases from the top to the bottom of the matrix.

A mask is a matrix representation of a translation rule for a given Data Model; hence, it is a matrix representation of a Behavior Model of the system. The dimensions of the mask are $(d + 1) \times n_{var}$, where d is the depth of the mask representing the number of sampling time units covered by the mask.

The active elements of a mask are called *sampling variables* and represent the variables that are to be considered in the translation rule associated with the time instant they occur.

Generative masks include in their structure the notion of *causality* among the variables. Elements of a generative mask are zero, negative, or positive, meaning "neutral element," "generating element," and "generated element," respectively. For example, a generative mask like

$$\begin{array}{c} \\ t-2\Delta t \\ t-\Delta t \\ t \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ -2 & 0 & 0 & -3 & 0 \\ 0 & -4 & 0 & 0 & +1 \end{pmatrix} \end{array}$$

corresponds to the translation rule

$$v_5(t) = f(v_3(t - 2\Delta t), v_1(t - \Delta t), v_4(t - \Delta t), v_2(t)),$$

where $v_i(\tau)$ is the state assumed by the variable v_i at time $t = \tau$. Within one set of sampling variables, for example, the inputs, the numbering sequence is immaterial. We chose to number them from the left to the right and from the top to the bottom.

10. The Sampling Interval

Note that the Behavior Model above takes samples of the Data Model at every Δtth data point to predict the state of v_5. Hence, Δt is the sampling interval t_s of the collected data set. There is not a precise way of determining the most effective sampling interval to be used, but a good rule of thumb is that the mask should cover the dynamics of the slowest mode in the model [2]. In the case of the given example, the mask has depth 2, and the sampling interval Δt should then be about half of the slowest time constant of the model. In our case, the slowest time constant was found to be approximately 3 time units. Accordingly, we select the sampling period to be 1.5 time units. Experimentation with different sampling periods verified this to be a good choice.

Notice, however, that, as outlined in [2], selection of an appropriate sampling rate is absolutely crucial to the success of our endeavor; thus, careful experimentation with this parameter is indicated under all circumstances.

11. Converting Quantitative Data into Qualitative Data

In order to be able to reason qualitatively about the behavior of our system, we need to convert the "measured" *quantitative data* (i.e., continuous variables) into *qualitative data* (i.e., variables of an enumerated type). GSPS calls this process the *recoding* of the measurement data. SAPS-II provides for various algorithms to recode *real* variables into sets of *integers*.

Due to the type of the quantitative simulation experiment, we notice that the control input u is already a binary variable and, therefore, does not require any further recoding.

The three output variables y_1, y_2, and y_3, however, are truly continuous variables, and an appropriate selection of the recoding procedure will decide over success or failure of our endeavor. The number of recoding levels to be used for each variable has, intuitively, to be odd if we want to have a "normal" range of operation and variations about it.

The choice of the appropriate number of levels (landmarks) is a somewhat problematic issue. There is always a conflict between the demands of simplicity for the purpose of a strong forecasting power, and an improved resolution for the purpose of a strong expressiveness of the model. Recoding each variable into one level only results in an infinitely "valid" model with no expressiveness whatsoever. On the other hand, recoding each variable into a high number of levels will result in a highly "expressive" model with little to no forecasting power. We decided to code each of these variables into three levels: "low," "medium," and "high."

How should the interval boundaries be chosen? It seems intuitively most appealing to request each "class" (range) to contain approximately the same number of "members" (samples). This can best be achieved by sorting each output variable separately (using the standard CTRL-C *sort*-function), thereafter split the resulting vector into three subvectors of equal size, and determine appropriate elements for the *from*-matrix used in the recoding by looking at the first and last elements of each subvector. A SAPS-II procedure implementing this algorithm was presented in [12]. However, for our simple demonstration problem, we got very good results with a much simpler algorithm, namely, by subdividing the interval between the lowest ever recorded value and the highest ever recorded value of each variable into three subintervals of equal size.

One last parameter still needs to be decided on, namely, the number of recordings that we need for our GSPS analysis. From classical statistical techniques, we know that each "class" (i.e., each possible state) should contain at least five "members" (i.e., should be recorded at least five times) [8].

Therefore, if n_{var} denotes the number of variables, and if n_{lev_i} denotes the number of levels assigned to the variable v_i after recoding, we can write down the following (optimistic) equation for the minimum necessary number of recordings (n_{rec})

$$n_{\text{rec}} = 5 \prod_{i=1}^{n_{\text{var}}} n_{\text{lev}_i};$$

that is, in our case,

$$n_{\text{rec}} = 5 * 2 * 3 * 3 * 3 = 270.$$

Thus, the number of recordings needed depends strongly on the number of levels chosen for each variable. On the other hand, if the number of available data points is given, this will decide on the maximum number of levels that each continuous variable can be recoded into. For our little demonstration problem, we got good answers with a considerably smaller number of recordings, namely, 100.

