describes the accuracy of our knowledge of $x(t)$ at time $t = t_0$, and

$$|v(t_0) - v_0| \ \leq \ \Delta\ v_0$$

describes the accuracy of our knowledge of $v(t)$ at time $t = t_0$. Then the above inequalities become

$$|v(t_1) - v_0| \ \leq \ \Delta\ v_0 + |t_1 - t_0| * A,$$

and therefore

$$|x(t_1) - x_0| \ \leq \ \Delta\ x_0 + |t_1 - t_0| * \Delta\ v_0 + \frac{1}{2} * |t_1 - t_0|^2 * A,$$

so we have to have

$$\Delta\ t \ \leq \ |(V - \Delta\ v_0)/A|$$

to keep the velocity within bounds, and

$$(\Delta\ t)^2 + \frac{2 * \Delta\ v_0}{A} * (\Delta\ t) \ \leq \ |2 * (P - \Delta\ x_0)/A|$$

to keep the position within bounds.

At this point, we are stuck unless we can say something more helpful about the acceleration. Suppose we know that the acceleration jumps around, and that it has a distribution of values with mean 0 and variance R. In this case, we might be able to reduce the estimates for position and velocity and improve the time intervals.

References

[1] P. Henrici, Elements of Numerical Analysis, Wiley (1964)

[2] J. Stoer, R. Bulirsch, Einfu:hrung in die Numerische Mathematik, II Springer (1973)

$$(x,y)(k+1) \;=\; (x,y)(k) \begin{pmatrix} cos(dt) & sin(dt) \\ -sin(dt) & cos(dt) \end{pmatrix},$$

and we have as above

$$\begin{aligned} X(0) &= (1,0), \\ X(k+1) &= X(k)(I*cos(dt)+J*sin(dt)), \end{aligned}$$

so

$$\begin{aligned} X(k) &= (1,0)\ (I*cos(dt)+J*sin(dt))^{\mathrm{k}}, \\ &= (1,0)\ (I*cos(k*dt)+J*sin(k*dt)), \end{aligned}$$

and

$$\begin{aligned} x(k*dt) &= cos(k*dt), \\ y(k*dt) &= sin(k*dt), \end{aligned}$$

from which we can hazard a guess as to the correct solution.

5.2 Measurement

Let us take a simple system in which the velocity and position are occasionally known through inexact measurement. Our state variables are p for the position, v for the velocity, and a for the unknown acceleration.
We assume that the acceleration a is bounded by some constant A, so that for any times $t_0 < t_1$

$$|v(t_1) - v(t_0)| \;<\; |t_1 - t_0| * A.$$

We assume that we have characterizers

$$(a(t); t_{i-1}, t_i)$$

that describe the acceleration, and model characterizers

$$(v = p'; 0^-, -),$$
$$(a = v'; 0^-, -).$$

Therefore, we can compute the velocity and position by

$$v(t) = v(t_0) \;+\; \int_{t_0<u<t} a(u)\ du,$$

$$p(t) = p(t_0) \;+\; \int_{t_0<u<t} v(u)\ du.$$

The problem is to choose measurement times and variables that maintain a certain accuracy in the estimates of position.
We assume that we can measure position within a bound

$$|p_{\mathrm{meas}}(t) - p(t)| \;<\; P,$$

and that we can measure velocity within a bound

$$|v_{\mathrm{meas}}(t) - v(t)| \;<\; V,$$

but that we want to keep our estimate p_{est} of position either more accurately than the position measurement error (this might or might not be possible) or using as few measurements as possible.
We assume first that x_0, v_0 are known, and consider an interval $[t_0, t_1)$. We compute

$$|v(t_1) - v_0| \;<\; |t_1 - t_0| * A,$$

and therefore

$$|x(t_1) - x_0| \;<\; \frac{1}{2} * |t_1 - t_0|^2 * A,$$

so we would have to choose

$$\Delta\ t = t_1 - t_0$$

so that

$$\Delta\ t \;\leq\; |V/A|$$

to keep the velocity within bounds, and

$$(\Delta\ t)^2 \;\leq\; |2 * P/A|$$

to keep the position within bounds.
But of course, we don't know $x(t)$ or $v(t)$ after the first time interval, so we need to change the previous derivation a bit.
We assume that we know x_0 and v_0, and that

