

Conditional Existence of Variables in Generalized Constraint Networks*

James Bowen[†] and Dennis Bahler[‡]
Dept. of Computer Science
Box 8206, North Carolina State University
Raleigh, NC 27695-8206

To appear in Proceedings of AAAI-91, Anaheim CA, July 1991

Abstract

Classical constraint systems require that the set of variables which exist in a problem be known *ab initio*. However, there are some applications in which the existence of certain variables is dependent on conditions whose truth or falsity can only be determined dynamically. In this paper, we show how this conditional existence of variables can be handled in a mathematically well-founded fashion by viewing a constraint network as a set of sentences in free logic. Based on these ideas, we have developed, implemented and applied, a constraint language in which any sentence in full first-order free logic, about a many-sorted universe of discourse which subsumes \mathfrak{R} , is a well-formed constraint.

1 Introduction

Classical constraint systems [8] require that the set of variables which exist in a problem be known *ab initio*. However, there are some applications [4, 9] in which not just the value of certain variables, but also their very existence, depends on conditions whose truth can only be determined dynamically. In this paper, we show how this conditional existence of variables can be handled in a mathematically well-founded fashion by viewing a constraint network as a set of sentences in first-order free logic (FOFL) [7]. As well as having a better-developed theoretical underpinning, this approach is more general than that presented in [9]. We briefly review a language, based on these ideas, which we have developed, implemented and applied to a range of CAD applications.

*This work was supported in part by NSF grant number DDM-8914200.

[†]jabowen@adm.csc.ncsu.edu

[‡]drb@adm.csc.ncsu.edu

2 Classical Constraint Processing

In the literature, several different definitions are given for classical constraint networks, with varying degrees of formality. However, they may all be regarded as variations of the following theme:

Definition 1, Constraint Network:

A constraint network is a triple $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$. \mathbf{D} is a finite set of $p > 0$ domains, the union of whose members forms a universe of discourse, \mathcal{U} . \mathbf{X} is a finite tuple of $q > 0$ non-recurring variables. \mathbf{C} is a finite set of $r \geq q$ constraints. In each constraint $C_k(T_k) \in \mathbf{C}$, T_k is a sub-tuple of \mathbf{X} , of arity a_k ; $C_k(T_k)$ is a subset of the a_k -ary Cartesian product \mathcal{U}^{a_k} . In \mathbf{C} , there are q unary constraints of the form $C_k(X_j) = D_i$, one for each variable X_j in \mathbf{X} , restricting it to range over some domain $D_i \in \mathbf{D}$.

The overall network constitutes an intensional specification of a joint possibility distribution for the values of the variables in the network. This distribution is a q -ary relation on \mathcal{U}^q , called the *intent* of the network:

Definition 2, The Intent of a Constraint Network:

The intent of a constraint network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$ is

$$\Pi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} = E_1(\mathbf{X}) \cap \dots \cap E_r(\mathbf{X}),$$

where, for each constraint $C_k(T_k) \in \mathbf{C}$, $E_k(\mathbf{X})$ is its cylindrical extension [5] in \mathcal{U}^q .

Three forms of constraint satisfaction problem (CSP) can be distinguished:

Definition 3, The Decision CSP:

Given a network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$, decide whether $\Pi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ is non-empty.

Definition 4, The Exemplification CSP:

Given a network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$, return some tuple from $\Pi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$, if $\Pi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ is non-empty, or nil otherwise.

Definition 5, The Enumeration CSP:

Given a network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$, return $\Pi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$.