12. The Optimal Mask Analysis

Given a Data Model, any topologically compatible mask associated with it is "valid" since it denotes a representation of a relationship among the sampling variables it contains. The questions now are, "How *good* is the mask?" and "How valid is the translation rule it represents?" There are numerous possible masks that can be written for one set of variables, and it is desirable to determine among all possible masks the one that shows the least uncertainty in its generating capability, that is, the one that maximizes the forecasting power. This is exactly what the *optmask*-function of SAPS-II evaluates. The measure of uncertainty that is currently employed by this function is the Shannon entropy.

SAPS-II requests the user to specify a *mask candidate matrix* that contains the element -1 for potential generating elements (potential input), the element 0 for neutral elements (do not care variables), and $+1$ for generated elements (outputs) of the optimal mask.

In our example, the data model contains four variables, namely, the input u and the three outputs y_1, y_2, and y_3. Consequently, any valid mask must have exactly four columns, one for each variable. We want to assume the depth d of the mask to be 2, and therefore, all masks that we consider have exactly three rows.

We want to assume that concurrent states of the outputs do not affect each other, whereas the input variable may affect any of the outputs instantaneously. It seems intuitively evident that more information can be extracted from the measured trajectories if each output variable is treated independently, that is, if a separate optimal mask is generated for each of the output variables. The following set of mask candidate matrices was therefore used for

the optimal mask evaluation:

$$Mcnd_1 = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & +1 & 0 & 0 \end{pmatrix},$$

$$Mcnd_2 = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 0 & +1 & 0 \end{pmatrix},$$

$$Mcnd_3 = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & +1 \end{pmatrix}.$$

The $Mcnd_1$ matrix determines that $y_1(t)$ may be a function of up to nine different variables, namely, $u(t-2\Delta t)$, $y_1(t-2\Delta t)$, $y_2(t-2\Delta t)$, $y_3(t-2\Delta t)$, $u(t-\Delta t)$, $y_1(t-\Delta t)$, $y_2(t-\Delta t)$, $y_3(t-\Delta t)$, and finally $u(t)$. The other two mask candidate matrices can be interpreted accordingly. It is the task of the optimal mask analysis to determine which of these potential influencing variables are relevant; that is, it will identify the simplest models that allow one to forecast the behavior of the outputs in a reasonably accurate fashion from any set of given data.

The optimal mask analysis performs an exhaustive search on all masks that are structurally compatible with the mask candidate matrix. The search starts with the simplest masks, that is, with masks that contain as few nonzero elements as the mask candidate matrix permits. In our example, the simplest masks are those of *complexity two*, that is, masks that have exactly one input and one output. For each of the three mask candidate matrices, there exist exactly nine structurally compatible masks of complexity two.

Each of the possible masks is compared to the others with respect to its potential merit. The optimality of the mask is evaluated with respect to the maximization of its forecasting power. The *Shannon entropy measure* is used to determine the uncertainty associated with the forecasting of a particular output state given any feasible input state.

The Shannon entropy relative to one input is calculated from the equation

$$H_i = -\sum_{\forall o} p(o|i) \cdot \log_2(p(o|i)),$$

where $p(o|i)$ is the conditional probability of a certain output state o to occur, given that the input state i has already occurred. The term *probability* is meant in a statistical rather than in a probabilistic sense. It denotes the quotient of the observed frequency of a particular state divided by the highest possible frequency of that state.

The overall entropy of a mask is then calculated as the sum

$$H_m = \sum_{\forall i} p_i H_i,$$

where p_i is the probability of that input to occur. The highest possible entropy

H_{max} is obtained when all probabilities are equal, and a zero entropy is encountered for relationships that are totally deterministic.

A normalized overall *entropy reduction* H_r is then defined as

$$H_r = 1.0 - \frac{H_m}{H_{max}}.$$

H_r is obviously a real number in the range between 0.0 and 1.0, where higher values usually indicate an *improved forecasting power*. The *optimal mask* among a set of mask candidates is defined as the one with the highest entropy reduction.

The algorithm then proceeds to higher levels of complexity. In our example, there exist exactly 36 masks of complexity three (i.e., masks with two inputs and one output) for each of the three mask candidate matrices. These can be compared with each other for the determination of the optimal mask of complexity three. Thereafter, we can proceed to even higher degrees of complexity. In our example, the highest possible degree of complexity is 10, and there exists exactly one mask of complexity 10 for each of the three mask candidate matrices.

