REPRESENTATION TRANSFORMATION IN CONSTRAINT SATISFACTION SYSTEMS

Richard A. Huntsinger

March 1988
CSD-880020

Representation Transformation in Constraint Satisfaction Systems*

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

March 7, 1988

A practical class of constraint satisfaction systems operate on *relaxable representations* of the form $N = f(N)$, where N is a set of variables, and the declarative semantics is the set of instantiations of N which preserve the equality. In general, relaxation provides a complete procedural semantics for only a subset ρ of such representations. Of interest, then, is the set of *transformable* representations $\alpha \supset \rho$ in which for each representation $M_r \in \alpha$ there exists a determinable transformation $T : \alpha \rightarrow \rho$ such that the declarative semantics of M_r is identical to that of $T(M_r)$.

Relaxable representations for which $f(N)$ is a polynomial are transformable, each corresponding to a transform of the form $N = (f(N)N^n)^{\frac{1}{n+1}}$, where n is a function of the degree and coefficients of the polynomial. This observation provides some intuition about more general transformations, applicable to the implementation of powerful (complete over a superset of ρ) constraint satisfaction systems.

Topic: Knowledge Representation — Paper Length: 2200 words

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

1 Constraint Satisfaction Paradigm

Constraint satisfaction is a modeling paradigm in which model representations are often easily generated, as noted in [9]. A formal generalization of such model representations illustrates that while the intuitive declarative semantics is appealingly simple, a correspondingly simple procedural semantics is not complete.

1.1 Representations

Within the constraint satisfaction paradigm, model representations assume the form of duples $\langle \chi, \gamma \rangle$, where χ is a *constraint network* and γ is a *goal*. Each *constraint* χ_i, an element of χ, specifies the value of a function C_i on variables N_i in terms of those variables, or in terms of (possibly zero) other functions on subsets of those variables, for all possible instantiations. The goal γ implicitly specifies a subset of all possible instantiations by restricting the function of a distinguished constraint C_1 to a value R.

$$\chi = \left\{ \begin{array}{l} C_1(N_1) = F_1(N_1, C_{j\neq 1}(N_j), C_{k\neq 1}(N_k), ..., C_{n\neq 1}(N_n)) \\ C_2(N_2) = F_2(N_2, C_{j'\neq 2}(N_{j'}), C_{k'\neq 2}(N_{k'}), ..., C_{n'\neq 2}(N_{n'})) \\ \ldots \\ C_m(N_m) = F_m(N_m, C_{j''\neq m}(N_{j''}), C_{k''\neq m}(N_{k''}), ..., C_{n''\neq m}(N_{n''})) \end{array} \right\}$$

$$\text{where} \quad \begin{array}{l} N_j \subseteq N_1,\ N_k \subseteq N_1,\ ...,\ N_n \subseteq N_1 \\ N_{j'} \subseteq N_2,\ N_{k'} \subseteq N_2,\ ...,\ N_{n'} \subseteq N_2 \\ \ldots \\ N_{j''} \subseteq N_m,\ N_{k''} \subseteq N_m,\ ...,\ N_{n''} \subseteq N_m \end{array}$$

$$\gamma = \{C_1(N_1) = R\}$$

As an example, consider the representation for the real-valued, five-node, linear Laplacian, bounded at one end by the value B_L and at the other end by the value B_R, where the value of each internal node is the average of the values of its neighbors.

$$\chi = \left\{ \begin{array}{l} C_1(\{X_1, X_2, X_3\}) = C_2(\{X_1, X_2\}) + C_3(\{X_1, X_2, X_3\}) + C_4(\{X_2, X_3\}) \\ C_2(\{X_1, X_2\}) = \left| X_1 - \dfrac{B_L + X_2}{2} \right| \\ C_3(\{X_1, X_2, X_3\}) = \left| X_2 - \dfrac{X_1 + X_3}{2} \right| \\ C_4(\{X_2, X_3\}) = \left| X_3 - \dfrac{X_2 + B_R}{2} \right| \end{array} \right\}$$

$$\gamma = \{C_1(\{X_1, X_2, X_3\}) = 0\}$$

Augmented representations comprise a subset of such model representations. They may assume the form of duples $\langle \chi', \gamma \rangle$, where χ' is an *augmented constraint network* and γ is a goal as defined before. Each constraint χ'_i, an element of χ', specifies the value of a function C_i on variables N_i as defined before, but also specifies a (possibly null) set of functions f_i. Each function f_{il}, an element of f_i, specifies the value of a single variable N_{il} in terms of the variables N_i, so as to commit the function C_i to the value R_{il}.