3 Classical Constraint Networks and Classical Logic

Classical constraint processing may be regarded as semantic modeling in first-order classical logic (FOCL), in the following sense. The constraints in \mathbf{C} correspond to sentences of an FOCL theory Γ , written in some first-order language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$.¹ The variables in \mathbf{X} correspond to object symbols, from \mathcal{K} ,

¹In the language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$, \mathcal{P} is a set of relation or predicate symbols, \mathcal{F} is a set of function symbols and \mathcal{K} is a set of object or constant symbols.

which appear in Γ . The decision CSP corresponds to deciding whether Γ is satisfied under any model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$ of the language \mathcal{L} , where \mathcal{U} is the union of the domains in \mathbf{D} and \mathcal{I} is an interpretation function from the symbols of \mathcal{L} to entities in, and relations over, \mathcal{U} . The exemplification CSP corresponds to finding \mathcal{I} for one such model, while the enumeration CSP corresponds to finding \mathcal{I} for all such models.

3.1 Example 1

Consider the task of semantically modeling the theory $\Gamma = \{positive(area), nonnegative(tensile_stress), nonnegative(load), load = area * tensile_stress, tensile_stress \leq 200\}$, in $\mathcal{L} = \{positive, nonnegative, =, \leq, \{*\}, \mathcal{R} \cup \{area, tensile_stress, load\}\}$.² Suppose we are given a universe of discourse $\mathcal{U} = \mathfrak{R}$ and a partial interpretation function \mathcal{I}_p for \mathcal{L} , which provides an interpretation for each predicate and function symbol of \mathcal{L} as well as a total one-to-one mapping from \mathcal{R} to the finitely expressible rationals $Q_f \subset Q \subset \mathfrak{R}$. That is, as well as object mappings of the form $200 \rightarrow 200$, \mathcal{I}_p contains these predicate and function mappings:

$$\begin{array}{ll}
positive & \rightarrow \mathfrak{R}^+ \\
nonnegative & \rightarrow \mathfrak{R}^{0+} \\
= & \rightarrow \{\langle X, Y \rangle \mid X \in \mathcal{U} \wedge Y \in \mathcal{U} \wedge \\
& \text{EQUALS}(X, Y)\} \\
\leq & \rightarrow \{\langle X, Y \rangle \mid X \in \mathfrak{R} \wedge Y \in \mathfrak{R} \wedge \\
& \text{LEQ}(X, Y)\} \\
* & \rightarrow \{\langle X, Y, Z \rangle \mid X \in \mathfrak{R} \wedge Y \in \mathfrak{R} \wedge \\
& Z \in \mathfrak{R} \wedge \text{EQUALS}(Z, \text{TIMES}(X, Y))\}.
\end{array}$$

The constraint network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$ corresponding to this situation is a possibility distribution for interpretations of the remaining uninterpreted object symbols of \mathcal{L} , such that Γ is satisfied. The components of this networks are: $\mathbf{D} = \{\mathfrak{R}^+, \mathfrak{R}^{0+}, \mathfrak{R}\}$, $\mathbf{X} = \langle area, tensile_stress, load \rangle$ and $\mathbf{C} = \{C_1(area), C_2(tensile_stress), C_3(load), C_4(area, tensile_stress, load), C_5(tensile_stress)\}$. The constraints are defined as follows:

$$\begin{array}{l}
C_1(area) = \mathfrak{R}^+ \\
C_2(tensile_stress) = \mathfrak{R}^{0+} \\
C_3(load) = \mathfrak{R}^{0+} \\
C_4(area, tensile_stress, load) = \\
\quad \{\langle X, Y, Z \rangle \mid X \in \mathfrak{R} \wedge Y \in \mathfrak{R} \wedge Z \in \mathcal{U} \wedge \text{EQUALS}(Z, \text{TIMES}(X, Y))\} \\
C_5(tensile_stress) = \{X \mid X \in \mathfrak{R} \wedge \text{LEQ}(X, 200)\}.
\end{array}$$

² \mathcal{R} is the set of object symbols composed from the characters $+$, $-$, $.$ and $0..9$ according to a grammar for real numeric strings. \mathcal{R} is distinguished from \mathfrak{R} , the set of real numbers. In this paper, to distinguish between symbols of \mathcal{L} and entities of \mathcal{U} , we use typewriter font for the latter. Thus, $200 \in \mathcal{R}$ is an object symbol while $200 \in \mathfrak{R}$ would be in a universe of discourse.

