

# Constraint Programming in Constraint Nets

Ying Zhang  
Department of Computer Science  
University of British Columbia  
Vancouver, B.C.  
Canada V6T 1Z2  
zhang@cs.ubc.ca

Alan K. Mackworth \*  
Department of Computer Science  
University of British Columbia  
Vancouver, B.C.  
Canada V6T 1Z2  
mack@cs.ubc.ca

## Abstract

We view constraints as relations and constraint satisfaction as a dynamic process of approaching a stable equilibrium. We have developed an algebraic model of dynamics, called Constraint Nets, to provide a real-time programming semantics and to model and analyze dynamic systems. In this paper, we explore the relationship between constraint satisfaction and constraint nets by showing how to implement various constraint methods on constraint nets.

## 1 Motivation

Constraints are relations among entities. Constraint satisfaction can be viewed in two different ways. First, in the logical deductive view, a constraint system is a structure  $\langle D, \vdash \rangle$ , where  $D$  is a set of constraints and  $\vdash$  is an entailment relation between constraints [20]. In this view, constraint satisfaction is seen as a process involving multiple agents concurrently interacting on the store-as-constraint system by checking entailment and consistency relations and refining the system monotonically. This approach is useful in database or knowledge-based systems, and can be embedded in logic programming languages [2, 5, 9]. Characteristically, the global constraint is not explicitly represented, even though for any given relation tuple the system is able to check whether or not it is entailed.

An alternative view, more appropriate for real-time embedded systems, is to formulate the constraint satisfaction problem as finding a relation tuple that is entailed by a given set of constraints [12]. In this paper, we present an approach to this problem. In this approach, constraint satisfaction is a dynamic process with each solution as a stable equilibrium, and the solution set as an attractor of the process. “Monotonicity” is characterized by a Liapunov function, representing the “distance” to the set of solutions over time. Moreover, soft as well as hard constraints can be represented and solved. This approach has been taken in neural nets [18], optimization, graphical simulation [16] and robot control [15]; however, it has not yet been investigated seriously in the area of constraint programming.

We have developed and implemented an algebraic model of dynamics, called Constraint Nets (CN), to provide a real-time programming semantics [22] and to model and analyze robotic systems [23]. Here we investigate the relationship between constraint satisfaction and constraint nets. The rest of this paper is organized as follows. Section 2 describes some basic concepts of dynamic systems. Section 3 introduces Constraint Nets. Section 4 presents various constraint methods for solving global consistency and unconstrained optimization problems. Section 5 discusses embedded constraint solvers and some implementation issues. Section 6 concludes the paper.

## 2 Properties of Dynamic Systems

In this section, we review some basic concepts in metric spaces, dynamic systems and the relationship among stability, attractors and Liapunov functions.

---

\*Shell Canada Fellow, Canadian Institute for Advanced Research

## 2.1 Metric spaces

Let  $\mathcal{R}$  be the set of all real numbers and  $\mathcal{R}^+$  denote the set of all nonnegative real numbers. A *metric* on a set  $X$  is a function  $d : X \times X \rightarrow \mathcal{R}^+ \cup \{\infty\}$  which satisfies the following axioms for all  $x, y, z \in X$ :

1.  $d(x, y) = d(y, x)$ .
2.  $d(x, y) + d(y, z) \geq d(x, z)$ .
3.  $d(x, y) = 0$  iff  $x = y$ .

A *metric space* is a pair  $\langle X, d \rangle$  where  $X$  is a set and  $d$  is a metric on  $X$ . In a metric space,  $d(x, y)$  is called “the distance between  $x$  and  $y$ ”. Given a metric space, we can define the distance between a point and a set of points as:  $d(x, X^*) = \inf_{x^* \in X^*} d(x, x^*)$ .

For any point  $x^* \in X$  and  $\epsilon > 0$ , if  $\{x | d(x, x^*) \leq \epsilon\} \supset \{x^*\}$ , we call this set the  $\epsilon$ -neighborhood of  $x^*$ , denoted  $N^\epsilon(x^*)$ . Similarly, for any subset  $X^* \subset X$ , if  $\{x | d(x, X^*) \leq \epsilon\} \supset X^*$ , we call this set the  $\epsilon$ -neighborhood of  $X^*$ , denoted  $N^\epsilon(X^*)$ .

Let  $\mathcal{T}$  be a set of totally ordered time points which can be either discrete or continuous. A *trace*  $v : \mathcal{T} \rightarrow X$  is a function from a set of time points to a set of values. We use  $\mathcal{V}_{\mathcal{T}}^X$  to denote the set of all traces from  $\mathcal{T}$  to  $X$ . Given a metric space  $\langle X, d \rangle$  and a trace  $v$ , a trace  $v$  *approaches a value*  $x^* \in X$  iff  $\lim_{t \rightarrow \infty} d(v(t), x^*) = 0$ ;  $v$  *approaches a set*  $X^* \subset X$  iff  $\lim_{t \rightarrow \infty} d(v(t), X^*) = 0$ .

## 2.2 Dynamic systems

The term *dynamic* refers to phenomena that produce time-changing patterns, the characteristics of the pattern at one time being interrelated with those at other times [10]. A *process*  $p : X \rightarrow \mathcal{V}_{\mathcal{T}}^X$  is a function from a set of values  $X$  to a set of traces  $\mathcal{V}_{\mathcal{T}}^X$ . Intuitively,  $p$  characterizes a set of traces which are solely determined by their initial values. We use  $\phi_p(x)$  to denote the set of values in the trace of  $p(x)$ , i.e.  $\phi_p(x) = \{p(x)(t) | t \in \mathcal{T}\}$ .