$$|x(t_0) - x_0| \;\leq\; \Delta\ x_0$$

$$\begin{aligned} y' &= x, \\ x(0) &= 1, \\ y(0) &= 0, \end{aligned}$$
as above. Our initial conditions are
$$\begin{aligned} &(x : 1; 0), \\ &(y : 0; 0), \end{aligned}$$
as before.
The method we use is a simplified second-order Runge-Kutta method [2], [1], which basically amounts to averaging the usual Euler approximation in an interval with a linear reapproximation at the endpoint of the interval. At a given time $t = t_0$, if we have
$$\begin{aligned} x(t_0) &= x_0, \\ y(t_0) &= y_0, \end{aligned}$$
then we have
$$\begin{aligned} x(t) &= x_0 - y_0 * dt - x_0 * dt^2/2, \\ y(t) &= y_0 + x_0 * dt - y_0 * dt^2/2, \end{aligned}$$
and it is the extra dt^2 terms that make the method second-order.
As above, we assume equal time intervals and get an iteration
$$\begin{aligned} x(0) &= 1, \\ y(0) &= 0, \\ x(k+1) &= x(k) - y(k) * dt - x(k) * dt^2/2, \\ y(k+1) &= y(k) + x(k) * dt - y(k) * dt^2/2, \end{aligned}$$
which can be written as a vector equation
$$\begin{aligned} (x, y)(0) &= (1, 0), \\ (x, y)(k+1) &= (x, y)(k) \begin{pmatrix} 1 - dt^2/2 & dt \\ -dt & 1 - dt^2/2 \end{pmatrix}, \end{aligned}$$
and we have as above
$$\begin{aligned} X(0) &= (1, 0), \\ X(k+1) &= X(k)(I * (1 - dt^2/2) + J * dt), \end{aligned}$$
so
$$X(k) = (1, 0)\ (I * (1 - dt^2/2) + J * dt)^{\mathrm{k}},$$
which can be computed exactly.
Since the eigenvalues of $(I * (1 - dt^2/2) + J * dt)$ are $1 - dt^2/2 \pm i * dt$, which have magnitude $1 + dt^4/4$, this simple method still does not converge (but much more slowly).

5.1.3 Higher-Order Example

A similar analysis of the usual 4th-order Runge-Kutta method leads to an iteration
$$\begin{aligned} x(t) &= x_0 - y_0 * dt - x_0 * dt^2/2 + y_0 * dt^3/6 + x_0 * dt^4/24, \\ y(t) &= y_0 + x_0 * dt - y_0 * dt^2/2 - x_0 * dt^3/6 + y_0 * dt^4/24, \end{aligned}$$
with matrix
$$\begin{pmatrix} 1 - dt^2/2 + dt^4/24 & dt - dt^3/6 \\ -dt + dt^3/6 & 1 - dt^2/2 + dt^4/24 \end{pmatrix},$$
and eigenvalue magnitude of $1 + dt^6/36 + dt^8/24^2$, which is still greater than one. In fact, since this equation (in (x, y) space) represents moving around a circle, any extrapolation method based on tangents at a single point will fail, since all of the tangent vectors point outward from the circle. We note that the iteration equations do have the first terms of the usual Maclaurin series for $\sin(dt)$ and $\cos(dt)$, so we try out a different iteration:
$$\begin{aligned} x(t) &= x_0 * cos(dt) - y_0 * sin(dt), \\ y(t) &= y_0 * cos(dt) + x_0 * sin(dt), \end{aligned}$$
which can be written as a vector equation
$$(x, y)(0) = (1, 0),$$

for t in some small interval

$[t_0, t_1 = t_0 + dt)$.

The characterizers that describe this situation are:

$(x : x_0 + z_0 * (t - t_0); t_0, t_0 + dt)$,

$(y : y_0 + x_0 * (t - t_0); t_0, t_0 + dt)$,

which we want to be true for all choices of x_0, y_0, t_0, and dt (which ones we actually use in our system description depend on how we choose the time intervals in the solution).

