1 Constraint Programming in Constraint Nets

Ying Zhang and Alan K. Mackworth

1.1 Abstract

We view constraints as relations and constraint satisfaction as a dynamic process of approaching the solution set of the constraints. We have developed a semantic model for dynamic systems, 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. In particular, we examine discrete and continuous methods for discrete and continuous domain constraint satisfaction problems. Hard and soft constraints within the framework of unconstrained and constrained optimization are considered. Finally, we present an application of this on-line constraint satisfaction framework to the design of robot control systems.

1.2 Motivation

Constraints are relations among entities. Constraint satisfaction can be viewed in two different ways. In a 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 a 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, though for any given relation tuple the system is able to check whether or not the relation tuple is entailed.

In an alternative view, which in our opinion is more appropriate for real-time embedded systems, the constraint satisfaction problem is formulated as finding a relation tuple that is entailed by a given set of constraints [12]. In this paper, we present a new approach in which constraint satisfaction is a dynamic process with the solution set as an attractor of the process. "Monotonicity" is characterized by a Liapunov

function, measuring the "distance" to the set of solutions over time. Moreover, both soft and 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 a semantic model for dynamic systems, called Constraint Nets, to provide a real-time programming semantics [24] and to model and analyze robotic systems [25]. Here we investigate the relationship between constraint satisfaction and constraint nets. The rest of this paper is organized as follows. Section 1.3 describes some basic concepts of dynamic processes. Section 1.4 introduces Constraint Nets and constraint solvers. Section 1.5 presents various constraint methods for solving global consistency and optimization problems. Section 1.6 summarizes the framework. Section 1.7 discusses an application to control synthesis for robotic systems.

1.3 Properties of Dynamic Processes

In this section, we define dynamic processes and discuss the relationship between dynamic processes and Liapunov functions.

1.3.1 Metric spaces

Let $\mathcal{R}$ be the set of real numbers and $\mathcal{R}^+$ be the set of nonnegative real numbers. A *metric* on a set X is a function $d : X \times X \rightarrow \mathcal{R}^+$ satisfying 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 $\langle X, d \rangle$, $d(x, y)$ is called "the distance between x and y." We will use X to denote the metric space $\langle X, d \rangle$ if no ambiguity arises.

Given a metric space $\langle X, d \rangle$, 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 $x^* \in X$ and $\epsilon > 0$, the set $N^\epsilon(x^*) = \{x | d(x, x^*) < \epsilon\}$ is an *ϵ-neighborhood* of x^*; it is *strict* if it has at least one point other than x^*. For $X^* \subset X$, $N^\epsilon(X^*) = \bigcup_{x^* \in X^*} N^\epsilon(x^*)$ is an *ϵ-neighborhood* of X^*; it is *strict* if it is

a strict superset of X^*.

Let $\mathcal{T}$ be a set of linearly ordered *time* points with a least element $\mathbf{0}$. $\mathcal{T}$ can be either discrete or continuous. Let X be a metric space representing a discrete or continuous *domain*. A *trace* $v : \mathcal{T} \rightarrow X$ is a function from time to a domain. We use $X^{\mathcal{T}}$ to denote the set of all traces from $\mathcal{T}$ to X, a *trace space*. Given a metric space $\langle X, d\rangle$ and a trace $v : \mathcal{T} \rightarrow X$, we say *$v$ approaches a point $x^* \in X$* iff $\lim_{t\rightarrow\infty} d(v(t), x^*) = 0$; *$v$ approaches a set $X^* \subset X$* iff $\lim_{t\rightarrow\infty} d(v(t), X^*) = 0$.

1.3.2 Dynamic processes

Let $p : X \rightarrow X^{\mathcal{T}}$ be a mapping from a domain X to a trace space $X^{\mathcal{T}}$ and $\phi_p(x) = \{p(x)(t) | t \in \mathcal{T}\}$ be the set of values in trace $p(x)$, p is a *dynamic process* iff p satisfies the following three conditions:

1. $\forall x, p(x)(\mathbf{0}) = x$.
2. $\forall x, y, t$, if $p(x)(t) = p(y)(t)$ then $\forall t' \geq t$, $p(x)(t') = p(y)(t')$.
3. $\forall x, y$, if $y \in \phi_p(x)$ then $\phi_p(y) \subseteq \phi_p(x)$.

Intuitively, p characterizes a state-based and time-invariant dynamic system. Furthermore, let $\phi_p(X^*) = \bigcup_{x\in X^*} \phi_p(x)$ for any $X^* \subset X$.