$$\chi'_i = \left\{ \begin{array}{l} C_i(N_i) = F_i(N_i, C_{j \neq i}(N_j), C_{k \neq i}(N_k), ..., C_{n \neq i}(N_n)) \\ \left\{ \begin{array}{l} N_{il} = f_{il}(N_i) \Rightarrow \; C_i(N_i) = R_{il} \\ N_{il'} = f_{il'}(N_i) \Rightarrow \; C_i(N_i) = R_{il'} \\ \ldots \\ N_{il''} = f_{i1''}(N_i) \Rightarrow \; C_i(N_i) = R_{il''} \end{array} \right\} \end{array} \right\}$$

Relaxable representations comprise a subset of augmented representations. They may assume the form of duples $\langle \chi', \; \gamma \rangle$ as defined before, but further, for each variable N_{1l}, an element of N_1, there must exist at least one corresponding function of the form $N_{1l} = f_{ll'}(N_l) \Rightarrow C_l(N_l) = R_{ll'}$, an element of some f_l for some χ_l, such that $C_l(N_l) = R_{ll'}$ is sufficient to ensure $C_1(N_1) = R$ according to criteria dictated by F_1. Under this condition, these corresponding functions can be consolidated into a single function of the form $N_1 = f(N_1) \Rightarrow C_1(N_1) = R$.

As an example, consider the relaxable representation for the LaPlacian described earlier, where the set N_1 is expressed as a vector.

$$[X_1, X_2, X_3] = \left[\frac{B_L + X_2}{2}, \frac{X_1 + X_3}{2}, \frac{X_2 + B_R}{2} \right] \Rightarrow C_1([X_1, X_2, X_3]) = 0$$

In practice, constraint satisfaction systems typically restrict representations to relaxable representations, e.g., [1,10], can be viewed this way.

1.2 Declarative Semantics

The inituitive declarative semantics $S_d(M)$ for a model representation M is the subset of all possible instantiations I simultaneously satisfying the constraint network and the goal. For a relaxable representation M_r, this is equivalent to the subset of all possible instantiations I satisfying the consolidated function $N_1 = f(N_1) \Rightarrow C_1(N_1) = R$.

$$s \in S_d(M_r) \models M_r \iff C_1(s) = R \iff s = f(s)$$

Definition of Declarative Semantics

1.3 Procedural Semantics

In practice, constraint satisfaction systems which restrict representations to relaxable representations typically employ relaxation as the procedural semantics $S_p(M_r, s', \epsilon, \delta)$ for a relaxable representation M_r, where the initial guess s' is an element of all possible instantiations I, the allowable error ϵ is an element of some partially ordered domain D, and the distance function δ maps pairs of instantiations from $I \times I$ to the domain D.

$$\begin{pmatrix} s' \in I \\ \epsilon \in D \\ \delta : I \times I \rightarrow D \end{pmatrix} \Longrightarrow$$

$$\left(s \in S_p(M_r, s', \epsilon, \delta) \vdash M_r \iff \exists_{k \geq 0} \middle| \, \delta(f^k(s'), f^{k+1}(s')) \leq \epsilon \, \wedge \, f^k(s') = s\right)$$

Definition of Procedural Semantics

1.4 Complete Procedural Semantics and Convergent Relaxable Representations

A *complete* procedural semantics is one subsumed by the declarative semantics. Equivalently, under an allowable error $\epsilon \in D$ and distance function $\delta : I \times I \rightarrow D$, a complete procedural semantics assuming any initial guess $s' \in I$ results in some value s for N_1 satisfying $s = f(s)$.

Iterative relaxation provides a complete procedural semantics for only the set of *convergent* relaxable representations ρ, a subset of all possible relaxable representations.

$$M_r \in \rho \iff$$

$$\left(S_d(M_r) \neq \emptyset \Longrightarrow \left(\begin{pmatrix} s' \in I \\ \epsilon \in D \\ \delta : I \times I \rightarrow D \end{pmatrix} \Longrightarrow \emptyset \neq S_p(M_r, s', \epsilon, \delta) \subseteq S_d(M_r)\right)\right)$$

Definition of Complete Procedural Semantics

2 Representation Transformation

In practice, non-convergent relaxable representations often arise; relaxation provides a non-complete, and therefore inadequate, procedural semantics for such representations. *Representation transformation* is an abstract technique which addresses this problem. It can be combined with relaxation to provide a procedural semantics complete over convergent and at least some non-convergent relaxable representations.