The characterizers that describe the initial conditions are difficult, because they cannot be described with half-open intervals of the shape we have thus far described:

$(x : 1; 0)$,

$(y : 0; 0)$,

which is always going to be a problem in systems that start at a certain time.

In a more sophisticated system, the choice of next time interval would depend on the computed accuracy of the current solution.

For this example, we simply make all the time intervals the same, and say that the characterizer pair

$(x : x_1 + z_1 * (t - t_1); t_1, t_1 + dt)$,

$(y : y_1 + x_1 * (t - t_1); t_1, t_1 + dt)$

propagates the pair

$(x : x_0 + z_0 * (t - t_0); t_0, t_0 + dt)$,

$(y : y_0 + x_0 * (t - t_0); t_0, t_0 + dt)$

iff

$$\begin{aligned} x_1 &= x_0 + z_0 * dt, \\ y_1 &= y_0 + x_0 * dt, \\ t_1 &= t_0 + dt, \end{aligned}$$

which are the conditions for the first pair to meet the second (the condition $z_1 = -y_1$ is part of the definition of these characterizer pairs).

Extending the iteration, we have

$$\begin{aligned} x(0) &= 1, \\ y(0) &= 0, \\ x(k+1) &= x(k) - y(k) * dt, \\ y(k+1) &= y(k) + x(k) * dt, \end{aligned}$$

which can be written as a vector equation (we put the matrix on the right so we can use row vectors)

$$(x, y)(0) = (1, 0),$$

$$(x, y)(k+1) = (x, y)(k) \begin{pmatrix} 1 & dt \\ -dt & 1 \end{pmatrix},$$

so if we write I for the identity matrix and J for the matrix

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

then we have (with X = (x,y))

$$\begin{aligned} X(0) &= (1, 0), \\ X(k+1) &= X(k)(I + J * dt), \end{aligned}$$

so

$$X(k) = (1, 0)\ (I + J * dt)^{\mathrm{k}},$$

which can be computed exactly.

Since the eigenvalues of $(I + J * dt)$ are $1 \pm i * dt$, which have magnitude $1 + dt^2$, the successive powers of the matrix diverge for any $dt > 0$, and therefore so does the iteration.

5.1.2 Second-Order Example

In this section, we use the same differential equation problem, with a different solver, a second-order one that is almost able to converge properly. We therefore have

$$x' = -y,$$

4.4.3 Truth Maintenance

Because we do not presume that the characterizers in a system are truths, we need to be much more careful about when they can be used together, especially in the inference and prediction processes. Since the inference rules themselves are time dependent, we need to keep track of the dependencies of every characterizer, both how and when it was derived (how tells us about hypotheses and inference rules; when helps us in checking temporal consistency) and its interval of activity.
We also need a way to indicate which characterizers we DO want to be true, so that different collections of characterizers can be compared and contrasted within the same context. We might want to consider computing various maximal consistent sets of irredundant assertions as an aid in this process.
Various rules can be activated that lead to new conclusions in an interval, which can supersede old ones; we also assume partial deduction, not total. We therefore need to use some kind of non-monotonic logic.

4.5 Analysis

Simulation is a continuing surprise.
We want tools with analytic power to help reduce our reliance on simulation, so we can make reliable predictions about the system behavior.
All of our computations are performed from the symbols active at a given time. The advantage of dealing explicitly with time in this formulation is that we can sit outside the usual sequencing of events, taking a kind of "side-long" look at the entire time line, and piece together parts of the models that we know more about regardless of whether or not they are the first ones in our time interval of interest.
We can also perform the deductions in an order that is different from the order imposed by time, using any of a number of simple mechanisms, such as rule-based systems or rewrite logics; both are being investigated.

5 Examples

This section contains several examples that illustrate the utility of the notation.