A point  $x^* \in X$  is an *equilibrium* (or *fixpoint*) of the process  $p$  iff  $\forall t, p(x^*)(t) = x^*$ . An equilibrium  $x^*$  is *stable* [13] iff  $\forall N^\epsilon(x^*) \exists N^\delta(x^*) \forall x \in N^\delta(x^*) \phi_p(x) \subseteq N^\epsilon(x^*)$ .

A set  $X^* \subset X$  is an *attractor* [19] of the process  $p$  iff  $\exists N^\epsilon(X^*) \forall x \in N^\epsilon(X^*) \lim_{t \rightarrow \infty} d(p(x)(t), X^*) = 0$ ;  $X^*$  is an *attractor in the large* iff  $\forall x \in X \lim_{t \rightarrow \infty} d(p(x)(t), X^*) = 0$ . If  $\{x^*\}$  is an attractor (in the large) and  $x^*$  is a stable equilibrium,  $x^*$  is called an *asymptotically stable equilibrium* (in the large).

## 2.3 Liapunov functions

Let  $X^* \subset X$  and  $\Omega = N^\epsilon(X^*)$  for some  $\epsilon > 0$ . A *Liapunov function* for  $X^*$  and a process  $p : X \rightarrow \mathcal{V}_{\mathcal{T}}^X$  is a function  $V : \Omega \rightarrow \mathcal{R}$ , satisfying:

1.  $\forall x, x' \in \Omega, V(x) \leq V(x')$  iff  $d(x, X^*) \leq d(x', X^*)$ ,
2.  $\forall x \in \Omega \forall t, V(p(x)(t)) \leq V(x)$ .

The first condition states that  $V$  has a local minimum at  $x^* \in X^*$ . The second condition guarantees that  $V$  moves downhill along any traces of  $p$  starting at  $x \in \Omega$ . This definition is a simplified version of the one given in [10]. The following two theorems are similar to those in [10].

**Theorem 1** *An equilibrium  $x^* \in X$  of a process  $p$  is stable if there exists a Liapunov function  $V$  for  $\{x^*\}$  and  $p$ .*

Proof: Let  $\Omega$  be the domain of  $V$ , which is an  $\epsilon'$ -neighborhood of  $x^*$  for some  $\epsilon' > 0$ . Given an  $\epsilon$ -neighborhood  $N^\epsilon(x^*)$  of  $x^*$ , let  $\delta = \min(\epsilon, \epsilon')$ , we have a  $\delta$ -neighborhood  $N^\delta(x^*) \subseteq \Omega$ , therefore,  $\forall x \in N^\delta(x^*) \phi_p(x) \subseteq N^\epsilon(x^*)$ .  $\square$

**Theorem 2** *Suppose a process  $p$  satisfies the following condition: for any  $x^* \in X$ , if there is  $x$  such that  $p(x)$  approaches  $x^*$ , then  $x^*$  is an equilibrium. A set of stable equilibria  $X^* \subset X$  of the process  $p$  is an attractor if there exists a Liapunov function  $V$  for  $X^*$  and  $p$ , such that  $V$  satisfies the following conditions:*

1.  $V$  is continuous, i.e.  $d(x, x') \rightarrow 0$  implies  $|V(x) - V(x')| \rightarrow 0$ .
2.  $\forall x \in \Omega \forall t, V(p(x)(t)) < V(x)$  if  $x \notin X^*$ .

Furthermore, if  $\Omega = X$ ,  $X^*$  is an attractor in the large.

Proof: For any  $x \in \Omega$ , since  $V$  is continuous,  $\lim_{t \rightarrow \infty} V(p(x)(t)) = V(\lim_{t \rightarrow \infty} p(x)(t)) = V(x^*)$ . We can prove that  $x^* \in X^*$  since otherwise according to Condition 2,  $x^*$  cannot be an equilibrium. If  $\Omega = X$ ,  $X^*$  is an attractor in the large.  $\square$

### 3 Constraint Nets: A General Model of Dynamics

In this section, we first introduce Constraint Nets, a model for dynamic systems, then examine the relationship between constraint nets and constraint satisfaction.

#### 3.1 Constraint Nets

A constraint net is composed of transductions and locations. A *transduction* is any mapping from a tuple of input traces to an output trace which is causal, viz. the output value at any time is determined by the input values prior to or at that time. Transductions are mathematical models of transformational processes.

There are two elementary types of transductions for dynamic systems: transliterations and delays. A *transliteration*  $f_{\mathcal{T}}$  is a pointwise extension of a function  $f$  over a time set  $\mathcal{T}$ . We use  $\delta(\text{init})$  and  $\int(\text{init})$  to denote a *unit delay* in a discrete system and an *integration* in a continuous system respectively with *init* as the initial value.

A *constraint net* is a triple  $CN \equiv \langle Lc, Td, Cn \rangle$ , where  $Lc$  is a set of *locations*,  $Td$  is a set of *transductions*, each of which is associated with a tuple of *input ports* and an *output port*,  $Cn$  is a set of directed *connections* between locations and ports of transductions, with the following restriction: (1) there is at most one connection pointing to each location, (2) each port of a transduction connects to a unique location and (3) no location is isolated.

A location is an *input* if there is no connection pointing to it otherwise it is an *output*. A constraint net is *closed* if it has no input location otherwise it is *open*. We use  $CN(I, O)$  to denote a *module* with a set of input locations  $I$  and a set of output locations  $O$  as the interface.

The graphical representation of a constraint net is a bipartite directed graph where locations are represented by circles, transductions by boxes and connections by arcs, each from a port of a transduction to a location or vice versa.