In essence, if a relaxable representation M_r cannot be identified as convergent, an element of the set of identifiably convergent relaxable representations $\mu \subset \rho$, then perhaps M_r is *transformable*, an element of the set of transformable relaxable representations α, such that a transformation function $T : \alpha \rightarrow \rho$ can be determined and applied to M_r, preserving its declarative semantics $S_d(M_r)$.

$$(M_r \in \alpha \wedge M_r \notin \mu \subset \rho \subset \alpha) \implies \exists_{T:\alpha\rightarrow\rho} | S_d(T(M_r)) = S_d(M_r)$$

2.1 Approach for Determination of Transformation Functions

One approach for the determination of an appropriate transformation function is to identify the component of the function f which causes $N = f(N)$ to be non-convergent, and then use that information to selet a function t decomposable into a pair of functions g and $\bar{g}$ such that $\bar{g}(g(s,s),s) = s$, and $g(f(s),s)$ is identifiably convergent, and so is $\bar{g}(g(f(s),s),s)$.

$$\begin{array}{rcrcl}
s \in S_d(N_1 = f(N_1)) & \implies & s & = & f(s) \\
s \in S_d(N_1 = f(N_1)) & \implies & g(s,s) & = & g(f(s),s) \\
s \in S_d(N_1 = f(N_1)) & \implies & \bar{g}(g(s,s),s) & = & \bar{g}(g(f(s),s),s) \\
s \in S_d(N_1 = f(N_1)) & \implies & s & = & \bar{g}(g(f(s),s),s) \\
s \in S_d(N_1 = f(N_1)) & \implies & s & = & t(s)
\end{array}$$

2.2 Polynomials

It is known from [8] that if the set of all possible instantiations I is partially ordered, where $\perp \in I$ is the least element, then iterative relaxation assuming an intial guess $\perp$ is a complete procedural semantics for the set of relaxable representations involving monotonic functions.

This result can be applied toward the determination of transformations for relaxable representations involving polynomials over non-negative real values. Following the just proposed approach, $g(\cdot, s)$ is selected to be multiplication by s^n, where n is large enough to cause the influence of the highest degree term of $f(s)$ to overwhelm the influence of the lower degree terms, resulting in a monotonic function. Then, $\bar{g}(\cdot, \cdot)$ is selected to be the $(n+1)^{th}$ root. Since compositions of monotonic functions are monotonic, $\bar{g}(g(\cdot, \cdot), \cdot)$ is monotonic, and assuming an initial guess of $s = \perp = 0$ results in $s = t(s)$ as a convergent relaxable representation.

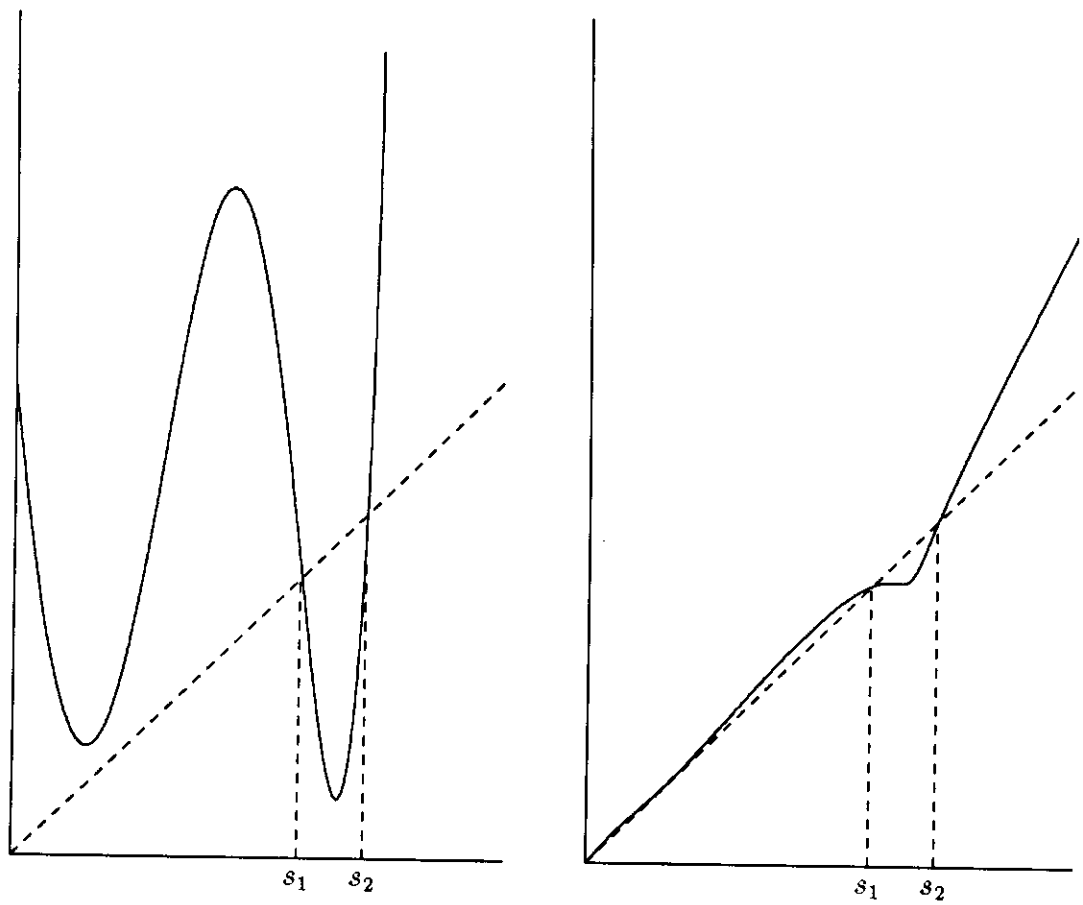
$$\begin{aligned} s &= A_d s^d + A_{d-1} s^{d-1} + \dots + A_1 s + A_0 \\ s s^n &= (A_d s^d + A_{d-1} s^{d-1} + \dots + A_1 s + A_0) s^n \\ s^{n+1} &= A_d s^{d+n} + A_{d-1} s^{d-1+n} + \dots + A_1 s^{1+n} + A_0 s^n \\ s &= (A_d s^{d+n} + A_{d-1} s^{d-1+n} + \dots + A_1 s^{1+n} + A_0 s^n)^{\frac{1}{n+1}} \end{aligned}$$

