ON CONSTRAINT-ORIENTED ENVIRONMENTS FOR CONTINUOUS SYSTEMS SIMULATION

Richard A. Huntsinger

March 1988
CSD-880018

On Constraint-Oriented Environments for Continuous Systems Simulation*

Richard A. Huntsinger
Computer Science Department
University of California, Los Angeles
Los Angeles, California 90024-1600

March 18, 1988

Sets of simultaneous differential equations and sets of queries on those equations are naturally expressible as *constraint networks* in the *constraint satisfaction* modeling paradigm. Further, relaxation enhanced to exploit *typed valued* constraints provides a procedural semantics for such constraints which in the best case reduces to propagation, and in the worst case performs comparably to other paradigms. Accordingly, constraint satisfaction is advocated as the paradigm of choice on which to base continuous systems simulation environments.

Examples are presented illustrating constraint network characterizations of continuous systems models, and their corresponding procedural semantics.

*Supported by Tangram project, DARPA Contract F29601-87-C-0072

1 Overview

Attractive modeling environments provide facilities for representing models and making queries regarding the behavior of these models. Often queries take the form, colloquially expressed, "Which parameter instantiations will cause the model to behave in this desirable way?" The *constraint satisfaction* modeling pardigm has recently been adopted by various research efforts as the basis for the development of such modeling environments, as in [3].

The constraint satisfaction modeling paradigm operates on models represented as *constraint networks*, where a constraint network is comprised of several *constraints*. Each constraint specifies a relationship between several *objects*, which are temporally instantiated to various *values*. A constraint is *satisfied* when the specified relationship applied to the values to which the objects are instantiated is true.

For example, the constraint $X^{(1)}(0) = X^{(0)}(1) - X^{(0)}(0)$ specifies a relationship between the three objects $X^{(1)}(0)$, $X^{(0)}(1)$, and $X^{(0)}(0)$. When the objects are instantiated to the values 5, 20, and 15, respectively, then the constraint is satisfied because the relationship $5 = 20 - 15$ is true.

The constraint satisfaction modeling paradigm is generalizable to allow constraints to be *typed* and *valued*. A constraint is typed when it is associated with some *type value*. A constraint is valued when its range is not restricted to two truth values, but extended to some partial order reflecting degrees of truth.

For example, the constraint $X^{(1)}(0) = X^{(0)}(1) - X^{(0)}(0)$ may be typed so as to be associated with a type value of 1, 2, or 3. It may be valued so as report the difference between the two sides of the equation, where a difference of zero reflects the highest degree of truth.

In the context of this paradigm, the procedural semantics is an algorithm which, when applied to a constraint network, attempts to instantiate objects to values such that the accumulated value of all of the constituent constraint values is minimized.

There are two primary advantages of constraint satisfaction over competing modeling paradigms.

1. Queries are naturally expressed.
2. A single procedural semantics handles all queries, so that construction of specialized algorithms for certain queries is not required.

2 Enhanced Relaxation

Under consideration is the procedural semantics provided by relaxation, which iterates as follows:

Step 1: Select an object which has not been selected more recently than any other object.

Step 2: Acquire a function which returns a value for the selected object given values for some other objects, such that the returned and given values result in some constraint being satisfied. Typically the function is supplied by the modeler along with the constraints.

Step 3: Instantiate the selected object to the valuation of the function given the instantiations of objects on which it depends.

Relaxation can be enhanced to improve its performance by exploiting the additional information provided by typed valued constraints. The simple enhancement to relaxation is to replace the object selection criterion with a function on the types and values of the constraints. The procedural semantics is then succinctly captured in the following code:

while $accumulated\_value(C) > \epsilon$ **do**
$C_i \leftarrow select\_typed\_valued\_constraint(C)$
$A \times F \leftarrow select\_object\_\&\_function(C_i)$
$A \leftarrow evaluate(F)$
endwhile

Relaxation on typed valued constraints has proved an appropriate procedural semantics in the graphics domain, as described in [5], where the degrees of truth are analogized to energy quantities, and the iterative instantiation of objects is analogized to the minimization of the energy of a system. The examples discussed presently suggest that it may be equally appropriate in the continuous systems simulation domain.

