The Use of Intelligently Controlled Simulation to Predict a Machine's Long-Term Behavior

Andrew Gelsey*
Computer Science Department
Rutgers University
New Brunswick, NJ 08903
gelsey@cs.rutgers.edu

Abstract

I present algorithms for automated long-term behavior prediction which can recognize when a simulation has run long enough to produce a representative behavior sample, characterize the behavior, and determine whether this behavior will continue forever, or eventually terminate or otherwise change its character. I have implemented these algorithms in a working program which does long-term behavior prediction for mechanical devices.

Introduction

Many physical systems are sufficiently complex that the process of analyzing their behavior must include either actual or simulated experiments [Forbus and Falkenhainer 1990, Abelson *et al.* 1989]. However, though these experiments may accurately show the behavior of a physical system over any particular time period, significant human effort is generally needed to predict a system's long-term behavior from experimental data. Automating this process involves

1. recognizing when an experiment has run long enough to produce a representative behavior sample
2. characterizing the system's behavior
3. determining whether this behavior will continue forever, or eventually terminate or otherwise change its character

These are the problems I address in this paper. I will focus on the use of simulated experiments, though many of my results should also apply to physical experiments. At present, this work is limited to mechanical devices, particularly clockwork mechanisms.

In this paper I present algorithms to solve the problems listed above. I have implemented these algorithms in a working program which acts as an intelligent controller for a numerical simulator. This program has three distinct modules:

1. the automated modeler, which transforms a description of a machine's raw physical structure into a behavioral model suitable for numerical simulation [Gelsey 1989]
2. the numerical simulator, which uses standard numerical simulation algorithms modified to handle the behavioral model generated by the automated modeler [Gelsey 1990]
3. the intelligent controller, which is based on the algorithms I will describe in this paper