5.1 ODE

A simple example that shows range extension is an ordinary differential equation (ODE). For ODEs, the solution method is part of changing an ODE into a set of characterizers.
So let us consider a simple second-order ODE for the sine function,

$$\begin{aligned} y'' &= -y, \\ y'(0) &= 1, \\ y(0) &= 0, \end{aligned}$$

and solve it with Euler's method (a particularly bad one for this kind of problem, by the way).
First, we transform the equations into a first order system (in the usual way) by taking $x = y'$,

$$\begin{aligned} x' &= -y, \\ y' &= x, \\ x(0) &= 1, \\ y(0) &= 0, \end{aligned}$$

and we also define $z = x' = y''$.

5.1.1 First-Order

Now the way Euler's method works is by linear extrapolation, so for a given time $t = t_0$, if we have

$$\begin{aligned} x(t_0) &= x_0, \\ y(t_0) &= y_0, \end{aligned}$$

then we have

$$z_0 = z(t_0) = -y_0,$$

and we take

$$\begin{aligned} x(t) &= x_0 + z_0 * (t - t_0), \\ y(t) &= y_0 + x_0 * (t - t_0), \end{aligned}$$

and

$(a : w; t_1, t_2)$,

we normally want smoothness, written

$$\left.\frac{\mathrm{d}\ v}{\mathrm{d}\ t}\right|_{t=t_1^-} = \left.\frac{\mathrm{d}\ w}{\mathrm{d}\ t}\right|_{t=t_1^+},$$

and continuity, written

$v(t = t_1^-) = w(t = t_1^+)$.

Both of these are point conditions on the attributes and their derivatives, and we can consider only conditions on attributes by using whatever derivatives are needed in the conditions: instead of

$(a : v; t_0, t_1)$,

we use

$(a : (v, v'); t_0, t_1)$,

and write our smoothness condition as

$$\begin{pmatrix} v \\ v' \end{pmatrix}_{t=t_0^-} = \begin{pmatrix} w \\ w' \end{pmatrix}_{t=t_0^+}$$

If we also require continuity in each attribute, so that

$w(t = t_1^+) = w(t = t_1)$,

then the upper limit in the previous expression can be omitted.

It is therefore clear that we must deal with point events at transitions

$[t_0 ... t_1)\,[t_1 ... t_2)$,

but not with point characterizers. If we make the transition continuity a property of the definition of continuation, then we can assert it or not in any given model.

Of course, the expression $t = t_1^-$ means that the interval $[t_1 - \epsilon, t_1)$ is part of the limit computation for every ϵ small enough, so we might be able to use these intervals for some small enough ϵ without having to take the limits.

We will deal with these considerations in the simplest way possible. We have a characterizer that asserts continuity of the relevant attribute across a larger interval, such as $[t_0, t_2)$ above. The only place that the continuity characterizer has new information is at the transition point t_1, but we simply do not worry about the redundancy.

4.4 Characterizer Semantics and Inference

A characterizer is what we want to assume about what is true over its interval. It need not be consistent with the other characterizers in a system description; we explicitly allow false assertions here, so we can reason using counterfactuals.

4.4.1 Inference

We can make inferences within intervals, according to some rules. If, say, there is a rule

$s_1 \& s_2 \Longrightarrow s_3$,

and two characterizers

$(v : s_1; t_0, t_1)$

and

$(v : s_2; t_2, t_3)$

with $t_0 < t_2 < t_1 < t_3$, then we can conclude

$(v : s_3; t_2, t_1)$.

4.4.2 Prediction

We can also make inferences that extend intervals in some cases. They take the form: If

$(v : s_1; t_0, t_1)$

and

$(w : s_2; t_0, t_1)$

are characterizers with $t_0 < t_1$, then there is a characterizer

$(x : s_3; t_2, t_3)$

for some t_2, t_3, with $t_0 < t_2 < t_1 < t_3$.

Rules can contain variable identifiers, with implicit universal quantification.
Relationships hold on intervals and the combination may extend the range. We generate new characterizers according to the relationships, either predictive (range extension) or deductive (knowledge extension).
The language in which the rules are written is important, since it has to accommodate notations from many different types, many of which will not be known when the language is defined. Some basic concepts that will be in any of these languages are continuity and derivatives.
It is important to remember that the system comes first, and that the state variables are our choices for modeling and understanding the system. This means in particular that the coordinate systems we use are temporary, and that the constraints among the state variables are expressed explicitly as relationships.