Semantically, each location  $l$  denotes a trace  $x$  and each transduction  $F_{\mathcal{T}}$  corresponds to an equation  $x = F_{\mathcal{T}}(x_1, \dots, x_n)$ . A constraint net corresponds to a set of equations; its semantics is the least fixpoint of the equation set [22].

In general, constraint nets can model hybrid dynamic systems, with components operating on different time structures or triggered by events. In this paper, we focus on only two types of constraint nets: discrete transition systems and continuous integration systems, corresponding respectively to two different types of constraint solvers.

#### 3.2 Constraint solvers

An output location is a *state* location if it is an output of a unit delay or an integration. A state of a constraint net is a mapping from the set of state locations to a set of values.

A *constraint solver* can be regarded as a special kind of constraint net which is closed and state-determined, i.e. a state trace is determined by its initial state. A constraint solver  $CS$  defines a process  $p: S \rightarrow \mathcal{V}_{\mathcal{T}}^S$  where  $S$  is a set of states and  $\mathcal{V}_{\mathcal{T}}^S$  is a set of state traces. A (stable) equilibrium of  $p$  is called a (stable) equilibrium of  $CS$ ; an attractor of  $p$  is called an attractor of  $CS$ .

$CS$  solves  $C$  iff (1) every solution of  $C$  is a stable equilibrium of  $CS$  and (2) the solution set of  $C$  is an attractor of  $CS$ ;  $CS$  solves  $C$  globally iff, in addition, the solution set of  $C$  is an attractor in the large.

**Lemma 1** *If  $CS$  solves  $C$  globally then every equilibrium of  $CS$  is a solution of  $C$ .*

Proof: Trivial.  $\square$

We discuss here two basic types of constraint solvers: state transition systems for discrete cases and state integration systems for continuous cases. A *state transition system* is a pair  $\langle S, f \rangle$  where  $S$  is the set of states and  $f : S \rightarrow S$  is the *state transition function*. A state transition system can be represented by a constraint net with a transliteration  $f_T$  and a unit delay  $\delta(s_0)$  where  $s_0 \in S$  is an initial state (Fig. 1). The solution of this net is an infinite sequence  $p(s_0) \equiv s_0, f(s_0) \dots f^n(s_0) \dots$

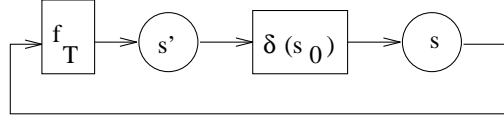


Figure 1: A constraint net representing  $\langle S, f \rangle$

Clearly, a state  $s^* \in S$  is an equilibrium of a state transition system  $\langle S, f \rangle$  iff  $s^* = f(s^*)$ .

**Lemma 2** *Let  $\langle S, d \rangle$  be a metric space. An equilibrium  $s^*$  in a state transition system is stable if  $\exists \Omega = N^\epsilon(s^*), \forall s \in \Omega, d(f(s), f(s^*)) \leq d(s, s^*)$ . Moreover,  $s^*$  is asymptotically stable if  $\exists \Omega, \forall s \in \Omega, d(f(s), f(s^*)) \leq kd(s, s^*)$  for  $0 \leq k < 1$ . If  $\Omega = S$ ,  $s^*$  is an asymptotically stable equilibrium in the large.*

Proof: Let  $V(s) = d(s, s^*)$ . It is easy to see that  $V$  is a Liapunov function for  $s^*$  and  $\langle S, f \rangle$ .  $\square$

Integration is a basic transduction on continuous time structures. A *state integration system* is a differential equation  $\frac{ds}{dt} = f(s)$ <sup>1</sup> which can be represented by a constraint net with a transliteration  $f_T$  and an integration  $\int(s_0)$  where  $s_0 \in S$  is an initial state (Fig. 2).

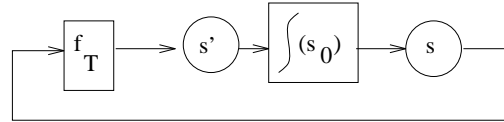


Figure 2: A constraint net representing  $\frac{ds}{dt} = f(s)$

Clearly, a state  $s^* \in S$  is an equilibrium of a state integration system  $\frac{ds}{dt} = f(s)$  iff  $f(s^*) = 0$ .

**Lemma 3** *An equilibrium  $s^*$  in a state integration system is stable if  $f$  is continuous at  $s^*$  and  $s^*$  is a local minimum of  $-\int f(s)ds$ . Moreover,  $s^*$  is asymptotically stable if it is the unique minimum in its neighborhood. If there is no other equilibrium and  $s^*$  is the global minimal point,  $s^*$  is an asymptotically stable equilibrium in the large.*

Proof: Let  $V(s) = -\int f(s)ds$ . It is easy to check that (1)  $\frac{dV}{dt} \leq 0$  and (2) since  $s^*$  is a local minimum of  $V$ , there is a neighborhood of  $s^*$  such that  $V(s) \leq V(s')$  iff  $d(s, s^*) \leq d(s', s^*)$ . Therefore  $V$  is a Liapunov function for  $s^*$  and  $\frac{ds}{dt} = f(s)$ .  $\square$

## 4 Properties of Constraint Methods

In this section, we examine various constraint methods and their properties. In particular, we discuss two types of constraint satisfaction problems, namely, global consistency and unconstrained optimization, for four classes of relations: relations on finite domains, linear, convex and nonlinear relations in  $n$ -dimensional Euclidean space  $\langle \mathcal{R}^n, d_n \rangle$ , where  $d_n(x, y) = |x - y| = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$ .