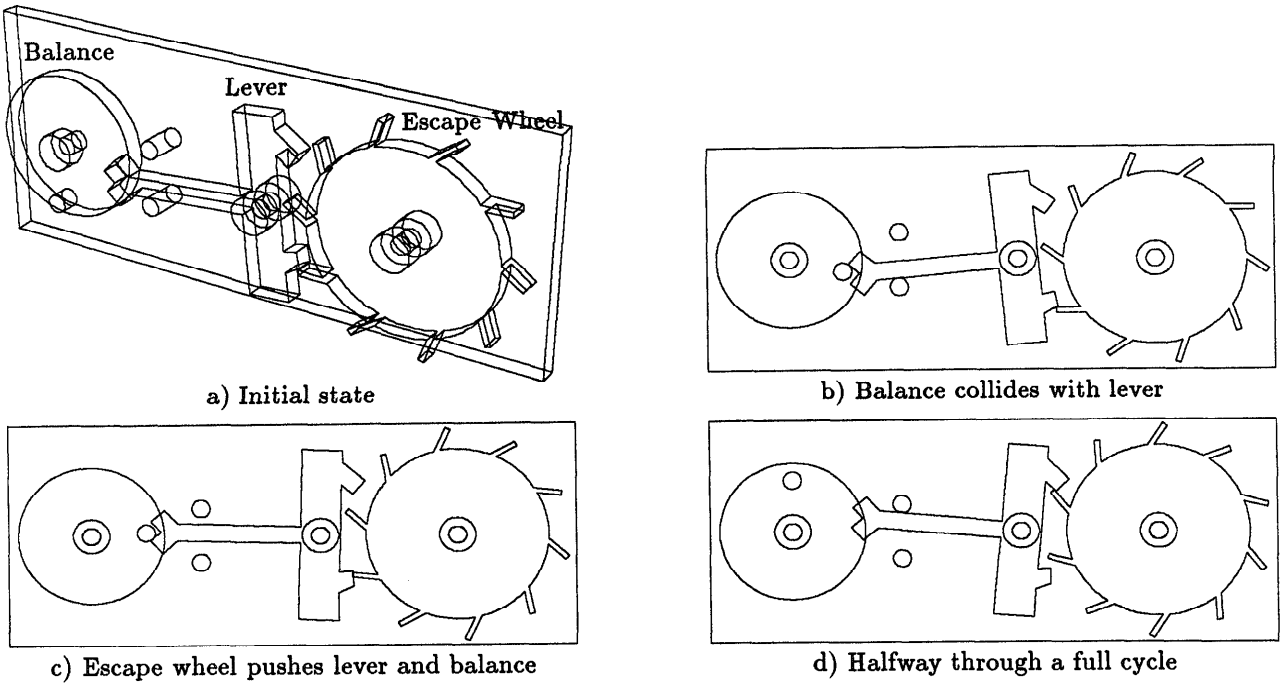
An Example

The escapement mechanism in Figure 1 keeps the average speed of a clock or watch constant by allowing the escape wheel, which is pushed clockwise by a strong spring, to advance by only one tooth for each oscillation of the balance. In Figure 1a the motion of the escape wheel is blocked by the lever, and the balance is motionless and about to be driven counterclockwise by its attached spring. In Figure 1b the balance has hit the lever. The momentum of the balance pushes the lever far enough to free the escape wheel, which then pushes both lever and balance as in Figure 1c. This pushing restores the energy the balance loses to friction, so that it can act as a harmonic oscillator in spite of damping. Finally, in Figure 1d, the escape wheel and lever are locked together again, and the balance has been brought temporarily to a halt by its spring.

My program's input is a CAD/CAM solid model[1] of the geometric structure of a machine, supplemented with information about masses, spring constants, and coefficients of friction. This input is first processed by the automated modeler module of the program, which generates a behavioral model of the machine. The automated modeler starts by identifying a set of *state variables* whose (numerical) values summarize the state of the machine at any particular time. A typical model for the escapement mechanism is a rigid

*This research was supported by National Science Foundation grant IRI-8812790 and by the Defense Advanced Research Projects Agency and the National Aeronautics and Space Administration under NASA grant NAG2-645.

[1] I use the PADL-2 solid modeling system developed by the Production Automation Project at the University of Rochester [Hartquist 1983].



a) Initial state

b) Balance collides with lever

c) Escape wheel pushes lever and balance

d) Halfway through a full cycle

Figure 1: Clock or watch escapement mechanism

body model with six state variables: the positions of the three moving parts, and their velocities.

At any particular time, a machine's behavior obeys a set of differential equations in the state variables, which may be numerically solved to predict its behavior.[2] However, the differential equations may dynamically change their form as parts of the machine come into contact and separate. Thus the behavioral model generated by my program includes data which allows the numerical simulation module to generate the appropriate differential equations for any particular state of the machine. After each numerical simulation step, the current set of equations is automatically checked for validity and reformulated if necessary [Gelsey 1989, Gelsey 1990].

Figure 2 shows a plot of my program's numerical simulation of the behavior of the escapement mechanism. To the human eye, this plot clearly shows the regularity of the mechanism's behavior, but this regularity is not explicit in the numerical simulation data. Furthermore, though the plot shows a sufficiently long behavior trace to make the behavioral regularity clear, the only reason the simulation ran for that length of time is because of explicit instructions. Also, though Figure 2 shows the regular behavior of the escapement, even a human would need to do further analysis to decide how long this regular behavior would continue. In the next section I present an algorithm for continuously processing a stream of simulation data to determine when it has become long enough to show regularities and to characterize those regularities. Later in this paper I present algorithms for doing controlled simulated experiments to determine the limits of validity of a hypothesized behavioral regularity.

[2]The differential equations for the escapement mechanism are nonlinear, as are those for all but the simplest mechanisms. Thus they cannot be solved directly, and instead must be simulated numerically. My current simulator uses a variable-step Runga-Kutta method.

Finding Behavioral Regularities

I define a machine's behavior to be regular if the behavior consists of continual repetitions of the same behavior pattern. A strict definition of regularity would require that each instance of the behavior pattern be identical, but my definition allows successive instances to change in a well-defined way, while maintaining the same underlying form. In particular, I require that each instance of the behavior pattern should have the same number of states with minimal kinetic energy, and that the states of the system at corresponding kinetic energy minima in successive instances of the behavior pattern should be related in a specific well-defined way.