Masks at different complexity levels are somewhat more difficult to compare to each other. Obviously, with increasing complexity, the masks tend to give the impression of a more and more deterministic behavior. Since, with increasing mask complexity, the number of possible input states (the *possible input state set*) grows larger and larger, chances are that more and more input states in the data model are observed exactly once, which makes them look completely deterministic, whereas many other possible input states are never observed at all, a fact that does not show up in the entropy reduction measure. In the case of the ultimately complex mask, that is, a mask of depth $n_{rec} - 1$ and complexity $n_{rec} \times n_{var}$, the *observed input state set* consists of exactly one sample, whereas the *possible input state set* is extremely large. Thus, the entropy reduction measure will have a value of 1.0, and yet, the forecasting power of this mask is negligibly small.

For this reason, Uyttenhove proposed [10] the following complexity weighting factor C_m

$$C_m = \frac{n_{var} \cdot d_{act} \cdot n_{compl}}{d_{max}},$$

where n_{var} is the number of variables in the source model, d_{act} is the actual depth of the mask plus one, n_{compl} is the number of nonzero entries in the mask, and d_{max} is the maximum possible depth the mask could have (the depth of the chosen mask candidate) plus one.

For example, the complexity weighting factor of the mask

$$mask = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & -2 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix}$$

can be evaluated to

$$C_m = \frac{4 \times 2 \times 4}{3} = 10.667.$$

Finally, the *mask quality measure* Q is defined as

$$Q = \frac{H_r}{C_m},$$

and that is how masks of different complexity are being compared to each other. Clearly, the above formula is strictly heuristic, and we are currently experimenting with improved formulae. We now believe that what should be punished is not the complexity of a mask, but its inability to make the *observed input state set* decently represent the *possible input state set*. We therefore experiment with the following *completeness weighting factor* F_c

$$F_c = \frac{n_1 + 2 \cdot n_2 + 3 \cdot n_3 + 4 \cdot n_4 + 5 \cdot n_5}{5 \cdot n_{\text{poss}}},$$

where n_1 is the number of input states that have been observed exactly once, n_2 is the number of input states that have been observed exactly twice, n_3 is the number of input states that have been observed exactly thrice, n_4 is the number of input states that have been observed exactly four times, n_5 is the number of input states that have been observed five times or more, and n_{poss} is the possible input state set. This formula is based on the statistical rule that, in a subinterval or class analysis, each class member should be observed at least five times [8].

Using the completeness weighting factor, we redefine the *mask quality measure* as

$$Q = F_c \cdot H_r.$$

If every possible input state is observed at least five times, F_c assumes a value of 1.0.

We applied this algorithm to our example problem. By repeating the optimal mask analysis several times using different random number streams for the input, it was determined that the set of optimal masks for this problem is

$$Mask_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -2 \\ 0 & +1 & 0 & 0 \end{pmatrix},$$

$$Mask_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & -2 & 0 & 0 \\ -3 & 0 & +1 & 0 \end{pmatrix},$$

$$Mask_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -2 & 0 & 0 & +1 \end{pmatrix};$$

that is,

$$y_1(t) = f_1(u(t - \Delta t), y_3(t - \Delta t)),$$
$$y_2(t) = f_2(u(t - \Delta t), y_1(t - \Delta t), u(t)),$$
$$y_3(t) = f_3(u(t - \Delta t), u(t)).$$

The details of the experiments performed to verify the validity of these optimal masks are described in [2].

13. Qualitative Inference Through Inductive Reasoning

These optimal masks can now be used to generate state-transition matrices that show the dependence of the generated elements (outputs) on the generating elements (inputs). Notice that the meaning of the words *input* and *output* is now different from before. In earlier paragraphs, these words referred to the one "input" and the three "outputs" of the physical model. Now, we talk about the "inputs" and the "outputs" of the Behavior Models. For example, the second Behavior Model has three "inputs," namely, $u(t - \Delta t)$, $y_1(t - \Delta t)$, and $u(t)$, and one "output," namely, $y_2(t)$.

Using these state-transition matrices, we can forecast the system behavior by simply looping through state transitions for any physical input sequence.

We performed the following experiment: We actually simulated the quantiative model over 200 communication intervals, that is, over a duration of 300 time units. We recoded the three continuous output variables over the entire time span. However, we thereafter used only the first 150 time units for the generation of the optimal masks. Now, we used the optimal masks found on the basis of the first 150 time units to qualitatively predict (forecast) the behavior of the system over the next 15 time units (corresponding to 10 steps) using the same input sequence as for the quantitative simulation. The following matrices compare the "measured" (i.e., quantitatively simulated) time-history to the "predicted" (i.e., qualitatively simulated) time-history:

$$Meas = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 1 & 0 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \end{bmatrix}, \qquad Pred = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 1 & 0 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \end{bmatrix}.$$

These two matrices contain excerpts of the data model. As before, the four columns represent the input u and the three outputs y_1, y_2, and y_3. The rows are recordings of these variables at various instances of time: $t = [150, 151.5, 153, \ldots, 173.5, 175]$. The *Meas* matrix contains the (recoded) data found during the quantitative simulation, while the *Pred* matrix contains the forecast outputs using qualitative simulation given the same past history and the same input data stream as in the quantitative case.