3 Constraint Network Characterization

Under consideration are continuous systems models expressible as sets of m differential equations on m variables and their derivatives, where the highest order derivative n_i for each equation i is explicitly isolated.

$$X_i^{(n_i)} = \phi_i \left(\begin{array}{l} X_0^{(n_0-1)}, X_0^{(n_i-2)}, \ldots, X_0^{(0)}, \\ X_1^{(n_1-1)}, X_1^{(n_1-2)}, \ldots, X_1^{(0)}, \\ \ldots, \\ X_m^{(n_m-1)}, X_m^{(n_m-2)}, \ldots, X_m^{(0)} \end{array} \right)$$

Such sets are expressible as constraint networks in the following way. Each equation i can be equivalently expressed as a set of n_i+1 types of constraints on $(n_i + 1)t_f$ objects, where t_f is the largest time of interest. n_i of the types correspond to the implicitly expressed knowledge that for each time $t|\ 0 \leq t < t_f$ and each order $d|\ 0 < d \leq n_i$, adjacently ordered variable instantiations preserve the relationship $X_i^{(d)}(t) = X_i^{(d-1)}(t+1) - X_i^{(d-1)}(t)$.

The remaining type corresponds to the explicitly expressed knowledge that for each time $t|\ 0 \leq t \leq t_f$, variable instantiations preserve the equation i. Each object corresponds to some variable $X_i^{(d')}(t)$, where $0 \leq d' \leq n_i$ and $0 \leq t \leq t_f$.

4 Example: Initial Condition Problem

Figure 1 illustrates a single-variable, second-order differential equation expressed as a constraint network of three types of constraints. Boxes represent objects, circles represent constraints, and arrows indicate on which objects the various constraints operate. The numbers in the circles indicate the type of constraint:

Type 1: A constraint of the form $X^{(1)}(t) = X^{(0)}(t+1) - X^{(0)}(t)$, where $0 \leq t < t_f$

Type 2: A constraint of the form $X^{(2)}(t) = X^{(1)}(t+1) - X^{(1)}(t)$, where $0 \leq t < t_f$

Type 3: A constraint of the form $X^{(2)}(t) = \phi(X^{(1)}(t),\ X^{(0)}(t))$, where $0 \leq t \leq t_f$

The degree of truth for each constraint is taken to be the difference between the valuations of the left and right sides. For example, the constraint $X^{(1)}(0) = X^{(0)}(1) - X^{(0)}(0)$, when objects $X^{(1)}(0)$, $X^{(0)}(1)$, and $X^{(0)}(0)$ are instantiated to 5, 20, and 6, respectively, takes on a degree of truth of 11. Uninstantiated objects are avoided by instantiating them to some special value **unknown**. A constraint which operates on at least two objects instantiated to **unknown** takes on a degree of truth of **veryfalse**. A constraint which operates on one object instantiated to **unknown** takes on a degree of truth of **false**. Here **veryfalse** is lower than **false**, and both are lower than all other degrees of truth.

The *select_typed_value_constraint* function operates as follows.