My definition of regularity focuses on kinetic energy minima because these minima tend to lie between qualitatively distinct regions of the behavior pattern, for example at a time when an oscillating part of a machine switches from moving in one direction to moving in another, or when a part of a machine receives a push

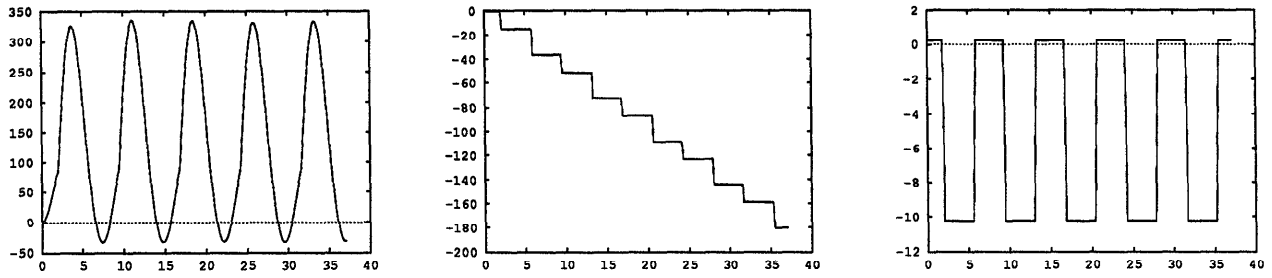


Figure 2: Motion of escapement mechanism (horizontal axes show time, vertical axes show positions in degrees)

or other energy boost so that its velocity stops decreasing and starts increasing. Thus a change in the number of kinetic energy minima in a behavior pattern tends to indicate a significant qualitative change in behavior.

My program guides its search for behavioral regularities by comparing states of the system having minimal kinetic energy. This strategy has two principal advantages: it is very efficient, giving a search time which is linear in the amount of input data, and it reduces the chance of incorrect matches by focusing the search on the most significant points in the system's behavior. In order to avoid being misled by spurious local kinetic energy minima, the program only considers minima that are approximately global minima. It tests for global minimality by maintaining range data which it gathers dynamically as the simulation proceeds.

It should be emphasized that the program looks for regularities in the behavior of the state variables (like that shown in Figure 2), not in the behavior of the kinetic energy. Kinetic energy is simply used as a guide to tell the program when to look at the state variables — it might be considered a measure of the "interestingness" of a particular state of the machine. It may seem that using kinetic energy as a guide could cause the program to miss some interesting behaviors, for example in a machine which maintained constant total kinetic energy while transferring kinetic energy back and forth internally. However, since kinetic energy is a sum of terms involving squares of velocities, it seems unlikely that such a machine could be built. In general, since the kinetic energy at any particular time is a function of the state of the system at that time, regularities in the behaviors of the state variables should be reflected in the kinetic energy.

Figure 3 shows my program's representation of a machine's behavior pattern. Figure 4 shows the algorithm my program uses to try to construct a hypothesis about the machine's behavior pattern which is consistent with the stream of behavior data coming from the numerical simulator.

The program simulates the behavior of the machine until it finds several kinetic energy minima, which it requires to be in the bottom $\mathcal{P}_{\text{range\_fraction}}$ of the range

1. Number of kinetic energy minima in an instance of the behavior pattern (N)
2. Duration of the behavior pattern (D)
3. Net change in each state variable over each pattern instance (Δ)
4. List of periodically superimposed processes (Figure 6)

Figure 3: The data describing a behavior pattern

of kinetic energies encountered during the simulation. The first $\mathcal{P}_{\text{ignore}}$ minima are ignored so that the program won't be misled by transient startup phenomena. Then the program searches for a locally satisfactory behavioral hypothesis by successively forming each possible hypothesis which is consistent with the currently available data. After each hypothesis is formed it is tested against the results of further simulation. Though the program requires that new data match the predictions of the hypothesis fairly closely, it does not require exact matches because they would never be found due to transients and other noise in the numerical data. The match parameters can be adjusted to "tune" the behaviors the program will recognize. For example, identical patterns with very different time durations could be detected by giving $\mathcal{P}_{\text{t\_match}}$ a large value.

This algorithm is based on the following simple but powerful ideas:

1. Given a set of simulation data including exactly n kinetic energy minima, there will be exactly one behavioral hypothesis (of the form specified in Figure 3) which is both consistent with the data and which requires that the machine's behavior pattern contain exactly n kinetic energy minima.