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

A set $X^* \subset X$ is an *attractor* [19] of a dynamic process p iff there exists a strict ϵ-neighborhood $N^\epsilon(X^*)$ such that $\forall x \in N^\epsilon(X^*)$, $p(x)$ approaches X^*. The largest neighborhood of X^* satisfying this property is called the *attraction basin* of X^*. X^* is an *attractor in the large* iff $\forall x \in X$, $p(x)$ approaches X^*, that is the attraction basin of X^* is X. If X^* is an attractor (in the large) and X^* is a stable equilibrium, X^* is called an *asymptotically* stable equilibrium (in the large).

LEMMA 1 If $\{X_i\}_{i\in I}$ are ((asymptotically) stable) equilibria, then $\bigcup_I X_i$ is an ((asymptotically) stable) equilibrium.

1.3.3 Liapunov functions

Let $\langle X, d\rangle$ be a metric space, $p : X \rightarrow X^{\mathcal{T}}$ be a dynamic process and $X^* \subset X$. A *Liapunov function* for p and X^* is a function $V : \Omega \rightarrow \mathcal{R}$, where Ω is a strict neighborhood of X^*, satisfying:

1. V is continuous, i.e., $d(x, x') \rightarrow 0$ implies $|V(x) - V(x')| \rightarrow 0$.

2. V has its unique minimum within Ω at X^*.
3. $\forall x \in \Omega, \forall t, V(p(x)(t)) \leq V(x)$.

The following two theorems are similar to those in [10].

THEOREM 1 $X^* \subset X$ is a stable equilibrium of a dynamic process p iff there exists a Liapunov function V for p and X^*.

Proof: The if part: Let V be a Liapunov function for p and X^*. First of all, X^* is an equilibrium since V takes the unique minimum at X^*. Suppose Ω is the domain of V. Given any ϵ, let $\epsilon' \leq \epsilon$ such that $N^{\epsilon'}(X^*) \subseteq \Omega$, and γ be the minimum over the boundary of $N^{\epsilon'}(X^*)$. $\gamma > V(X^*)$ since X^* is the unique minimum. Because V is continuous, there exists a δ-neighborhood $N^{\delta}(X^*)$ such that $\forall x \in N^{\delta}(X^*), V(x) < \gamma$. Therefore, $\phi_p(N^{\delta}(X^*)) \subseteq N^{\epsilon'}(X^*) \subseteq N^{\epsilon}(X^*)$.

The only if part: If X^* is a stable equilibrium of p, let $V(x) = \sup_{x' \in \phi_p(x)}\{d(x', X^*)\}$. We have (1) $V(X^*) = 0$ since X^* is an equilibrium, (2) $V(p(x)(t)) \leq V(x)$ since $\phi_p(p(x)(t)) \subseteq \phi_p(x)$, and (3) V is continuous since X^* is stable. Therefore, V is a Liapunov function for p and X^*. □

THEOREM 2 $X^* \subset X$ is an asymptotically stable equilibrium of a dynamic process p iff there exists a Liapunov function $V : \Omega \rightarrow \mathcal{R}$ for X^* and p, such that $\forall x \in \Omega, \lim_{t \rightarrow \infty} V(p(x)(t)) = V(X^*)$. Furthermore, if $\Omega = X$, X^* is an asymptotically stable equilibrium in the large.

Proof: Since X^* is the unique minimum in Ω, $p(x)$ approaches $X^*, \forall x \in \Omega$. Given V defined as the same as that in the previous proof, if X^* is an asymptotically stable equilibrium, $V(p(x)(t))$ approaches $V(X^*)$. □

1.4 Constraint Nets and Constraint Solvers

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

1.4.1 Constraint Nets

A dynamic system can be modeled by a constraint net. Intuitively, a constraint net consists of a finite set of locations, a finite set of transductions, each with a finite set of input ports and an output port, and a finite set of connections between locations and ports of transductions.

A *location* can be regarded as a wire, a channel, a variable, or a memory cell, whose value may change over time.

A *transduction* is a mapping from input traces to output traces which is causal, viz., the output value at any time is determined by the input values up to that time. Transductions are mathematical models of transformational processes. For example, a *transliteration* $f_{\mathcal{T}}$ is a transduction whose output value at any time $t \in \mathcal{T}$ is the function f of the input value at that time only. Intuitively, a transliteration is a transformational process without memory or internal state, for example, a combinational circuit. We use f to denote the transliteration $f_{\mathcal{T}}$ if no ambiguity arises. On the other hand, unit delays and temporal integrations are transductions modeling sequential processes. A *unit delay* $\delta(v_0)$ is a transduction defined mainly for discrete time structures, such that the output value at initial time $\mathbf{0}$ is v_0 and the rest of the output values are the input values at the previous time. A unit delay acts as a unit memory in discrete time dynamic systems. Corresponding to a unit delay, a *temporal integration* is a typical transduction in continuous time. We use $\int(v_0)$ to denote the temporal integration with initial output value v_0.