*Global consistency* corresponds to solving hard constraints and *unconstrained optimization* corresponds to solving soft constraints. A problem of the first kind can be translated into one of the second by introducing an energy function representing the degree of global consistency.

<sup>1</sup> If  $s$  is a tuple,  $\frac{ds}{dt} = f(s)$  represents a set of equations.

## 4.1 Unconstrained optimization

The problem of *unconstrained optimization* is to minimize an energy function  $\mathcal{E} : \mathcal{R}^n \rightarrow \mathcal{R}$ . Here we first discuss two methods for this problem: the gradient method (GM) [16] and Newton's method (NM) [19], and then study the schema model (SM) for minimizing an energy function  $\mathcal{E} : [0, 1]^n \rightarrow \mathcal{R}$ .

### 4.1.1 Gradient method

The gradient method is based on the *gradient descent* algorithm, where state variables slide downhill in the opposite direction of the gradient. Formally, if the function to be minimized is  $\mathcal{E}(x)$  where  $x = \langle x_1, \dots, x_n \rangle$ , then at any point, the vector that points towards the direction of maximum increase of  $\mathcal{E}$  is the gradient of  $\mathcal{E}$ . Therefore, the following gradient descent equations model the gradient method:

$$\frac{dx_i}{dt} = -k_i \frac{\partial \mathcal{E}}{\partial x_i}, \quad k_i > 0. \quad (1)$$

Let  $X^* = \{x^* | x^* \text{ is a true local minimum of } \mathcal{E}\}^2$  and  $Y^* = \{x^* | x^* \text{ is a global minimum of } \mathcal{E}\}$ . Let GM be a constraint net representing the gradient descent equations (1). The following theorem specifies the conditions under which GM solves the problems  $X^*$  and  $Y^*$ .

**Theorem 3** *GM solves  $X^*$  if  $\frac{\partial \mathcal{E}}{\partial x}$  is continuous at every  $x^* \in X^*$ . GM solves  $Y^*$  if, in addition,  $\mathcal{E}$  is bounded from below. GM solves  $Y^*$  globally if, in addition,  $\mathcal{E}$  is convex<sup>3</sup>.*

Proof: According to Lemma 3 every solution is a stable equilibrium. According to Theorem 2, by letting  $\mathcal{E}$  be a Liapunov function, we can prove that  $X^*$  is an attractor. If  $\mathcal{E}$  is convex,  $Y^*$  contains all of the equilibria, therefore, the set of global minima is the attractor in the large.  $\square$

### 4.1.2 Newton's method

Newton's method is to minimize a second-order approximation of the given energy function, at each iterative step. Let  $\Delta \mathcal{E} = \frac{\partial \mathcal{E}}{\partial x}$  and  $J$  be the Jacobian of  $\Delta \mathcal{E}$ . At each step with current point  $x^{(k)}$ , Newton's method is to minimize the function:

$$\mathcal{E}_a(x) = \mathcal{E}(x^{(k)}) + \Delta \mathcal{E}^T(x^{(k)})(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T J(x^{(k)})(x - x^{(k)}).$$

Let  $\frac{\partial \mathcal{E}_a}{\partial x} = 0$ , we have:

$$\Delta \mathcal{E}(x^{(k)}) + J(x^{(k)})(x - x^{(k)}) = 0.$$

The solution of the above equation becomes the next point, i.e.

$$x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})\Delta \mathcal{E}.$$

Newton's method defines a state transition system  $\langle \mathcal{R}^n, f \rangle$  where  $f(x) = x - J^{-1}(x)\Delta \mathcal{E}(x)$ .

Let NM be the constraint net representing Newton's method. The following theorem specifies the conditions under which NM solves the problem  $X^*$  and  $Y^*$ .

**Theorem 4** *NM solves  $X^*$  if  $|J(x^*)| \neq 0$  at every  $x^* \in X^*$ , i.e.  $\mathcal{E}$  is strictly convex at  $x^*$ . NM solves  $Y^*$  if, in addition,  $\mathcal{E}$  is bounded from below. NM solves  $Y^*$  globally if, in addition,  $\mathcal{E}$  is convex.*

Proof: First, we prove that  $x^* = f(x^*)$  and  $|J(x^*)| \neq 0$  implies  $x^*$  is asymptotically stable. Let  $R$  be the Jacobian of  $f$ . It is easy to check that  $|R(x^*)| = 0$ . There exists a neighborhood of  $x^*$ ,  $N^\epsilon(x^*)$ , for any  $x \in N^\epsilon(x^*)$ ,  $|f(x) - f(x^*)| \leq k|x - x^*|$  for  $0 < k < 1$ . According to Lemma 2,  $x^*$  is asymptotically stable. If  $\mathcal{E}$  is convex, there is no other equilibrium, so that  $x^*$  is asymptotically stable in the large.  $\square$

Here we assume that the Jacobian and its inverse are obtained off-line. Newton's method can also be used to solve a set of nonlinear equations  $g(x) = 0$  by replacing  $\Delta \mathcal{E}$  with  $g$ .

<sup>2</sup>This excludes flat maximum and saddle surfaces.

<sup>3</sup>A function  $f$  is convex iff for any  $\lambda \in (0, 1)$ ,  $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$ ; it is strictly convex iff the inequality is strict. Obviously, a strictly convex function has a unique minimal point. Linear functions are convex, but not strictly convex. A quadratic function  $x^T M x + c^T x$  is convex if  $M$  is semi-positive definite; it is strictly convex if  $M$  is positive definite.