2. Given a set of simulation data including more than n kinetic energy minima, either the single n-minima hypothesis which is consistent with the first n minima will also be consistent with the rest of the data, or else no n-minima hypothesis can be consistent with all of the data, which implies that

[Default parameter values:
$\mathcal{P}_{\text{confirm}} = 2$, $\mathcal{P}_{\text{range\_fraction}} = 10^{-5}$, $\mathcal{P}_{\text{ignore}} = 4$]
While no behavioral hypothesis has been formed
OR ⟨hypothesis belief level⟩ $< N * \mathcal{P}_{\text{confirm}}$
Perform a simulation step
Update range information
If the previous state was a local kinetic energy minimum
AND its energy level $< \mathcal{P}_{\text{range\_fraction}} *$ ⟨top of kinetic energy range⟩
AND $\mathcal{P}_{\text{ignore}}$ kinetic energy minima have been ignored
Then
If there is no hypothesis
Then hypothesize a behavior pattern:
$N \leftarrow 1$
$D \leftarrow$ difference in time between the current kinetic energy minimum and the previous one
$\Delta \leftarrow$ differences in the values of the state variables between the current kinetic energy minimum and the previous one
Else the current hypothesis remains valid if and only if:
D matches the time elapsed since the Nth previous kinetic energy minimum to within $\mathcal{P}_{\text{t\_match}}$ (default: 1%)
AND Δ matches the changes in the values of the state variables since the Nth previous kinetic energy minimum to within $\mathcal{P}_{\text{s\_v\_match}}$ (default: 2%)
If the current hypothesis fails then form a new hypothesis:
$N \leftarrow N + 1$
$D \leftarrow$ difference in time between the current kinetic energy minimum and the Nth previous one
$\Delta \leftarrow$ differences in the values of the state variables between the current kinetic energy minimum and the Nth previous one
Else increment ⟨hypothesis belief level⟩

Figure 4: Algorithm to find a locally satisfactory behavioral hypothesis

3. Once the program has found new data not consistent with a previously formed n-minima hypothesis (which was consistent with some particular sequence of n minima), it need never again consider any hypothesis with n kinetic energy minima.

Thus the program can iterate through the possible behavioral hypotheses, testing each in turn.

The behavior pattern of the escapement mechanism in Figure 1 has two kinetic energy minima, one at each extreme position of the balance. The initial hypothesis formed by my program is that each behavior pattern instance has only one kinetic energy minimum, and that all three moving parts have a net position change per pattern. When the program tests this behavioral hypothesis at the next minimum, it finds that the net displacements of the balance and lever are the negatives of the hypothesized changes. The initial behavioral hypothesis is then rejected, and a new hypothesis is formed in which the behavior pattern has two kinetic energy minima, and neither the balance nor the lever has a net position change per pattern. This hypothesis is then confirmed over the next $\mathcal{P}_{\text{confirm}}$ pattern instances, and finally accepted as locally valid.

Hypothesis Failures

After the program finds a locally satisfactory behavioral hypothesis, it performs a variety of tests to determine whether the hypothesis is also globally satisfactory and therefore useful for long-term behavior prediction. If the hypothesis fails a test it is modified if possible or otherwise rejected. I classify hypothesis failures into two categories: gradual effects of steadily changing state variables, and sudden transitions.

Gradual Effects of Steadily Changing State Variables

What I call a "gradual" effect of steadily changing state variables is an effect which is too small to be detected by the algorithm in Figure 4 because its magnitude over a small number of behavior cycles is comparable to that of the noise in the data. The escapement example includes several such effects. The initial locally satisfactory hypothesis for the escapement example predicts that the balance and lever will return to the same position at the end of each pattern, but that the escape wheel will have a net position change. As the position of the escape wheel changes, the tension in its attached spring decreases, and it is clear that the clock must eventually stop running when the stored energy of the spring is no longer sufficient to compensate for the mechanism's energy losses due to friction. The energy stored in the mainspring thus imposes an upper time limit on the validity of the behavioral hypothesis. It also turns out that the decreasing tension in the mainspring results in a very slow decrease in the amplitude of the oscillations of the balance. However, this decrease is much smaller than transient amplitude variations and therefore cannot be detected when the initial hypothesis is formed, so the decrease is not predicted by the initial locally valid behavioral hypothesis.

Though these gradual effects generally can't be detected in a short simulation, they become quite apparent in a long one because their magnitudes grow steadily while data noise remains bounded. However, running a long simulation may be quite costly. Instead, my program uses a technique for "sampling" pieces of a long simulation without having to actually run the simulation.