A *connection* links a location with a port of a transduction. The set of connections has the following restrictions: (1) there is at most one output port connected to each location, (2) each port of a transduction connects to a unique location and (3) no location is isolated.

A location l is an *input location* of a transduction F iff l connects to an input port of F; l is the *output location* of F iff l connects to the output port of F. A location is an *input* if it is not connected to the output location of any transduction; it is otherwise an *output*. A constraint net is *open* if there is an input; it is otherwise *closed*.

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

Semantically, a transduction F denotes an equation $l_0 = F(l_1, \cdots, l_n)$ where l_0 is the output location of F and $\langle l_1, \cdots, l_n \rangle$ is the tuple of input locations of F. A constraint net CN denotes a set of equations, $\vec{o} = \vec{F}(\vec{i}, \vec{o})$, each corresponds to a transduction in CN. The semantics of CN is a "solution" of the set of equations [24].

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

1.4.2 Constraint solvers

We view a *constraint* as a possibly implicit relation on a set of variables. The *constraint satisfaction problem* is defined as follows. Given a set of variables V with the associated domains $\{D_v\}_{v\in V}$ and a set of constraints $\{C_j\}_{j\in J}$ each on a subset of the variables, i.e., $C_j \subseteq \times_{V_j} D_v$ where $V_j \subseteq V$, find an explicit relation tuple $x \in \times_V D_v$ that satisfies all the given constraints, i.e., for all $j \in J$, $x_{|V_j} \in C_j$ where $x_{|S}$ denotes the restriction of x onto $S \subseteq V$. If $C = \{C_j\}_{j\in J}$ is a set of constraints, we use $sol(C)$ to denote the set of solutions, called *the solution set.*

A constraint solver for a constraint satisfaction problem is a closed constraint net whose semantics is a dynamic process approaching the solution set of the constraints. Formally, a closed constraint net CS^V is a *constraint solver* for a constraint satisfaction problem C on domain $X = \times_V D_v$ iff (1) the semantics of CS^V is a dynamic process $[\![CS^V]\!] : X \to X^{\mathcal{T}}$ and (2) $sol(C)$ is an asymptotically stable equilibrium of $[\![CS^V]\!]$. CS^V *solves* C *globally* iff $sol(C)$ an asymptotically stable equilibrium of $[\![CS^V]\!]$ in the large.

LEMMA 2 If a constraint solver CS^V solves a set of constraints C on variables V globally, every equilibrium of $[\![CS^V]\!]$ is a solution of C.

We discuss here two basic types of constraint solver: state transition systems for discrete methods and state integration systems for continuous methods.

A *state transition system* is a pair $\langle S, f\rangle$ where S is a set of states and $f : S \to S$ is a *state transition function.* A state transition system can be represented by a constraint net with a transliteration f and a unit delay $\delta(s_0)$ where $s_0 \in S$ is an initial state (Figure 1.1). The semantics of this net is a dynamic process $p : S \to S^{\mathcal{N}}$ with $p(s_0)$ as an infinite sequence $\langle s_0, f(s_0), \cdots, f^n(s_0), \cdots\rangle$. A state $s^* \in S$ is an equilibrium of $\langle S, f\rangle$ iff $s^* = f(s^*)$.

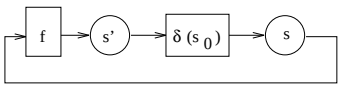

Figure 1.1
A constraint net representing $\langle S, f\rangle$

LEMMA 3 If $V : \Omega \rightarrow \mathcal{R}$ is a Liapunov function for $\langle S, f\rangle$ and $S^* = \{s^* | s^* = f(s^*)\} \subset \Omega$, then $V(f(x)) \leq V(x), \forall x \in \Omega$. In addition, if f is continuous and $V(f(x)) < V(x), \forall x \notin S^*$, S^* is an asymptotically stable equilibrium.