### 4.1.3 Schema model

The schema model has been used for finite constraint satisfaction in the PDP framework [18]. Basically, there is a set of units, each can be on or off; constraints between units are represented by weights on connections. The energy is typically a quadratic function in the following form:

$$\mathcal{E}(a) = -(\sum_{i,j} w_{ij} a_i a_j + \sum_i b_i a_i) = -(a^T W a + b^T a)$$

where  $a_i \in [0, 1]$  indicates the activation value and  $b_i$  specifies the bias for unit  $i$ ,  $w_{ij}$  represents the constraint between two units  $i$  and  $j$ .  $w_{ij}$  is positive if units  $i$  and  $j$  support each other, it is negative if the units are against each other and it is zero if the units have no effect on each other.

There are various methods for solving this problem. The schema model [18] provides the simplest discrete relaxation method. Let  $n_i(a) = \frac{\partial \mathcal{E}}{\partial a_i} = -\sum_j w_{ij} a_j - b_i$ . The schema model defines a state transition system  $\langle [0, 1]^n, f \rangle$  where  $f = \langle f_1, \dots, f_n \rangle$  with  $f_i(a) = a_i - n_i(a) a_i$  if  $n_i(a) > 0$  and  $f_i(a) = a_i - n_i(a)(1 - a_i)$  otherwise. In other words,  $f_i(a) = (1 - |n_i(a)|) a_i - \min(0, n_i(a))$ .

**Theorem 5** *Let SM be a constraint net representing the schema model with  $|n_i(a)| \leq 1$  for any  $i$  and  $a$ . SM solves the set of minima of  $\mathcal{E}$ , denoted  $A^*$ .*

Proof: Let  $a^{(k+1)}$  denote  $f(a^{(k)})$ . First, because  $|n_i(a)| \leq 1$ ,  $a^{(k)} \in [0, 1]^n$  implies  $a^{(k+1)} \in [0, 1]^n$ . Therefore  $f$  is well defined. Second, for each minimum  $a^*$  of  $\mathcal{E}$ , and for any  $i$ , either (1)  $n_i(a^*) = 0$  or (2)  $n_i(a^*) > 0$  and  $a_i^* = 0$  or (3)  $n_i(a^*) < 0$  and  $a_i^* = 1$ . Therefore  $a^*$  is an equilibrium. Now we prove that  $a^*$  is stable. Let  $N^\epsilon(a^*)$  be a neighborhood such that  $\forall a \in N^\epsilon(a^*)$  and for any  $i$  if  $n_i(a^*) \neq 0$ , then  $n_i(a)$  and  $n_i(a^*)$  have the same sign otherwise if  $n_i(a) \geq 0$ , then  $a_i \geq a_i^*$  otherwise  $a_i \leq a_i^*$ . Such a neighborhood exists because  $n_i$  is continuous. Considering  $|f_i(a) - a_i^*|$ , there are four cases.

1.  $n_i(a^*) > 0$ : In this case,  $a_i^* = 0$  and  $|f_i(a) - a_i^*| = |f_i(a)| = |1 - n_i(a)| \times |a_i| \leq |a_i - a_i^*|$ .
2.  $n_i(a^*) < 0$ : In this case,  $a_i^* = 1$  and  $|f_i(a) - a_i^*| = |f_i(a) - 1| = |1 + n_i(a)| \times |a_i - 1| \leq |a_i - a_i^*|$ .
3.  $n_i(a^*) = 0$  and  $n_i(a) \geq 0$ :  $|f_i(a) - a_i^*| = |(1 - n_i(a)) a_i - a_i^*| = |a_i - a_i^* - n_i(a) a_i| \leq |a_i - a_i^*|$ .
4.  $n_i(a^*) = 0$  and  $n_i(a) \leq 0$ :  $|f_i(a) - a_i^*| = |(1 + n_i(a)) a_i - a_i^*| = |a_i - a_i^* - n_i(a)(1 - a_i)| \leq |a_i - a_i^*|$ .

Therefore  $\forall a \in N^\epsilon(a^*)$ ,  $|f_i(a) - a_i^*| \leq |a_i - a_i^*|$  and  $|f(a) - a^*| \leq |a - a^*|$ . According to Lemma 2,  $a^*$  is stable. Furthermore, let  $V(a) = |a - A^*|$  be defined on a neighborhood of  $A^*$ ,  $V$  is a Liapunov function for  $A^*$  and SM. According to Theorem 2,  $A^*$  is an attractor.  $\square$

## 4.2 Global consistency

Unconstrained optimization methods can also be used to solve a set of equations  $g_i(x) = 0, i = 1..n$ , by letting  $\mathcal{E}_g(x) = \sum_{i=1..n} w_i g_i^2(x)$  where  $w_i > 0$  and  $\sum_i w_i = 1$ . If a constraint solver  $CS$  solves  $\min \mathcal{E}_g(x)$ ,  $CS$  solves  $g(x) = 0$ . Inequality constraints can be transformed into equality constraints. There are two approaches. Let  $g_i(x) \leq 0$  be an inequality constraint, the equivalent equality constraint is (i)  $\max(0, g_i(x)) = 0$  or (ii)  $g_i(x) + z^2 = 0$  where  $z$  is introduced as an extra variable. Similarly, the schema model can be used to solve a set of constraints with finite domains, by assigning each possible value a unit and each constraint between two values a weight.

However, in many cases it is more efficient to solve a set of (in)equality constraints directly. Moreover, a method for solving a set of equality constraints can also be used to solve an unconstrained optimization problem, since  $x^*$  is a local extremum of  $\mathcal{E}$  implies  $\frac{\partial \mathcal{E}}{\partial x}(x^*) = 0$ . Similarly, the problem of finite domain constraint satisfaction can be solved directly on constraint nets.