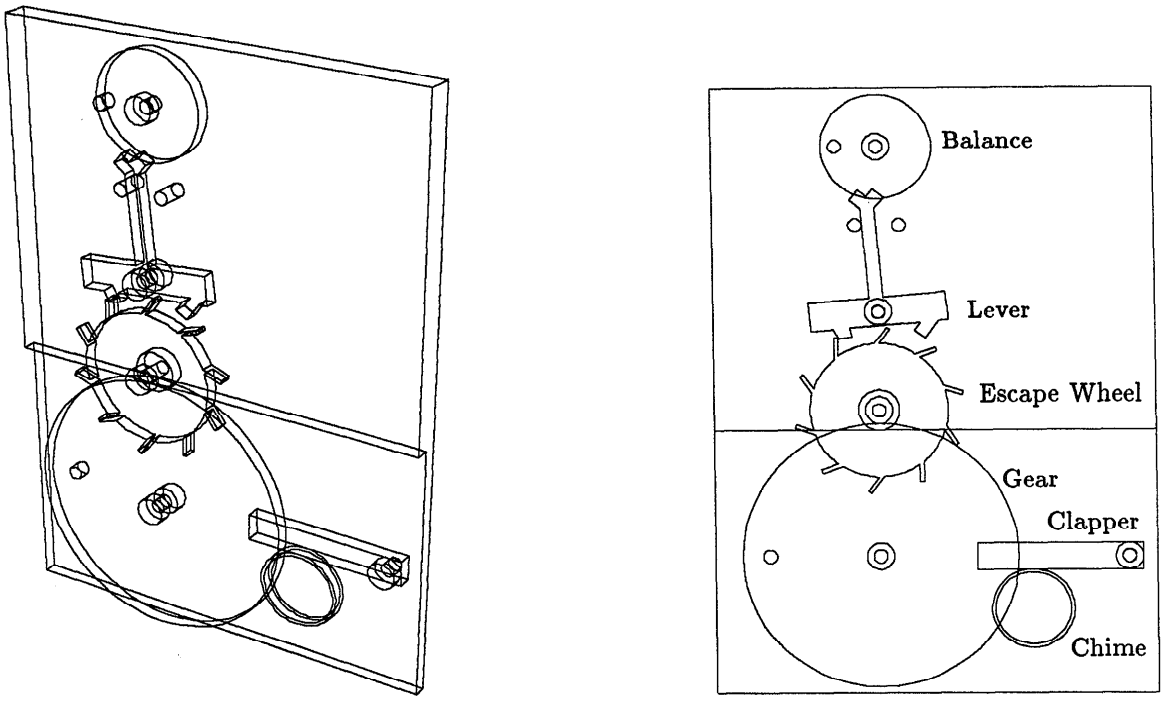


Figure 5: Chiming clock

The program does this "sampling" by "jumping" the state of the system ahead in time and then performing simulated experiments to retest its behavioral hypothesis. The data my program associates with a hypothesized behavior pattern includes, not coincidentally, exactly the information needed to make the system "jump". Jumping the system ahead by n behavior cycles is simply a matter of adjusting the time and state variables values by n times the values recorded for the behavioral hypothesis (Figure 3). This jumping strategy cannot erroneously disconfirm a correct hypothesis, though it can erroneously confirm an incorrect hypothesis, which however may subsequently be rejected by the additional tests I describe below. The capability to sample a system's future without having to run a long simulation is one of the principal justifications for forming behavioral hypotheses as specified in Figure 3.

To look for failures due to gradual effects of steadily changing state variables, my program jumps the state of the machine ahead in time so that about half of its total energy has been dissipated and then retests the local validity of the hypothesis in this new environment. If the hypothesis is still locally valid, the program then tests whether the predicted changes in each state variable match the changes revealed by simulation. If the changes do not match, the program modifies the hypothesis if possible to be consistent with the new data as well as the original data.

Sudden Transitions

Figure 5 shows a clock which chimes at regular intervals (e.g., every hour). A small gear on the escape wheel drives a large gear, which moves relatively slowly counterclockwise. The clapper is normally pressed against the chime by a spring, but the protrusion on the large gear periodically pushes it away from the chime and then releases it so that it makes a chiming sound when it hits the chime again.