Proof: If $\lim_{n\rightarrow\infty} V(f^n(s)) = \epsilon > V(S^*)$, let $X = \{s | V(s) \leq \epsilon\} \supset S^*$, $f^n(s)$ approaches X. If f is continuous, however, $f^n(s)$ approaches $f(X) \subset X$ and $\lim_{n\rightarrow\infty} V(f^n(s)) < \epsilon$, contradiction. □

For continuous time structures and domains, integration is used to replace the unit delay. A *state integration system* is a differential equation $\dot{s} = f(s)$ that can be represented by a closed constraint net with a transliteration f and an integration $\int(s_0)$ where s_0 is an initial state (Figure 1.2). The semantics of this net is a dynamic process $p : S \rightarrow S^{\mathcal{R}^+}$ with $p(s_0)$ as the solution of $\dot{s} = f(s)$ and $s(0) = s_0$. A state $s^* \in S$ is

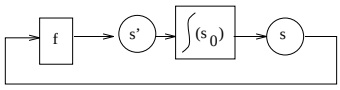

Figure 1.2
A constraint net representing $\dot{s} = f(s)$

an equilibrium of $\dot{s} = f(s)$ iff $f(s^*) = 0$.

LEMMA 4 A set $S^* = \{s^* | f(s^*) = 0\} \subset \Omega$ is an asymptotically stable equilibrium of the state integration system $\dot{s} = f(s)$ if f is continuous at S^* and S^* is the unique minimum of $-\int f(s)ds$ in Ω. If $\Omega = S$, S^* is an asymptotically stable equilibrium in the large.

Proof: Let $V(s) = -\int f(s)ds$ be defined on a neighborhood of S^*. V is a Liapunov function for $\dot{s} = f(s)$ and S^* since $\dot{V}(s) = -f^2(s) \leq 0$. Furthermore, $\dot{V}(s) < 0, \forall s \notin S^*$ since $f(s) \neq 0$. □

1.5 Dynamic Properties of Constraint Methods

In this section, we examine some typical constraint methods and their dynamic properties. In particular, we discuss two types of constraint satisfaction problem, namely, global consistency and optimization, for four classes of relation: relations on finite domains, and linear, convex and nonlinear relations in n-dimensional Euclidean space $\langle \mathcal{R}^n, d_n \rangle$, where $d_n(x, y) = |x - y| = \sqrt{\Sigma_{i=1}^n (x_i - y_i)^2}$.

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. On the other hand, *constrained optimization* can be considered as a combination of the two, which corresponds to solving the soft constraints within the solution set of the hard constraints.

There are two types of constraint method, discrete relaxation, which can be implemented as state transition systems, and differential optimization, which can be implemented as state integration systems. In the rest of this section, we demonstrate the use of both types of constraint method.

1.5.1 Global consistency

The problem of *global consistency* is to find a solution tuple which satisfies all the given constraints. Here we first discuss a projection method (PM) for solving convex constraints, and then study a method for solving global consistency of finite domain constraints (FM).

Projection method The projection method [8] can be used for solving convex constraints. A function $f : \mathcal{R}^n \rightarrow \mathcal{R}$ 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. 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. A set $R \subseteq \mathcal{R}^n$ is *convex* iff for any $\lambda \in (0, 1), x, y \in R$ implies $\lambda x + (1 - \lambda)y \in R$. If g is a convex function, $\{x | g(x) \leq 0\}$ is a convex set.

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 5 (From [8]) Let $R \subset \mathcal{R}^n$ be closed and convex. 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, $\{X_i\}_{i \in I}$, each representing a constraint. The problem is to solve $\{X_i\}_{i \in I}$, or to find $\cap_I X_i$. Let $P(x) = P_{X_l}(x)$ be a projection of x to a least satisfied set X_l, i.e., $d(x, X_l) = \max_I d(x, X_i)$. 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 $0 < \lambda < 2$.

Let PM be a constraint net representing the projection method. The following theorem is derived from [8].

THEOREM 3 PM solves $\{X_i\}_{i \in I}$ globally if all the X_i's are convex.

Proof: Let $X^* = \cap_I X_i$ be the solution set of the problem. First of all, it is easy to see that if $x^* \in X^*$ 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 and $x^* \in X^*$ as follows.

$$|f(x) - x^*|^2 = |x + \lambda(P(x) - x) - x^*|^2$$

$$= |x - x^*|^2 + \lambda^2 |P(x) - x|^2 + 2\lambda(x - x^*)^T(P(x) - x)$$

$$= |x - x^*|^2 + (\lambda^2 - 2\lambda)|P(x) - x|^2 + 2\lambda(P(x) - x)^T(P(x) - x^*)$$

$$\leq |x - x^*|^2 - \lambda(2 - \lambda)|P(x) - x|^2 \quad \text{according to Lemma 5}$$

$$\leq |x - x^*|^2 \quad \text{since } 0 < \lambda < 2.$$

Therefore, $d(f(x), X^*) \leq d(x, X^*)$. Thus, X^* is stable.