Here we first discuss the projection method (PM) for solving (in)equality constraints, and then study the method for solving global consistency of finite domain constraints (FM).

### 4.2.1 Projection method

A *projection* of a point  $x$  to a set  $R$  in a metric space  $\langle X, d \rangle$  is a point  $P_R(x) \in R$ , such that  $d(x, P_R(x)) = d(x, R)$ . Projections in the  $n$ -dimensional Euclidean space  $\langle \mathcal{R}^n, d_n \rangle$  share the following properties.

**Lemma 4** [8] *Let  $R \subset \mathcal{R}^n$  be closed and convex<sup>4</sup>. The projection  $P_R(x)$  of  $x$  to  $R$  exists and is unique for every  $x$ , and  $(x - P_R(x))^T(y - P_R(x)) \leq 0$  for any  $y \in R$ .*

Suppose we are given a system of convex and closed sets,  $X_i$  for  $i = 1..m$ . The problem is to find  $x^* \in \cap_i X_i$ . Let  $P(x) = P_{X_i}(x)$  be a projection of  $x$  to a least satisfied set  $X_i$ , i.e.  $d(x, X_i) = \max_i d(x, X_i)$ . The projection method [8] for this problem defines a state transition system  $\langle \mathcal{R}^n, f \rangle$  where  $f(x) = x + \lambda(P(x) - x)$  for  $0 < \lambda < 2$ .

The following theorem is derived from a similar one in [8], however, the proof given here is simplified by the use of Liapunov functions.

**Theorem 6** *Let PM be a constraint net representing the projection method. PM solves  $X^* = \cap_i X_i$  globally if all the  $X_i$ 's are convex.*

Proof: First of all, it is easy to see that if  $x^*$  is a solution, then  $x^* = f(x^*)$ , i.e.  $x^*$  is an equilibrium. Moreover, we can prove that  $|f(x) - x^*| \leq |x - x^*|$  for any  $x$  as follows.

$$\begin{aligned} |f(x) - x^*|^2 &= |x + \lambda(P(x) - x) - x^*|^2 \\ &= |x - x^*|^2 + \lambda^2|P(x) - x|^2 + 2\lambda(x - x^*)^T(P(x) - x) \\ &= |x - x^*|^2 + (\lambda^2 - 2\lambda)|P(x) - x|^2 + 2\lambda(P(x) - x)^T(P(x) - x^*) \\ &\leq |x - x^*|^2 - \lambda(2 - \lambda)|P(x) - x|^2 \quad \text{according to Lemma 4} \\ &\leq |x - x^*|^2 \quad \text{since } 0 < \lambda < 2. \end{aligned}$$

According to Lemma 2,  $x^*$  is stable.

Then, we can prove that  $X^*$  is an attractor in the large. Let  $V(x) = |x - X^*|$  on  $\mathcal{R}^n$ .  $V(x)$  is a Liapunov function on  $\mathcal{R}^n$  since  $V(f(x)) \leq V(x)$  for any  $x$ . Moreover  $V(f(x)) < V(x)$  for any  $x \notin X^*$  since for any  $x^* \in X^*$  and  $x \notin X^*$ ,  $|f(x) - x^*| < |x - x^*|$ . In addition, we can prove that the process defined by PM satisfies the condition in Theorem 2, since  $\lim_{n \rightarrow \infty} x^n = x^*$  implies  $\lim_{n \rightarrow \infty} |x^{n+1} - x^n| = 0$ . Therefore  $\lim_{n \rightarrow \infty} |P(x^n) - x^n| = 0$  and  $P(x^*) = x^*$ , so that  $x^*$  is an equilibrium. According to Theorem 2,  $X^*$  is an attractor in the large.  $\square$

The projection method can be used to solve a set of inequality constraints, i.e.  $X_i = \{x | g_i(x) \leq 0\}$  for convex function  $g_i$ . Linear functions are convex. Therefore the projection method can be applied to a set of linear inequalities  $Ax \leq b$ , where  $x = \langle x_1, \dots, x_n \rangle \in \mathcal{R}^n$ . Let  $A_i$  be the  $i$ th row of  $A$ . The projection of a point  $x$  to a half space  $A_i x - b_i \leq 0$  is defined as:

$$P_i(x) = \begin{cases} x & \text{if } A_i x - b_i \leq 0 \\ x - cA_i^T & \text{otherwise} \end{cases}$$

where  $c = (A_i x - b_i) / |A_i^T|^2$ . This reduces to the method described in [1]. Without any modification, this method can be also applied to a set of linear equalities, by simply replacing each linear equality  $g_i(x) = 0$  with two linear inequalities:  $g_i(x) \leq 0$  and  $-g_i(x) \leq 0$ .

There are various ways to modify this method for faster convergence. For instance, [3] gives a simultaneous projection method in which  $f(x) = x + \lambda \sum_{i \in I} w_i (P_i(x) - x)$  where  $I$  is an index set of violated constraints,  $w_i > 0$  and  $\sum_{i \in I} w_i = 1$ . [21] gives a method in which  $f(x) = x + \lambda(P_S(x) - x)$  where  $S = \{x | \sum_{i \in I} w_i g_i(x) \leq 0\}$ . Furthermore, for a large set of inequalities, the problem can be decomposed into a set of  $K$  subproblems with  $f_k$  corresponding to the transition function of the  $k$ th subproblem. The whole problem can be solved by combining the results of  $\{f_1, \dots, f_K\}$ .