4.2 Normalization and Continuation

Characterizers may have overlapping intervals. *Normalization* is the process of breaking each characterizer into two or more others, to fit the time scale. If t is an event time, and

$(a : v; s, e)$

is a characterizer with $s < t \leq e$, then we can replace it with two characterizers

$(a : v; s, t)$ and $(a : v; t, e)$.

If two characterizers use the same attribute,

$(a : v; s, e)$

and

$(a : w; t, u)$,

then we say that the second one *continues* the first one iff they are adjacent in time, so $t = e$. Continuity considerations in the transition from v to w at time t are treated in the next section.
In any system with a finite density of event times, if we split every characterizer that spans an event time, then we end up with characterizers that start and stop at consecutive event times (though they may be *continued* by other characterizers). This has some computational conveniences.
If we have two characterizers

$(a : v; t_1, t_2)$

and

$(a : w; t_2, t_3)$,

so that the second one continues the first, then we need some kind of explicit characterizer for the transition, active in an interval containing the transition time. If there is a description u in an appropriate domain for which

$$u = \begin{cases} v, & \text{for } t_1 \leq t < t_2, \\ w, & \text{for } t_2 \leq t < t_3, \end{cases}$$

then we can conclude

$(a : u; t_1, t_3)$.

This is the opposite of normalization.
If there is an overlap, that is, if the two characterizers

$(a : v; t_1, t_2)$

and

$(a : w; t_3, t_4)$

have

$[t_1, t_2) \cap [t_3, t_4)$ non-empty,

and

$v(t) = w(t)$ for $t \in [\max(t_1, t_3), \min(t_2, t_4))$,

then we can also conclude

$(a : u; \min(t_1, t_2), \max(t_3, t_4))$.

4.3 Continuation and Continuity

One aspect of continuity is transitions from one symbol to another across interval boundaries. The transition relations are extra conditions that have to hold *at* the transition time (usually they are smoothness conditions for model transitions).
A typical smoothness property is infinitesimal: for characterizers

$(a : v; t_0, t_1)$

An *attribute identifier* is a name for a state variable (a state variable is like a probe into some aspect of the system behavior, and the attribute identifier is only the label).

3.4 Expression

An *expression* is a pair

(attribute identifier: symbol),

which is interpreted to mean the assertion that the state variable can be described by the symbol (when the expression is active). We will describe the precise semantics of these expressions later on.
These are models of the state variable values.

3.5 Interval

An *interval* is a pair

[start time, end time),

assumed to describe a half-open interval (to save us from trouble with the topology). The end time may be omitted, in which case it is interpreted to mean infinity by default.

3.6 Characterizer

A *characterizer* is a pair

(expression, interval),

also written

(attribute identifier: symbol; start time, end time),

interpreted to mean that the expression is *active* during the specified interval. It becomes active at the start time, and becomes inactive at the end time. Each characterizer has a *range* (its *interval of activity*) and a *scope* (the set of attribute identifiers that occur in its expression).
We may also consider a symbol set that includes arithmetic expressions that contain an explicit time variable t. For example,

$(p : p_0 + v_0 * t; t_0, t_1)$

represents a continuous change along the interval.
We will also have occasion to reason about conditions at particular points in time, so the assertion language will also have characterizers of the form

(expression, point).

3.7 Event

An *event* is the activation or deactivation of a characterizer. We make no limiting assumptions about simultaneous events.

4 System Description

A *system description* is a finite set of characterizers, so we assume explicitly that a system can be described by a finite set of characterizers. We insist that only a finite set of characterizers be active at any one time. Since each of those characterizers is active over a positive interval, there is therefore some small interval thereafter during which all of them are still active.
Everything we know about a system's behavior is described by characterizers and relationships among the characterizers. Domain models and context can be written as characterizers, generally with large intervals.