Furthermore, $|f^k(x) - x^*|$ is nonincreasing and bounded below. Therefore, $|f^k(x) - x^*|$ has a limit and $\max_I d(f^k(x), X_i)$ approaches 0. According to [8], $\lim_{k \to \infty} d(f^k(x), X^*) = 0$, since $\mathcal{R}^n$ is finite dimensional. Thus, X^* is an asymptotically stable equilibrium of PM in the large, i.e., PM solves the problem globally. □

The projection method can be used to solve a set of inequality constraints, i.e., $X_i = \{x | g_i(x) \leq 0\}$, where each g_i is a convex function.

Linear functions are convex. Therefore, the projection method can be applied to a set of linear inequalities $Ax \leq b$, where $x = \langle x_1, \cdots, x_n \rangle \in \mathcal{R}^n$. Let A_i be the ith 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, a simultaneous projection method is given in [3], in which $f(x) = x + \lambda \Sigma_{j \in J} w_j (P_j(x) - x)$ where $J \subseteq I$ is an index set of violated constraints, $w_j > 0$ and $\Sigma_{j \in J} w_j = 1$. A similar method is given in [21] in which $f(x) = x + \lambda(P_S(x) - x)$ where $S = \{x | \Sigma_{j \in J} w_j g_j(x) \leq 0\}$, with the same assumption about J and w_j. 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 kth subproblem. The whole problem can be solved by combining the results of $\{f_1, \cdots, f_K\}$.

Finite constraint satisfaction Many problems can be formalized as finite constraint satisfaction problems (FCSPs), which can be represented by constraint networks [23]. Formally, a *constraint network* C is a quadruple $\langle V, dom, A, con \rangle$ where

* V is a set of variables, $\{v_1, v_2, \cdots, v_N\}$,
* associated with each variable v_i is a finite domain $d_i = dom(v_i)$,
* A is a set of arcs, $\{a_1, a_2, \cdots, a_n\}$,
* 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 FCSP problem is to find one or all relation tuples in $sol(C) = r_1 \bowtie \cdots \bowtie r_n$.

An FCSP can be solved using various methods [4, 6, 11, 12, 14], one of which is to find the minimal network [14]. Let $Scheme(C) = \{R_1, \cdots, R_n\}$ be the scheme of a constraint network C. The *minimal network 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 Π_{R_i} is a pro-

jection operator. Here we present a relaxation method (FM) which finds the minimal network of a constraint network with an acyclic scheme. Such methods have been studied by many researchers, for instance, [7, 17, 23]. We examine the properties of the method within the framework of dynamic processes.

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

Let FM be a constraint net representing a state transition system $\langle \mathcal{C}, f \rangle$.

THEOREM 4 FM solves the minimal network problem globally for $\mathcal{C}$ if the scheme of $\mathcal{C}$ is acyclic.

Proof: It is clear that a minimal 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 sets of relation tuples r_1, r_2 on the same relation scheme R 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 $d(C_1, C_2) = \sqrt{\Sigma_{Scheme(\mathcal{C})} d_R^2(r_1, r_2)}$. Let us define a function L on $\mathcal{C}$ as $L(C) = \sqrt{\Sigma_{Scheme(\mathcal{C})} |r|^2}$. It is easy to check that L is a Liapunov function for $\langle \mathcal{C}, f \rangle$ and C^*.

If the scheme of $\mathcal{C}$ is acyclic, an equilibrium implies a minimal network [23], i.e., if $C \neq C^*$, $L(f(C)) < L(C)$. Furthermore, f is continuous since the metric space on $\mathcal{C}$ is discrete. According to Lemma 3, C^* is an asymptotically stable equilibrium. □

1.5.2 Unconstrained optimization

The problem of *unconstrained optimization* is to minimize a function $\mathcal{E} : \mathcal{R}^n \rightarrow \mathcal{R}$. Global consistency can be solved via unconstrained optimization. For instance, given a set of equations $g_i(x) = 0, i = 1 \cdots n$, let $\mathcal{E}_g(x) = \Sigma_{i=1}^k w_i g_i^2(x)$ where $w_i > 0$ and $\Sigma_{i=1}^k w_i = 1$. 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. Here we first discuss two methods for this problem: the gradient method (GM) and Newton's method (NM), and then

study the schema model (SM) for solving finite constraint satisfaction problems by minimizing an energy function $\mathcal{E} : [0,1]^n \rightarrow \mathcal{R}$.