As can be seen, there was not a single forecasting error in this sequence. We repeated the experiment for other input sequences and found that, on the average, the forecasting would exhibit about two incorrect entries per 10 time steps, that is, the probability of correct forecasting of a value was roughly 28 out of 30, or 93%. The details of these experiments can be found in [2].

These results encouraged us to try our methodology on a much more involved system, namely, a B747 airplane in high-altitude horizontal flight. The results of that analysis were presented in [11] and [12].

14. Discussion of Results

Similar to the discussion of the "naive physics" approach, we want to analyze the conditions that must be satisfied for the "inductive reasoning" approach to be successful. Here are our findings:

1. Contrary to the naive physics approach, inductive reasoning can operate on systems, the structure of which is not completely known.
2. Inductive reasoning therefore works well in application areas, such as biomedical or social systems, where the physical laws have not been well established.
3. Contrary to the naive physics approach, with inductive reasoning, it is difficult to improve the results by incorporating more a priori knowledge of the system structure. This is one of the major drawbacks of the sheer generality of the technique.
4. Similar to the naive physics approach, also inductive reasoning mimics the way how humans (and dogs) think about the behavior of physical systems. However, while the naive physics approach tries to utilize preconceived knowledge about basic principles of operation of a physical system, inductive reasoning mimics the process of learning by observation.
5. Inductive reasoning works well for quite complex systems.
6. Feedback loops do not pose any difficulty. On the contrary, the more tightly coupled a system is, the better will be the results that we expect from the inductive reasoning process.

15. Conclusions

In this chapter we have discussed several quite different approaches to qualitative modeling and simulation. In particular, the approaches of "naive physics" and of "inductive reasoning" were discussed in more detail. It was found that

all of these techniques have their particular virtues and shortcomings. The techniques presented in this chapter are not really in competition with each other. On the contrary, they complement each other rather well. It would seem worthwhile to study properties of neural networks as an alternative to the inductive reasoning approach to qualitative modeling. To our knowledge, this has not yet been tried.

References

1. Blalock, J.M., Jr. *Causal Inferences in Nonexperimental Research.* The University of North Carolina Press, Chapel Hill, N.C., 1964.
2. Cellier, F.E. Qualitative simulation of technical systems using the general system problem solving framework. *Int. J. Gen. Syst.* 13, 4 (1987), 333–344.
3. Cellier, F.E., and Yandell, D.W. SAPS-II: A new implementation of the systems approach problem solver. *Int. J. Gen. Syst.* 13, 4 (1987), 307–322.
4. Forbus, K.D. Qualitative process theory. *Artif. Intell.* 24 (1984), 85–168.
5. de Kleer, J., and Brown, J.S. A qualitative physics based on confluences. *Artif. Intell.* 24 (1984), 7–83.
6. Klir, G.J. *Architecture of Systems Problem Solving.* Plenum Press, New York, 1985.
7. Kuipers, B.J. Qualitative simulation. *Artif. Intell.* 29 (1986), 289–338.
8. Law, A.M., and Kelton, W.D. *Simulation Modeling and Analysis.* McGraw-Hill, New York, 1982.
9. Systems Control Technology. *CTRL-C, A Language for the Computer-Aided Design of Multivariable Control Systems, User's Guide.* Systems Control Technology, Palo Alto, Calif., 1984.
10. Uyttenhove, H.J. SAPS—System Approach Problem Solver. Ph.D. dissertation (G.J. Klir, Adv.), SUNY Binghampton, 1979.
11. Vesanterä, P.J. Qualitative Flight Simulation: A Tool for Global Decision Making. M.S. thesis, Dept. of Electrical and Computer Engineering, Univ. of Arizona, Tucson, Ariz., 1988.
12. Vesanterä, P.J., and Cellier, F.E. Building intelligence into an autopilot—Using qualitative simulation to support global decision making. *Simulation* 52, 3 (1989), 111–121.
13. Zadeh, L.A. Syllogistic reasoning in fuzzy logic and its application to usuality and reasoning with dispositions. *IEEE Trans. Syst., Man, Cybern. SMC*-15, 6 (1985), 754–763.
14. Zadeh, L.A. Test-score semantics as a basis for a computational approach to the representation of meaning. *Literary Linguistic Comput.* 1, 1 (1986), 24–35.
15. Zadeh, L.A. A simple view of the Dempster–Shafer theory of evidence and its implication for the rule of combination. *The AI Mag.* (Summer 1986), 85–90.
16. Zadeh, L.A. A computational theory of disposition. *Int. J. Intell. Syst.* 2 (1987), 39–63.