Each sentence in the theory has a corresponding constraint in the network, the definition of which depends on whatever information is provided, by the partial interpretation \mathcal{I}_p , about the symbols appearing in the sentence. Consider, for example, $C_4(\textit{area}, \textit{tensile_stress}, \textit{load})$, which corresponds to the sentence $\textit{load} = \textit{area} * \textit{tensile_stress}$. There are three variables in this constraint, corresponding to the three object symbols in the sentence. The restriction imposed by the constraint is derived from the interpretations in \mathcal{I}_p for the symbols $=$ and $*$ in the sentence. Consider $C_5(\textit{tensile_stress})$, which corresponds to the sentence $\textit{tensile_stress} = < 200$. Although two object symbols appear in the sentence, there is only one variable in the constraint, because \mathcal{I}_p provides an interpretation for 200.

3.2 Example 2: The Inadequacy of Classical Constraint Networks

Constraint networks and constraint processing have wide applicability, in areas as diverse as design [2, 9] and computer vision [10]. The network in Example 1 represents some considerations affecting the cross-sectional *area* of, and *tensile_stress* in, a bar carrying a *load*. However, consider the following:

Example 2: Extend Example 1 to incorporate considerations about the cross-section of the bar. Circular and rectangular shapes are allowed; circular bars have a cross-sectional radius; rectangular bars have a breadth and height.

This example is beyond the expressive competence of classical constraint networks, because the *radius* cannot exist if the *height* and *breadth* exist. We have encountered a fundamental ontological inadequacy of classical logic: its inadequacy for reasoning in depth about existence or non-existence [6]. In what follows, we show how this difficulty can be overcome, by using FOFL [7], rather than FOCL, as the theoretical basis for constraint networks.

4 Free Logic

In classical logic, a model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$ for a first-order language $\mathcal{L} = \langle \mathcal{P}, \mathcal{K} \rangle$ ³ comprises a universe \mathcal{U} and an interpretation function \mathcal{I} . The interpretation function \mathcal{I} , assigns to each predicate symbol $p \in \mathcal{P}$ some relation $\mathcal{I}(p)$ over \mathcal{U} and, because of the absence of function symbols in \mathcal{L} , provides a mapping from \mathcal{K} to \mathcal{U} which is total and surjective. That is, the function \mathcal{I} assigns to each object symbol $\kappa \in \mathcal{K}$ some element $u \in \mathcal{U}$ and makes each $u \in \mathcal{U}$ the image of at least one $\kappa \in \mathcal{K}$.

Free logic [7] differs from classical logic in that a free logic model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S} \rangle$ for a first-order language $\mathcal{L} = \langle \mathcal{P}, \mathcal{K} \rangle$ contains a third item, a story \mathcal{S} . As before, the function \mathcal{I} assigns to each $p \in \mathcal{P}$ a relation $\mathcal{I}(p)$ over \mathcal{U} . However,

³For simplicity, in this section we discuss a function-free language, since an n -ary function is really an $(n + 1)$ -ary relation.

although \mathcal{I} provides a surjective mapping from \mathcal{K} to \mathcal{U} (i.e., every $u \in \mathcal{U}$ is the image of some $\kappa \in \mathcal{K}$), this mapping need not be total: it need not assign a $u \in \mathcal{U}$ to every $\kappa \in \mathcal{K}$. The story \mathcal{S} is a (possibly empty) set of atomic sentences, each of which contains some $\kappa \in \mathcal{K}$ to which \mathcal{I} has assigned no $u \in \mathcal{U}$. A special symbol Ω , additional to \forall and \exists , is used in atoms like $(\Omega \kappa)$ or $\neg(\Omega \kappa)$ to talk about whether or not \mathcal{I} maps $\kappa \in \mathcal{K}$ to some $u \in \mathcal{U}$; $(\Omega \kappa)$ can be read as “ κ designates some element of \mathcal{U} ” or simply “the object denoted by κ exists.”⁴

The model-theoretic rules for determining truth in FOFL are as given below. Rules (a), (g) and (h) are different from their counterparts in FOCL, while rule (i) has no counterpart.