Gradient method The gradient method [16] is based on the *gradient descent* algorithm, where state variables slide downhill in the direction opposed to the gradient. Formally, if the function to be minimized is $\mathcal{E}(x)$ where $x = \langle x_1, \cdots, x_n \rangle$, then at any point, the vector that points in the direction of maximum increase of $\mathcal{E}$ is the gradient of $\mathcal{E}$. Therefore, the following gradient descent equations model the *gradient method*:

$$\dot{x}_i = -k_i \frac{\partial \mathcal{E}}{\partial x_i}, \quad k_i > 0. \tag{1.5.1}$$

Let $\mathcal{E} : \mathcal{R}^n \rightarrow \mathcal{R}$ be a function. Let GM be a constraint net representing the gradient descent equations (1.5.1). The following theorem specifies conditions under which GM solves the problem of local minimization of $\mathcal{E}$.

THEOREM 5 Let X^* be the set of local minima of $\mathcal{E}$. GM solves the problem if $\frac{\partial \mathcal{E}}{\partial x}$ is continuous at X^*. GM solves the problem globally if, in addition, $\mathcal{E}$ is convex.

Proof: According to Lemma 4, a local minimum is an asymptotically stable equilibrium. A set of local minima is also an asymptotically stable equilibrium. If $\mathcal{E}$ is convex, X^* is the unique minimal set, which is an attractor in the large. □

Newton's method Newton's method [19] minimizes a second-order approximation of the given 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 minimizes 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 a constraint net representing Newton's method. The following theorem specifies conditions under which NM solves the problem of local minimization of a function $\mathcal{E}$.

THEOREM 6 Let $X^* \in \mathcal{R}^n$ be the set of local minima of $\mathcal{E}$. NM solves the problem if $|J(x^*)| \neq 0$, $\forall x^* \in X^*$, i.e., $\mathcal{E}$ is strictly convex at each local minimal point. NM solves the problem globally if, in addition, $\mathcal{E}$ is convex.

Proof: First, we prove that $\forall x^* \in X^*$, $x^* = f(x^*)$ and $|J(x^*)| \neq 0$ imply that 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 $x \in N^\epsilon(x^*)$, $|f(x) - f(x^*)| \leq \lambda|x - x^*|$ for $0 < \lambda < 1$. Therefore, $\lim_{k \to \infty} |f^k(x) - x^*| = 0$ and x^* is asymptotically stable. Therefore, X^* is an asymptotically stable equilibrium. If $\mathcal{E}$ is convex, x^* is the unique minimal point, which is an attractor in the large. □

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

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

$$\mathcal{E}(x) = -(\Sigma_{ij} w_{ij} x_i x_j + \Sigma_i b_i x_i) = -(x^T W x + b^T x)$$

where $x_i \in [0, 1]$ indicates the activation value and b_i specifies the bias for unit i. Value 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. The problem is to minimize $\mathcal{E}$ within the closed set $[0, 1]^n$.

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

Let SM be a constraint net representing the schema model.

THEOREM 7 SM solves the problem of minimizing $\mathcal{E}$ if $|n_i(x)| \leq 1$ for any i and x.

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

1. $n_i(x^*) > 0$: In this case, $x_i^* = 0$ and $|f_i(x) - x_i^*| = |f_i(x)| = |1 - n_i(x)| \times |x_i| \leq |x_i - x_i^*|$.

2. $n_i(x^*) < 0$: In this case, $x_i^* = 1$ and $|f_i(x) - x_i^*| = |f_i(x) - 1| = |1 + n_i(x)| \times |x_i - 1| \leq |x_i - x_i^*|$.

3. $n_i(x^*) = 0$ and $n_i(x) \geq 0$: $|f_i(x) - x_i^*| = |(1 - n_i(x))x_i - x_i^*| = |x_i - x_i^* - n_i(x)x_i| \leq |x_i - x_i^*|$.

4. $n_i(x^*) = 0$ and $n_i(x) \leq 0$: $|f_i(x) - x_i^*| = |(1 + n_i(x))x_i - n_i(x) - x_i^*| = |x_i - x_i^* - n_i(x)(1 - x_i)| \leq |x_i - x_i^*|$.

Therefore, $\forall x \in \Omega, |f_i(x) - x_i^*| \leq |x_i - x_i^*|$ and $|f(x) - x^*| \leq |x - x^*|$. Therefore, x^* is stable. Thus, X^* is a stable equilibrium.