2.2.1 Example

The following parameters specify a non-monotonic sixth degree polynomial and its corresponding monotonic transformation.

$A_6 = 1.65 \times 10^{-11}$, $A_5 = -1 \times 10^{-8}$, $A_4 = 1.27 \times 10^{-6}$, $A_3 = -4 \times 10^{-5}$
$A_2 = 0.065$, $A_1 = -10$, $A_0 = 500$

$d = 6$, $n = 15$



A Polynomial ($d = 6$) and its Transform ($n = 15$)

2.2.2 Transform Degree

The monotonicity of the transformed function t ensures that any points with zero derivative must be a point of inflection.

$$x \in I \implies \left(\frac{dt}{ds}(s) = 0 \implies \frac{d^2t}{ds^2}(s) = 0 \right)$$

Solving for the minimum value of n which preserves this implication shows that n is a function of the coefficients and degree of the original function f. Though an appropriate minimum value of n always exists, its determination may be impractically computationally complex. Employment of a non-minimum value of n still preserves the implication.

3 Applications

Constraint satisfaction has been advocated in [4,11] as the modeling paradigm of choice for realistic solid-body animation, and likewise in [6] for automatic music composition. Investigation of transformation functions for polynomials is not intended to lead to a constraint satisfaction system for finding fixpoints of polynomials, but rather to gather intuition about how representation transformation can be applied to interesting domains such as animation and composition.

As an example, consider an animation preventing circular objects from moving through each other. If $\langle X_{it}, Y_{it}\rangle$ and $\langle X_{jt}, Y_{jt}\rangle$ are the coordinates of the centers of the respective objects i and j at time t, and B_i and B_j are their respective radii, and d is the distance between their edges, then the x-coordinate of object i can be expressed as a function of the other variables.

$$X_{it} = \pm\left((B_i + B_j + d)^2 - (Y_{it} - Y_{jt})^2\right)^{\frac{1}{2}} + X_{jt}$$

It is the relationship between the y-coordinates of the objects which renders the function non-monotonic, suggesting the existence of an appropriate non-monotonic-to-monotonic transformation function dependent on this relationship. This prospect is being explored as part of a constraint-satisfaction-oriented animation system development effort in [9].

4 Conclusions

In the context of the constraint satisfaction modeling paradigm, representation transformation may prove a practical improvement to the typically employed non-complete procedural semantics provided by iterative relaxation for relaxable representations. The successful application to polynomials of a proposed approach for determining transformation functions provides a promising point of departure toward the discovery of transformation functions for non-polynomial functions.

References

[1] Borning, Alan. "The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory," *ACM Transactions on Programming Languages and Systems*, Vol. 3, No. 4, October 1981.

[2] Dechter, Rina & Judea Pearl. *Network-Based Heuristics for Constraint-Satisfaction Problems*, Department of Computer Science, University of California, Los Angeles, Technical Report CSD-870037, 1987.

[3] Jaffar, Joxan & Jean-Louis Lassez. "Constraint Logic Programming," *Proceedings of 12th ACM Symposium on Principles of Programming Languages*, January 1987.

[4] Isaacs, Paul M. & Michael F. Cohen "Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics," *SIGGRAPH '87 Conference Proceedings*, July 27-31, 1987.

[5] 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.

[6] Levitt, David. "Machine Tongues X: Constraint Languages," *Computer Music Journal*, Volume 8, Number 1, Spring 1984.

[7] Nelson, Greg. "Juno, A Constraint-Based Graphics System," *SIGGRAPH '85 Conference Proceedings*, July 22-26, 1985.

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

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

[10] 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.

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