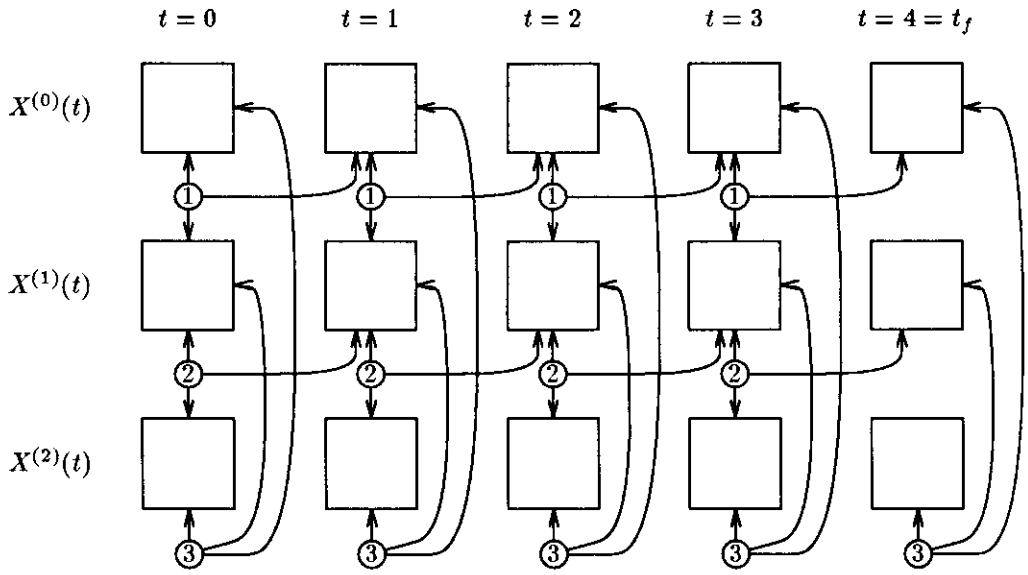

Figure 1: *Single-variable, Second-order Differential Equation*

Step 1: Consider all constraints except those with degrees of truth of **veryfalse**.

Step 2: If there are any constraints being considered with degrees of truth of **false**, then consider only those.

Step 3: If there are any constraints being considered of type 3, then consider only those.

Step 4: If there are any constraints being considered of type 2, then consider only those.

Step 5: If there are any constraints being considered of type 1, then consider only those.

Step 6: Of the constraints being considered, determine those with the single lowest degree of truth (highest real number), and then consider only those.

Step 7: Of the constraints being considered, select the constraint which has least recently been selected.

The *select_object_&_function* function operates as follows.

Step 1: If the constraint is of type 3, then return the left side of the equation as the object, and return the right side of the equation as the function.

Step 2: If the constraint is of type 1 or 2, and the degree of truth is **false**, then determine a new equation which preserves the equality expressed in the constraint, such that the left side of the new equation is an object instantiated to **unknown**. Return the left side of the new equation as the object, and return the right side of the equation as the function.

The initial condition problem for a single-variable, second-order differential equation can be expressed as a constraint network as just described, where the objects $X^{(0)}(0)$ and $X^{(1)}(0)$ are instantiated to values corresponding to initial conditions. All other objects are instantiated to the value **unknown**.

Application of the procedural semantics just described results in all objects instantiated to **unknown** to become newly instantiated to real values. The new instantiations occur in the order $X^{(2)}(t)$, then $X^{(1)}(t+1)$, then $X^{(0)}(t+1)$, for $0 \leq t < t_f$. Note that for the initial condition problem, the procedural semantics is identical to propagation.

5 Example: Shooting Problem and More

Figure 2 illustrates the constraint network of the initial condition example extended to reflect fourth and fifth types of constraints. This example addresses the opening comments made regarding queries.

Under consideration first is the shooting problem, where models are expressible as differential equations with known initial and non-initial conditions. In practice, such models are typically constructed so that a query can be made regarding its behavior, specifically, colloquially expressed, "Which