Since the clock will have been running for quite a while before it first chimes, the initial behavioral hypothesis will not take the chiming into account and will therefore be an incomplete behavior prediction. This sort of hypothesis failure is quite different from the failures discussed in the previous section because random sampling of future behavior is unlikely to detect the sudden transition.

In mechanical devices, sudden transitions are changes in the patterns of contact between parts in the mechanism, which are of two types:

1. New contacts between parts not previously in contact, like the moving gear and the clapper in the chiming clock. (See the next section.)
2. Changes in the pattern of contact between two parts which regularly make contact. For example, if the escape wheel in Figure 1 had a missing tooth, the pattern of contact between the escape wheel and the lever would be affected. My program computes the minimum time the simulation must run in order to

determine whether a locally valid behavioral hypothesis will be violated by such changes, and increases $\mathcal{P}_{\text{confirm}}$ to postpone local confirmation of the hypothesis until the passing of at least that much time has been simulated. The program takes advantage of any symmetries in the shapes of the parts involved in order to reduce the required simulation time if possible. The algorithms for handling this case are given in my dissertation [Gelsey 1990], but are too lengthy to reproduce here.

New Contacts Between Parts Not Previously in Contact

The problem of predicting collisions between two moving parts is quite difficult. For example, with just the right timing a machine gun may be fired through the space where an airplane's propeller is turning without hitting the propeller. With slightly different timing, the propeller will be shot off. Writing a general algorithm capable of distinguishing these two cases is not simple. My program is currently limited to the case of contact between a moving part and one which was not moving prior to the collision.

When a new contact occurs, several things may happen. If a moving part hits one which is not capable of moving, then the behavioral hypothesis will fail completely at the time of contact. Simulation can then be used to determine the subsequent behavior of the mechanism; typically it will quickly come to a halt. On the other hand, if the part which was not moving prior to the contact is capable of motion, a new behavior pattern may emerge.

Many mechanisms are multiply periodic; different sorts of regular behavior occur at very different time scales. For example, a chiming clock like that in Figure 5 will have an escapement with a regular behavior pattern that might be repeated several times a second, and a chiming mechanism whose pattern might only be repeated once an hour. Perhaps the commonest case of multiple periodicity is what I call a *periodically superimposed process*: an additional behavior pattern which is regularly superimposed on the basic behavior pattern without disturbing it.

Because a superimposed process does not disturb the basic behavior pattern, it can only change state variable which remain constant during the basic pattern. I call these state variables *static*. The static state variables are identified during the simulation needed to generate the original locally valid behavioral hypothesis. Figure 6 shows my program's representation of a periodically superimposed process.

Figure 7 summarizes the algorithm my program uses to identify periodically superimposed processes. The motion envelope of a part is the volume it sweeps out as it moves. If two parts have intersecting motion envelopes but have never been in contact, their first point of contact is computed directly from the geometry of the mechanism using algorithms given in my dissertation [Gelsey 1990] which are too lengthy to present in this paper. If one of the parts is fixed (part of the frame) then my program determines that the valid region for the original behavioral hypothesis ends at the new contact; otherwise, the program attempts to identify a new periodic process that will be superimposed on the originally hypothesized behavior pattern.

1. Period (time between starts of successive instances of the superimposed process)
2. Duration of each instance of the process
3. Time at which the first instance of the process starts
4. List of static state variables involved in this periodically superimposed process
5. Net change in each involved state variable over each process instance

Figure 6: Periodically superimposed process

1. Use geometric computations to determine when the first contact will occur between two parts that have never been in contact but which have intersecting motion envelopes
2. Jump system state halfway to new contact to check for gradual effects of steadily changing state variables, and modify behavioral hypothesis if necessary
3. If neither part is fixed, jump to time of first new contact and start the behavior simulator. While simulating,
   (a) Monitor behavior of nonstatic state variables to make sure none of them violate the original behavioral hypothesis
   (b) Monitor behavior of static state variables to gather data describing a periodically superimposed process (Figure 6)
4. If the superimposed process does not violate the original hypothesis, jump until only half of total energy is left to check for gradual effects of steadily changing state variables, and modify hypothesis if necessary

Figure 7: Analyzing behavior resulting from a new contact

Implementation

I have implemented all of the algorithms presented in this paper and tested them on a number of examples, which are described in detail with test results in [Gelsey 1990]. These examples include the mechanisms in Figure 1 and Figure 5. In both cases the program finds a behavioral hypothesis with two kinetic energy minima corresponding to the two extreme positions of the balance. For the chiming clock, the program also finds a periodically superimposed process representing the chiming.