Furthermore, let $X_0^* \subseteq X^*$ be a set which takes a unique minimum in its neighborhood. X_0^* is convex and closed since it is an intersection of a set of linear equations on or within the boundary. Let Ω be a strict neighborhood of X_0^*. If $x \in \Omega - X_0^*$, let x^* be the projection of x on X_0^*. There exists x_i, $|f_i(x) - x_i^*| < |x_i - x_i^*|$, so $|f(x) - x^*| < |x - x^*|$ and $d(f(x), X_0^*) < d(x, X_0^*)$. Since f is continuous, according to Lemma 3 (with $V(x) = d(x, X^*)$), X_0^* is asymptotically stable, so is X^*. □

1.5.3 Constrained optimization

Unconstrained optimization can be used to solve soft constraints as well as hard constraints. *Constrained optimization* is a problem of solving (soft) constraints subject to the satisfaction of a set of hard constraints, or solving a constraint satisfaction problem within a subspace characterized by the set of hard constraints.

The prototypical constrained optimization problem can be stated as [16]: locally minimize $f(x)$, subject to $g(x) = 0$, where $g(x) = 0$ is a set of equations describing a manifold of the state space. There are various methods for solving the constrained optimization problem. Here we focus on methods derived from the gradient method. During constrained optimization, the state x should be attracted to the manifold $g(x) = 0$ and slide along the manifold until it reaches the locally smallest value of $f(x)$ on $g(x) = 0$.

Different methods arise from the design of the energy function $\mathcal{E}$ for minimizing $f(x)$ under constraints $g_k(x) = 0$ for $k = 0 \cdots m$. Let $\mathcal{E}_c$ be the energy function generated from the constraints, we have $\mathcal{E}(x) = f(x) + \mathcal{E}_c(x)$.

$*$ *Penalty Methods*: The penalty method constructs an energy term that penalizes violations of the constraints, i.e., $\mathcal{E}_c(x) = \Sigma_{k=0}^{m} c_k g_k^2(x)$.

$*$ *Lagrange Multipliers*: The Lagrange multiplier method introduces a Lagrange multiplier λ for each constraint and λ varies as long as its constraint is not satisfied, i.e., $\mathcal{E}_c(x) = \Sigma_{k=0}^{m} \lambda_k g_k(x)$. In addition, there is a set of differential equations for λ, i.e., $\dot{\lambda}_k = g_k(x)$.

The advantage of the penalty method is its simplicity; however, the constrained optimization problem may not be solved with finite c_i. The advantage of the Lagrange multiplier method is its ability to satisfy the hard constraints.

Let LM be the constraint net representing the Lagrange multiplier method. The following theorem specifies a condition under which LM solves the constrained optimization problem globally.

THEOREM 8 Let A be a matrix where $A_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} + \Sigma_{k=0}^{m} \lambda_k \frac{\partial^2 g_k}{\partial x_i \partial x_j}$. If A is positive definite, LM solves the constrained optimization problem $\min f(x)$ subject to $g_k(x) = 0$ globally.

Proof: Let

$$V(x) = \frac{1}{2}\Sigma_i \dot{x}_i^2 + \frac{1}{2}\Sigma_k g_k^2(x).$$

It has been shown in [16] that

$$\dot{V} = -\Sigma_{i,j} \dot{x}_i A_{ij} \dot{x}_j.$$

V is a Liapunov function for LM and the solution set. □

1.6 Summary

We have presented here a framework for constraint satisfaction. Figure 1.3 illustrates the overall approach. First, we view constraints as

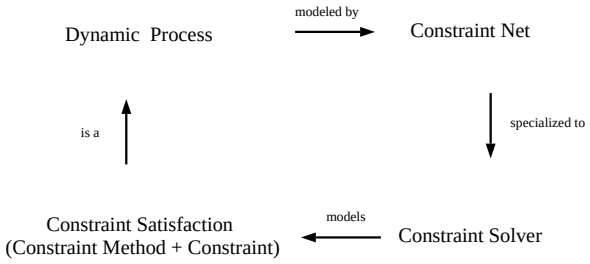

Figure 1.3
A framework for constraint satisfaction

relations and constraint satisfaction as a dynamic process of approaching the solution set of the constraints. Then, we explore the relationship between constraint satisfaction and constraint nets through constraint solvers.

Within this framework, *constraint programming* is seen as the creation of a constraint solver that solves the set of constraints. A constraint solver "solves" a set of constraints in the following sense (Figure 1.4). Given a constraint satisfaction problem C, and a discrete or continuous (time) constraint method, a constraint solver CS is generated. Starting from any initial state in the attraction basin of $sol(C)$, CS will approach $sol(C)$ asymptotically. In this framework, constraint programming is off-line and constraint satisfaction is on-line.