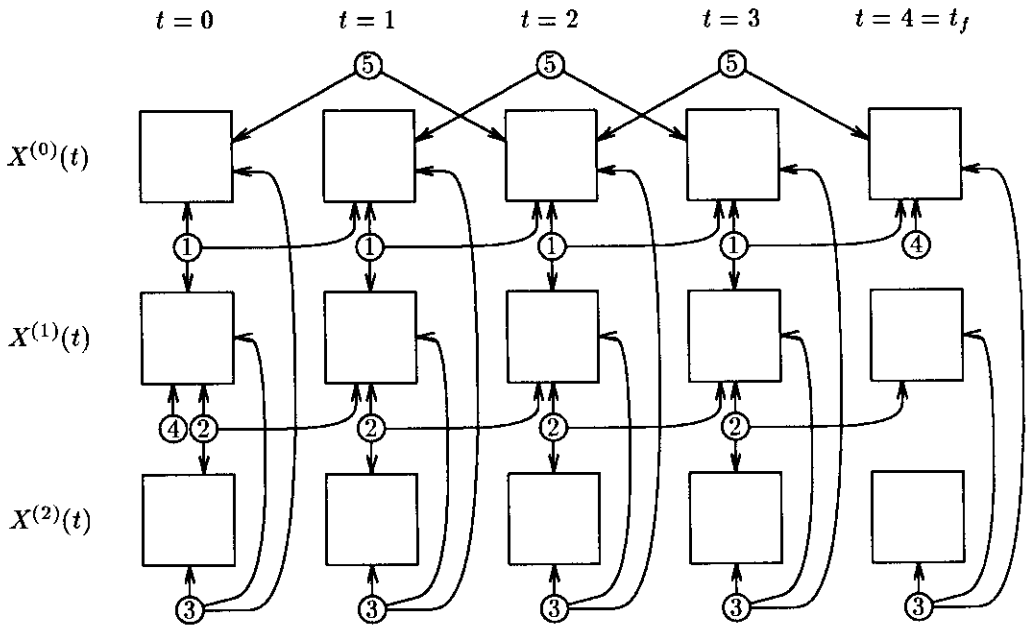

Figure 2: *Extension for Shooting and Oscillation Control*

values for initial conditions not known will cause the model to satisfy the known non-initial conditions?"

The shooting problem for a single-variable, second-order differential equation can be expressed as a constraint network, where objects $X^{(1)}(0)$ and $X^{(0)}(t_f)$ are instantiated to values corresponding to known conditions. The object $X^{(0)}(0)$ is instantiated to some guessed value. All other objects are instantiated to the value **unknown**. Also, additional constraints of a fourth type must be included of the form $X^{(1)}(0) = \alpha$ and $X^{(0)}(t_f) = \beta$, to prevent the instantiations of these objects from permanently changing.

Application of the procedural semantics described, augmented to account for constraints of the fourth type, results in propagation behavior until immediately following the instantiation of $X^{(1)}(3)$. Then, since both of the constraints $X^{(1)}(2) = X^{(0)}(3) - X^{(0)}(2)$ and $X^{(1)}(3) = X^{(0)}(4) - X^{(0)}(3)$ are applicable, and since the latter has been selected less recently, the latter is selected. Objects previously instantiated to real values subsequently become reinstantiated to other real values. The order in which they are

reinstantiated depends on how the procedural semantics have been augmented.

The shooting problem example motivates a generalization of the query to the following: "Which values for initial conditions not known will cause the model to satisfy all of a set of constraints?" Here the set of constraints can represent any relationship between variables, not just non-initial consitions.

For example, constraints of a fifth type can reflect a relationship between values of variables at regular time intervals, perhaps of the form $X^{(0)}(t) = X^{(0)}(t+2)$, corresponding to a query of the form, "Which values for initial conditions not known will cause the model to behave such that the lowest order derivative cycles with a period of 2?"

Empirical results for some simple non-linear constraint network testbeds suggest that for many practical models, the instantiations converge appropriately, reasonably quickly.

6 Conclusions

Models expressible as sets of simultaneous differential equations and sets of queries on those equations have been shown to be equivalently naturally expressible as constraint networks in the constraint satisfaction modeling paradigm. In the latter form, all queries are handled by a single procedural semantics, so that construction of specialized algorithms for certain queries is not required. This, along with an investigation of some simple examples, suggests that constraint satisfaction may be the paradigm of choice on which to base continuous systems simulation environments.

References

[1] Leler, William. *Specification and Generation of Constraint Satisfaction Systems*, Ph.D. Dissertation, Department of Computer Science,

University of North Carolina at Chapel Hill, Technical Report 87-006, 1987.

[2] Parker, D. Stott, Jr. *Partial Order Programming*, Department of Computer Science, University of California, Los Angeles, Technical Report CSD-870067, 1987.

[3] Parker, D. Stott, Jr., Richard R. Muntz, & Gerald Popek. *Tangram*, Department of Computer Science, University of California, Los Angeles, Unpublished, October 1987.

[4] Steele, Guy Lewis, Jr. *The Definition and Implementation of a Computer Programming Language Based on Constraints*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Technical Report AI-TR 595, August 8, 1980.

[5] Witkin, Andrew, Kurt Fleischer & Alan Barr. "Energy Constraints on Parameterized Models," *SIGGRAPH '87 Conference Proceedings*, July 27-31, 1987.