- (a) $\mathcal{M} \models p(a_1, \dots, a_n)$ iff $p(a_1, \dots, a_n)$ is in \mathcal{S} or $\langle \mathcal{I}(a_1), \dots, \mathcal{I}(a_n) \rangle$ is in $\mathcal{I}(p)$.
- (b) $\mathcal{M} \models \neg A$ iff $\mathcal{M} \not\models A$.
- (c) $\mathcal{M} \models A \wedge B$ iff $\mathcal{M} \models A$ and $\mathcal{M} \models B$.
- (d) $\mathcal{M} \models A \vee B$ iff $\mathcal{M} \models A$ or $\mathcal{M} \models B$.
- (e) $\mathcal{M} \models A \Rightarrow B$ iff $\mathcal{M} \not\models A$ or $\mathcal{M} \models B$.
- (f) $\mathcal{M} \models A \Leftrightarrow B$ iff $(\mathcal{M} \not\models A \text{ and } \mathcal{M} \not\models B)$ or $(\mathcal{M} \models A \text{ and } \mathcal{M} \models B)$.
- (g) $\mathcal{M} \models (\forall X)A$ iff $\mathcal{M} \models (\kappa/X)A$ for every $\kappa \in \mathcal{K}$ to which \mathcal{I} has assigned a $u \in \mathcal{U}$.
- (h) $\mathcal{M} \models (\exists X)A$ iff $\mathcal{M} \models (\kappa/X)A$ for some $\kappa \in \mathcal{K}$ to which \mathcal{I} has assigned a $u \in \mathcal{U}$.
- (i) $\mathcal{M} \models (\Omega \kappa)$ iff $\kappa \in \mathcal{K}$ is assigned a $u \in \mathcal{U}$ by \mathcal{I} .

5 Free Constraint Processing

Definition 6, Free Constraint Network:

A free constraint network is a quadruple $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$. \mathbf{D} is a non-empty, finite set of domains, the union of whose members forms a universe of discourse \mathcal{U} . ∇ is a distinguished entity, $\nabla \notin \mathcal{U}$. \mathbf{X} is a non-empty, possibly infinite, tuple of non-recurring variables. \mathbf{C} is a finite set of $r > 0$ constraints. Each constraint $C_k(T_k) \in \mathbf{C}$ is a possibility distribution which restricts the existence of the variables in T_k , a (possibly infinite) sub-tuple of \mathbf{X} , and the values that these variables may assume if they exist. $C_k(T_k)$ is a subset of the Cartesian product $(\prod_{X_j \text{ in } T_k} (v(X_j)))$ where $v(X) = (\mathcal{U} \cup \{\nabla\})$.

Free constraint networks are a generalization of classical networks. A classical constraint network is a free constraint network containing a finite number of variables, all of which exist; that is, in a classical network \mathbf{X} is a finite tuple, and the constraint(s) in \mathbf{C} unconditionally require(s) every variable in \mathbf{X} to exist. In contrast, a free constraint network may, in general, contain an infinite number of variables, none of which need, in general, exist.

⁴ Ω may also be regarded as a unary predicate symbol whose interpretation is all of \mathcal{U} ; when viewed as such, Ω is special, in the sense that no atom with Ω as predicate can be in the story \mathcal{S} .

The intent of a free constraint network is a joint possibility distribution on the existence of the variables in the network and on the values that these variables may assume, if they exist:

Definition 7, Intent of a Free Constraint Network:

The intent of a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$ is

$$\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} = F_1(\mathbf{X}) \cap \dots \cap F_r(\mathbf{X}),$$

where $F_j(\mathbf{X})$ is the cylindrical extension of the constraint $C_j(T_j)$ in the Cartesian product $(\prod_{X_j \text{ in } \mathbf{X}} (v(X_j)))$, the existence and value space for \mathbf{X} .