We have also studied various continuous and discrete time constraint methods, which can be realized by state integration systems and state transition systems, respectively.

This framework for constraint satisfaction has two advantages. First, the definition of constraint solvers relaxes the condition of solving constraints from finite computation to asymptotic stability. For example, many relaxation methods with the local convergence property are in fact "solvers" under this definition and many problems become "semi-computable" in this sense. This concept is very useful in practice and

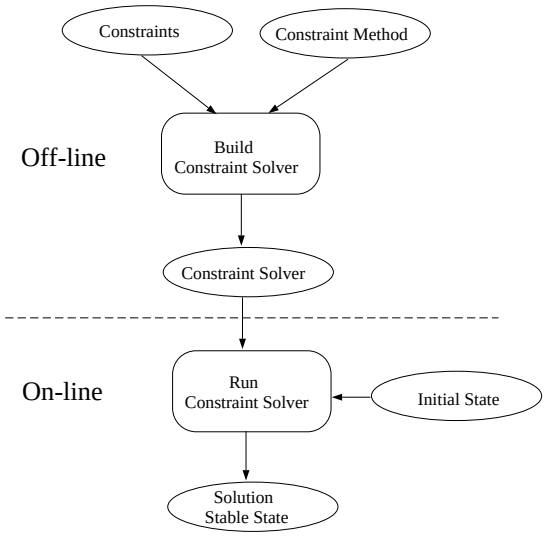

Figure 1.4
Constraint solvers and constraint satisfaction

can be used for generalizing Turing computability from discrete domains to continuous domains. Second, dynamic constraints can be solved in this framework as well. The importance of this characteristic is demonstrated here using an application to control synthesis.

1.7 Application to Control Synthesis

One of the significant applications of constraint solvers is the design of robot control systems [15]. A robotic system is a dynamic system consisting of a plant, a controller and an environment (Figure 1.5). The roles of these three subsystems can be characterized as follows:

* *Plant*: a plant is a set of entities which must be controlled to achieve certain requirements. For example, a robot arm with multiple joints, a car with throttle and steering, an airplane or a nuclear power plant can be considered as the *plant* of a robotic system.

* *Controller*: a controller is a set of sensors and actuators, which, together with software/hardware computational systems, senses the states

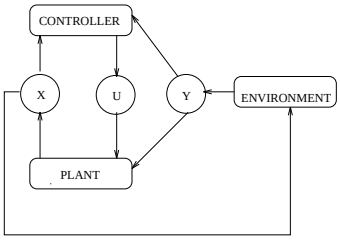

Figure 1.5
A robotic system

of the plant (X) and the environment (Y), and computes desired inputs (U) to actuate the plant. For example, an analog circuit, a program in a digital computer, various sensors and actuators can be considered as parts of the *controller* of a robotic system.

$*$ *Environment*: an environment is a set of entities beyond the (direct) control of the controller, with which the plant may interact. For example, obstacles to be avoided, objects to be reached, and rough terrain to be traversed can be considered as the *environment* of a robotic system.

In most cases, desired goals, safety requirements and physical restrictions of a robotic system can be specified by a set of constraints on variables $U \cup X \cup Y$ (Figure 1.5). The controller is then synthesized to regulate the system to satisfy the set of constraints.

A controller is *constraint-based* iff the integration of the controller and the plant solves the set of constraints, in response to the state of the environment. Consider the design of a tracking system S which chases a target T. Let x be the position of S and y be the position of T: the constraint to be satisfied is $x = y$. Suppose the plant follows the dynamics $u = \dot{x}$ where u is the control input. One possible design for the controller uses the following feedback control law $u = k(y-x), k > 0$ where the distance between the target and the current position $y-x$ can be sensed. This controller is constraint-based since $\dot{x} = k(y - x)$ solves $x = y$ for any parameter y.

The constraint techniques we have introduced can be applied to control synthesis and behavior verification for robotic systems [22].

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, the Canadian Institute for Advanced Research and the Institute for Robotics and Intelligent Systems.

Bibliography

[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.

[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. A foundation for the design and analysis of robotic systems and behaviors, 1994. PhD thesis, forthcoming.

[23] 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.

[24] Y. Zhang and A. K. Mackworth. Constraint Nets: A semantic model for hybrid dynamic systems, 1994. Accepted for TCS Special Issue on Hybrid Systems.

[25] Y. Zhang and A. K. Mackworth. Will the robot do the right thing? In *Proc. Artificial Intelligence 94*, pages 255 – 262, Banff, Alberta, May 1994.