4.1 Dynamics

Relationships among characterizers are rules that define the dynamics. These rules take the form:

if *these* characterizers (with a list) are active on *these* intervals, then *this* new one is also active on *this* other interval (not necessarily contained in the intersection of the original intervals).

1 Introduction

Traditionally, systems have been modelled using state variables defined in a metric space and the system dynamics defined using differential equations. This approach uses continuous descriptions of space and time. When we use computers for expressing and manipulating such models we have to use symbols to represent it. Symbols are discrete by their very nature, and require use of mapping from the continuous spaces to discrete spaces. These mappings cause problems unless carried out rather carefully. Further, when we consider the problems in which some aspects of the system are genuinely discrete, hybrid models have been used. As different techniques have to be used for continuous and discrete aspects of the system, significant complexity gets added to such models.
Recognizing that the computer systems only use symbols for any representations, in this paper we present a formulation of system dynamics directly in terms of symbols. In order to handle the synamics, time interval over which a symbol is considered valid is explicitly attached. The symbols describing different aspects of the system may be from a set appropriate for that aspect. The dynamics is described in terms of rules connecting the symbolic representations.
This paper contains the preliminary formulation of system dynamics in the framework of *Symbol Dynamics*.

2 Descriptions of System Behavior

For the purposes of this paper, *behavior* includes all the relationships among parts of a system at the same or different times. In particular, the combined relationships among parts of a system at the same time is usually called *structure*. Both of these aspects are subsumed in our use of the term behavior.
We assume that our ability to generate or derive new information about the system behavior changes only at discrete points in time, since we expect to perform these processes on digital computers. The event times define the time scale. In this paper, we introduce *Symbol Dynamics*, a totally symbolic way to represent the important aspects of dynamical systems and processes, so that we can reason about them using computers.

3 Concepts and Notations

This section contains the basic notions of Symbol Dynamics.

3.1 State Variable

We assume that systems exist and change over time. We are looking for a method of describing those changes so we can compute how to control them.
The systems we consider can be described with state variables. Each state variable is an observation on the system or a derivation from other state variables.
We may or may not know a priori which state variables are important, or even which ones are determinable (i.e., the system comes first, and the state variables are chosen to be helpful in describing the behavior). We might call the state variables *attributes* of the state.

3.2 Symbol

We want to measure and compute with information about a system, so we need to map the system into formal spaces we understand better.
A *type* is a symbol set, both representing a set of values and including some operations on those values; this is the notion of formal space used here. It includes collections of mutually dependent types and functions between different types.
A *symbol* of a given type is an element of the set of values that type. Any notions of credibility, confidence, or uncertainty are part of the type system that is used. It is especially important to define the allowable operations on these kinds of types. For example, for measurements of a system, the symbol would include the measured value and the associated uncertainty value.

3.3 Attribute Identifier

We assume that we will want to know different things about the system behavior. We need names to keep track of the different things we measure or compute.

Contents

Notes on Symbol Dynamics*†

Ashok K. Agrawala
Department of Computer Science, University of Maryland
College Park, Maryland 20742
E-mail: agrawala@cs.umd.edu

Christopher Landauer
System Planning and Development Division, The Aerospace Corporation
The Hallmark Building, Suite 187, 13873 Park Center Road, Herndon, Virginia 22071
Phone: (703) 318-1666, FAX: (703) 318-5409
E-mail: cal@aero.org

13 February 1995

Abstract

This paper introduces a new formulation of dynamic systems that subsumes both the classical discrete and differential equation models as well as current trends in hybrid models. The key idea is the express the system dunamics using symbols to which the notion of time is explicitly attached. The state of the system is described using symbols which are active for a defined period of time. The system dynamics is then represented as relations between the symbolic representations.
We describe the notation and give several examples of its use.

*This work is supported in part by ONR and DARPA under contract N00014-91-C-0195 to Honeywell and Computer Science Department at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, ONR, the U.S. Government or Honeywell.
Computer facilities were provided in part by NSF grant CCR-8811954.

†This work is supported in part by ARPA and Philips Labs under contract DASG60-92-0055 to Department of Computer Science, University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, PL, or the U.S. Government.