Related Work

Yip[1989] wrote a program which can automatically plan, execute, and interpret numerical experiments concerning Hamiltonian systems with two degrees of freedom. His program, like mine, intelligently controls numerical simulations, but the research problems addressed by the two programs are quite different. His program is more powerful than mine in the sense that it can usefully analyze physical systems, including chaotic systems, which do not meet the definition of regularity I give in this paper. On the other hand, his program is not capable of dynamically changing the set of differential equations it is working on, while my program, using a behavioral model which is closer to a system's underlying physical structure, can dynamically reformulate its differential equations to gracefully handle "surprises" like those found in the chiming clock in Figure 5.

Forbus and Falkenhainer[1990] describe an intelligent controller for numerical simulations based on Qualitative Process Theory [Forbus 1984]. Their work emphasizes automatically generating a system's equations, which may change over time, from a physical model. Their input model is, however, at a considerably higher level than the model of raw physical structure my program uses as input, which would prevent their program from being able to generate a numerical simulation of a device like a clock in which contacts between parts appear and disappear dynamically. They do not presently address the problem of identifying repetitive behavior from a numerical simulation.

Joskowicz[1988] and Faltings[1987] partition a mechanism's configuration space into qualitatively distinct regions. Joskowicz[1989] presents operators to simplify and abstract such qualitative descriptions. These operators are not powerful enough, however, to predict regular behaviors for mechanisms like the examples I have given in this thing. Nielsen[1988] builds on Faltings' work to construct a qualitative simulator for devices like mechanical clocks. His simulation suffers from a qualitative model's inherent liability to make ambiguous behavior predictions, and thus predicts that the clock may run, but not whether it will run and if so, for how long. Furthermore the simulation, being qualitative, cannot predict what the period of the clock will be or whether it will be constant, which would seem to be the most essential characteristic of a clock.

Weld[1986] describes a program for the detection and compact representation of repetitive behavior in qualitative simulations. His algorithms are designed for systems having a small finite set of possible states, and do not appear to be easily applicable to systems whose state variables have values which may be any real number, and whose repetitive behavior is noisy and thus never precisely repeats a state.

Yeh[1990] wrote a program which transforms a description of one iteration of a repetitive behavior into a summary description of the behavior of many repetitions. His work does not address the question of detecting repetition in a behavior sample.

The output of a simulator is time series data and thus may be analyzed by well known statistical techniques including Fourier analysis [Bloomfield 1976]. However, these techniques are mainly useful when the data to be analyzed contains a large number of repetitions of a behavior pattern, unlike my algorithms which can identify regular behavior after just a few repetitions of the behavior pattern. Furthermore, these statistical techniques lack the capability my algorithms have to determine when a data sample is long enough to be representative. This lack is especially inconvenient because these statistical techniques do not work in an incremental way: they must have the entire data sample available to begin processing, unlike my algorithms which operate on a stream of simulation data, processing each piece as it becomes available.

Limitations and Future Work

One obvious limitation of the work described in this paper is that the algorithms I present have only been tested on clockwork mechanisms. This limitation is a result of limitations in my current automated behavior modeling algorithms [Gelsey 1989, Gelsey 1990]. As those modeling algorithms improve they will allow the algorithms presented in this paper to be tested in a wider context.

A more important limitation of this work is that the set of behavioral hypotheses my program can construct consists only of those expressible in the rather simple "language" given in Figure 3. The simplicity of this language is the source of much of the power of my current algorithms, but it remains to be seen how well this limited language can describe the wide variety of physical systems that we consider to have regular behavior.

My algorithm for finding a locally satisfactory behavioral hypothesis (Figure 4) appears to be fairly general and not at all limited to mechanical devices. However, it is quite dependent on the simplicity of the hypothesis language and on my definition of regular behavior. Similarly, "jumping" the state of a system forward in time to test the global validity of a locally satisfactory hypothesis is a general technique that could be applied to any physical system whose regular behavior could be described by my hypothesis language. Of course, if the hypothesis language had to be extended to handle a wider variety of physical systems, it is quite possible that these algorithms could also be extended to handle the more general situation.