<sup>4</sup>A set  $R$  in  $n$ -dimensional Euclidean space is convex iff for any  $\lambda \in (0, 1)$ ,  $x, y \in R$  implies  $\lambda x + (1 - \lambda)y \in R$ . Clearly if  $g$  is a convex function,  $\{x | g(x) \leq 0\}$  is a convex set.

### 4.2.2 Finite constraint satisfaction

Many problems can be formalized as finite constraint satisfaction problems (FCSPs), which can be represented by constraint networks [24]. Formally, a *constraint network*  $C$  is a quadruple  $\langle V, dom, A, con \rangle$  where

- $V$  is a set of variables,  $\{v_1, v_2, \dots, v_N\}$ ,
- associated with each variable  $v_i$  is a finite domain  $d_i = dom(v_i)$ ,
- $A$  is a set of arcs,  $\{a_1, a_2, \dots, a_n\}$ ,
- associated with each arc  $a_i$  is a constraint  $con(a_i) = r_i(R_i)$  where  $R_i \subseteq V$  is a relation scheme and  $r_i$  is a set of relation tuples on  $R_i$ .

The *solution set* for the constraint network  $C$  is the join of all the relations,  $sol(C) = r_1 \bowtie \dots \bowtie r_n$ .

An FCSP can be solved using the schema model (SM) by assigning each possible value in the finite domain of a variable a unit. The units of two values from the same variable are against each other; the units of two values from different variables support each other if they are consistent. However, SM does not solve an FCSP globally.

An FCSP can be solved directly using various methods [4, 6, 11, 12, 14]. Let  $Scheme(C) = \{R_1, \dots, R_n\}$  be the scheme of a constraint network  $C$ . The *solution of a constraint network*  $C$  is a network  $C'$ , with  $sol(C) = sol(C')$ ,  $Scheme(C) = Scheme(C')$ , and  $r'_i = \Pi_{R_i}(sol(C'))$  where  $\Pi_{R_i}$  is a projection operator. Such a solution network is called a *minimal* network [14]. Here we present a relaxation method (FM) which finds the solution network of a constraint network with an acyclic scheme. This kind of method has been studied by many researchers, for instance, [7, 17, 24]. We examine the property of the method within the framework of dynamic systems.

Let  $\mathcal{C}$  be the set of constraint networks with the same scheme and solution set. We define a state transition system  $\langle \mathcal{C}, f \rangle$  where  $f = \{f_i\}_{a_i \in A}$  with  $f_i(r_i) = \bigcap_{\{j | R_i \cap R_j \neq \emptyset\}} \Pi_{R_i}(r_i \bowtie r_j)$ .

**Theorem 7** *Let FM be a constraint net representing a state transition system  $\langle \mathcal{C}, f \rangle$ . FM finds the solution network globally in  $\mathcal{C}$  if the scheme of  $\mathcal{C}$  is acyclic.*

Proof: First of all, it is clear that a solution network  $C^*$  is an equilibrium of the state transition system. Now let us define a metric on the set  $\mathcal{C}$ . Given a relation scheme  $R$ , the distance between two relation tuples can be defined as  $d_R(r_1, r_2) = |(r_1 - r_2) \cup (r_2 - r_1)|$  where  $|r|$  denotes the number of relation tuples. The distance between two constraint networks in  $\mathcal{C}$  can be defined as  $d(C_1, C_2) = \sqrt{\sum_{Scheme(C)} d_R^2(r_1, r_2)}$ . Let us define a function  $L$  on  $\mathcal{C}$  as:  $L(C) = \sqrt{\sum_{Scheme(C)} |r|^2}$ .  $L$  is a Liapunov function for the solution network  $C^*$  and  $\langle \mathcal{C}, f \rangle$  since (1)  $L(C_1) \leq L(C_2)$  iff  $d(C_1, C^*) \leq d(C_2, C^*)$  and (2)  $L(f(C)) \leq L(C)$  for any  $C \in \mathcal{C}$ . Therefore  $C^*$  is a stable equilibrium. Finally, we can prove that if the scheme of  $\mathcal{C}$  is acyclic,  $C^*$  is an asymptotically stable equilibrium. For an acyclic network, an equilibrium implies a minimal network [24] and clearly if  $C \neq C^*$ ,  $L(f(C)) < L(C)$ . According to Theorem 2,  $C^*$  is asymptotically stable.  $\square$

## 5 Embedded Constraint Solvers and Implementation Issues

In this section, we consider two variations of constraint solvers. The first corresponds to open constraint nets, for designing embedded control systems. The second corresponds to constraint nets with latency.

### 5.1 Embedded constraint solvers

One of the important applications of constraint solvers is the design of robot control systems [15]. There are two kinds of embedded constraint solvers for this application. First, a constraint solver is coupled to a dynamic environment. Second, a constraint solver is coupled to the plant of a robot. In both cases, the embedded constraint solver is part of the robot controller. The combination of these two embeddings will occur in real applications.

An embedded constraint solver coupled to an environment (resp. a plant) is an open constraint net  $CN(I, O)$ , where the set of input locations  $I$  act as sensors of the environment (resp. the plant). Constraints

are relations on input and output values. A constraint net is an *embedded constraint solver* for the set of constraints  $C$  and the environment (resp. the plant) iff the composition of the constraint net and the environment (resp. the plant) solves  $C$ .

Consider the case of designing a tracking system  $S$  which chases a target  $T$ . Let  $x$  be the position of  $S$  and  $x_d$  be the position of  $T$ , the constraint to be satisfied is  $|x - x_d| = 0$ . Suppose we design a tracking system with the following law:  $\frac{dx}{dt} = -k(x - x_d)$  where  $x_d$  is an input trace<sup>5</sup>. However, this system is not an embedded constraint solver for  $|x - x_d| = 0$  if  $|\frac{dx_d}{dt}| > 0$ . A correct design is  $\frac{dx}{dt} = \frac{dx_d}{dt} - k(x - x_d)$ . To see why this is the right design, we define a Liapunov function  $V(x, x_d) = \frac{1}{2}|x - x_d|^2$  and observe that  $\frac{dV}{dt} = (x - x_d)(\frac{dx}{dt} - \frac{dx_d}{dt}) \leq 0$ . In reality, both  $x_d$  and  $\frac{dx_d}{dt}$  can be inaccurate. However, the system is robust with respect to the inaccuracy.

## 5.2 Implementation issues

Constraint solvers (or embedded constraint solvers) can be implemented as analog or digital circuits, or as programs in multiprocessor environments. For a discrete constraint solver, the efficiency can be characterized by the convergence rate of the method and the computation cost of the transition function. Constraint nets are inherently parallel, while sequential computation can be considered as a special case. Clearly, for discrete constraint solvers, except those embedded in dynamic environments, the computation time will not affect the dynamic behaviors.

However, for a continuous constraint solver, latencies in the circuit may change the dynamic behavior of the constraint solver totally. Consider a simple example:  $\frac{dx}{dt} = -kx$  with  $k > 0$  solves  $x = 0$  globally. However, if there is a latency  $\delta$  in the wires, the actual equation becomes  $\frac{dx}{dt} = -kx(t - \delta)$ . In this case,  $x = 0$  is still an equilibrium, but it may not be an asymptotically stable equilibrium. In fact, if  $\delta k > 2$ , it is unstable at  $x = 0$ . Therefore, it is important at the design stage to model the possible latencies and to choose the right value for  $k$ .

## 6 Conclusion

We have presented a unitary model of constraint satisfaction as a dynamic process. Various constraint methods and their dynamic properties have been studied, and their applications to control system design are examined. The Constraint Net model serves as a useful abstract target machine for constraint programming languages, providing both semantics and pragmatics.

**Acknowledgements:** We wish to thank Uri Ascher, Peter Lawrence, Dinesh Pai, Nick Pippenger and Runping Qi for valuable discussions and suggestions. This research was supported by the Natural Sciences and Engineering Research Council and the Institute for Robotics and Intelligent Systems.

## References

- [1] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:382–392, 1954.
- [2] A. Aiba, K. Sakai, Y. Sato, and D.J. Hawley. Constraint logic programming language cal. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 263–276, 1988.
- [3] Y. Censor and T. Elfving. New method for linear inequalities. *Linear Algebra and Its Applications*, 42:199–211, 1982.
- [4] R. Dechter. Constraint networks. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 285–293. Wiley, N.Y., 1992.

---

<sup>5</sup>In practice,  $x - x_d$  would be sensed.

- [5] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The constraint logic programming language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 693 – 702, 1988.
- [6] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceeding of AAAI-90*, 1990.
- [7] E. C. Freuder. Completable representations of constraint satisfaction problems. In *KR-91*, pages 186 – 195, 1991.
- [8] L.G. Gubin, B.T. Polyak, and E.V. Raik. The method of projections for finding the common point of convex sets. *U.S.S.R. Computational Mathematics and Mathematical Physics*, pages 1–24, 1967.
- [9] J. Jaffar and J.L. Lassez. Constraint logic programming. In *ACM Principles of Programming Languages*, pages 111 – 119, 1987.
- [10] D.G. Luenberger. *Introduction to Dynamic Systems: Theory, Models and Applications*. John Wiley & Sons, 1979.
- [11] A. K. Mackworth. Constraint satisfaction. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 276 – 285. Wiley, N.Y., 1992.
- [12] A.K. Mackworth. The logic of constraint satisfaction. *Artificial Intelligence*, 58:3–20, 1992.
- [13] M. D. Mesarovic and Y. Takahara. *General Systems Theory: Mathematical Foundations*. Academic Press, 1975.
- [14] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7:95–132, 1974.
- [15] D. K. Pai. Least constraint: A framework for the control of complex mechanical systems. In *Proceedings of American Control Conference*, pages 426 – 432, Boston, 1991.
- [16] J. Platt. Constraint methods for neural networks and computer graphics. Technical Report Caltech-CS-TR-89-07, Department of Computer Science, California Institute of Technology, 1989.
- [17] F. Rossi and U. Montanari. Exact solution in linear time of networks of constraints using perfect relaxation. In *Proceedings First Int. Principles of Knowledge Representation and Reasoning, Toronto, Ontario, Canada*, pages 394–399, May 1989.
- [18] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing — Exploration in the Microstructure of Cognition*. MIT Press, 1986.
- [19] J. T. Sandfur. *Discrete Dynamical Systems: Theory and Applications*. Clarendon Press, 1990.
- [20] V. A. Saraswat, M. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. Technical Report SSL-90-86, Palo Alto Research Center, 1990.
- [21] K. Yang and K.G. Murty. New iterative methods for linear inequalities. Unpublished.
- [22] Y. Zhang and A. K. Mackworth. Constraint nets: A semantic model of real-time embedded systems. Technical Report 92-10, Department of Computer Science, University of British Columbia, 1992.
- [23] Y. Zhang and A. K. Mackworth. Will the robot do the right thing? Technical Report 92-31, Department of Computer Science, University of British Columbia, 1992.
- [24] Y. Zhang and A. K. Mackworth. Parallel and distributed constraint satisfaction: Complexity, algorithms and experiments. In Laveen N. Kanal, editor, *Parallel Processing for Artificial Intelligence*. Elsevier/North Holland, 1993. to appear.