In contrast, my algorithms for detecting sudden transitions are quite specific to the mechanical device domain. An important area of future research will be the development of a more general theory of sudden transitions in physical systems.

Another area of future work is to make the algorithms I have presented more capable of dealing with noisy data. A central idea behind my algorithm

for finding a locally satisfactory behavioral hypothesis (Figure 4) is that the program may correctly reject a behavioral hypothesis as soon as it contradicts available data, but it seems quite possible that a fundamentally valid hypothesis might be spuriously ruled out by noise in some portion of the data. A reasonable approach to this problem might be an improved measure of how strongly the data violates the hypothesis.

Various other extensions to my program might be desirable. For example, the single level of periodically superimposed processes I describe in this paper may be insufficient to describe very complicated mechanisms — multiple levels might be necessary. Also, the program might produce confusing output when applied to a mechanism having several completely independent submechanisms. The kinematic analysis I use to form behavioral models [Gelsey 1987, Gelsey 1990] could easily identify the independent submechanisms and present them separately to the program I describe in this paper.

An interesting but difficult problem would be to use the algorithms I have presented to find a machine's behavior pattern, and then to apply that knowledge in trying to extract additional information from the machine's original behavioral model, for example to provide additional evidence for a hypothesis or to modify or disconfirm it.

Conclusion

One of the most basic capabilities we should expect in a program that reasons about physical systems is the ability to predict the system's long-term behavior. For all but the simplest systems, global analysis techniques are not adequate for the task of behavior prediction unless supplemented by numerical simulation controlled intelligently by either a human or a program. In this thing I have presented algorithms to control a numerical simulator and

1. recognize when an experiment has run long enough to produce a representative behavior sample
2. characterize the representative behavior
3. determine whether the representative behavior will continue forever, or eventually terminate or otherwise change its character

for a certain class of machines, and have described a working program that can do long-term behavior prediction for these machines.

References

Abelson, H.; Eisenberg, M.; Halfant, M.; Katzenelson, J.; Sacks, E.; Sussman, G.J.; Wisdom, J.; and Yip, K. 1989. Intelligence in scientific computing. *Comm. ACM* 32(5):546–562.

Bloomfield, Peter 1976. *Fourier Analysis of Time Series: An Introduction.* John Wiley & Sons, New York.

Faltings, Boi 1987. *Qualitative Place Vocabularies For Mechanisms in Configuration Space.* Ph.D. Dissertation, Dept. of Computer Science, University of Illinois at Urbana-Champaign.

Forbus, Kenneth D. and Falkenhainer, Brian 1990. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *Proceedings, Eighth National Conference on Artificial Intelligence*, Boston, MA. AAAI-90.

Forbus, Kenneth 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Gelsey, Andrew 1987. Automated reasoning about machine geometry and kinematics. In *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, Orlando, Florida.

Gelsey, Andrew 1989. Automated physical modeling. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan USA.

Gelsey, Andrew 1990. *Automated Reasoning about Machines.* Ph.D. Dissertation, Yale University. YALEU/CSD/RR#785.

Hartquist, Gene 1983. Public PADL-2. *IEEE Computer Graphics and Applications* 30–31.

Joskowicz, Leo 1988. *Reasoning about Shape and Kinematic Function in Mechanical Devices.* Ph.D. Dissertation, New York University Dept. of Computer Science.

Joskowicz, Leo 1989. Simplification and abstraction of kinematic behaviors. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan USA.

Nielsen, Paul E. 1988. *A Qualitative Approach to Rigid Body Mechanics.* Ph.D. Dissertation, Dept. of Computer Science, University of Illinois at Urbana-Champaign.

Weld, Daniel S. 1986. The use of aggregation in causal simulation. *Artificial Intelligence* 30:1–34.

Yeh, Alexander 1990. Finding the average rates of change in repetitive behavior. In *Proceedings, Eighth National Conference on Artificial Intelligence*, Boston, MA. AAAI-90.

Yip, Kenneth 1989. *KAM: Automatic Planning and Interpretation of Numerical Experiments Using Geometrical Methods.* Ph.D. Dissertation, Dept. of EE and CS, M.I.T.