Definition 8, Free Decision CSP:

Given a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$, decide whether $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ is non-empty.

Definition 9, Free Exemplification CSP:

Given a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$, return some tuple in $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$, if $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ is non-empty; otherwise return nil.

Definition 10, Free Enumeration CSP:

Given a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$, return $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$.

The possibility that some variables in a free constraint network need not exist (need not be mapped into \mathcal{U}) means that several additional forms of CSP may be defined [1]. Here, however, we are interested in just two of these: the Minimal Exemplification CSP and the Minimal Enumeration CSP. These are defined as follows.

Definition 11, Interpretation Set:

Given a mapping from a tuple of variables T_k to a tuple of values t_k , the expression $\iota(T_k, t_k)$ denotes the corresponding interpretation set, the set of all those variable to value mappings in which the variables denote something in the universe \mathcal{U} .

Definition 12, Minimal Intent of a Free Constraint Network:

The minimal intent of a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$, written $\mu\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$, is

$$\mu\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} = \{Y \mid Y \in \Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} \wedge \neg((\exists Z) (Z \in \Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} \wedge \iota(\mathbf{X}, Z) \subset \iota(\mathbf{X}, Y)))\}.$$

Definition 13, Minimal Exemplification CSP:

Given a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$, return some tuple in $\mu\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$, if $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ is non-empty; otherwise return nil.

Definition 14, Minimal Enumeration CSP:

Given a free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$, return $\mu\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$.

6 Free Constraint Networks and Free Logic

A free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$ specifies the possibility distribution for the existence, as well as the interpretation, of object symbols which appear in

a free logic theory Γ , written in a first-order language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$. As in the classical case, the definition of the network depends on Γ , the universe of discourse \mathcal{U} , and a partial interpretation function \mathcal{I}_p for \mathcal{L} which gives interpretations for all function and predicate symbols, and some of the object symbols, of \mathcal{L} . However, the free network definition also depends on the story \mathcal{S} .

Taking any tuple Y in $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ and computing $\mathcal{I} = \mathcal{I}_p \cup \iota(\mathbf{X}, Y)$ produces an interpretation function \mathcal{I} such that the theory Γ is satisfied under the free logic model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S} \rangle$ of \mathcal{L} . Similarly, taking any tuple Z in $\mu\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}}$ and computing $\mathcal{I} = \mathcal{I}_p \cup \iota(\mathbf{X}, Z)$ produces a minimal interpretation function \mathcal{I} such that the theory Γ is satisfied under the minimal free logic model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S} \rangle$ of \mathcal{L} .

6.1 Example 3

Consider, for example, the following situation:

Language: $\mathcal{L} = \langle \{p, q\}, \{\}, \{1, 2, 3, a, b, c\} \rangle$

Theory: $\Gamma = \{p(a, b), p(a, c) \Rightarrow q(a, c)\}$

Universe of discourse: $\mathcal{U} = \{1, 2, 3\}$

Story: $\mathcal{S} = \{p(a, b)\}$

Partial Interpretation: $\mathcal{I}_p = \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3,$
 $p \rightarrow \{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 3 \rangle\}, q \rightarrow \{\langle 1, 2 \rangle, \langle 2, 3 \rangle\}$.

The free constraint network $\langle \mathbf{D}, \nabla, \mathbf{X}, \mathbf{C} \rangle$ corresponding to this is the possibility distribution for the existence and interpretation of a, b and c . The components \mathbf{D} , \mathbf{X} and \mathbf{C} of the network are as follows: $\mathbf{D} = \{\{1, 2, 3\}\}$; $\mathbf{X} = \langle a, b, c \rangle$; $\mathbf{C} = \{C_1(a, b), C_2(a, c)\}$.

$C_1(a, b)$ corresponds to $p(a, b) \in \Gamma$. Since $p(a, b) \in \mathcal{S}$, at least one of a or b must not exist. Thus, $C_1(a, b) = (\{\nabla\} \times \{\nabla, 1, 2, 3\}) \cup (\{\nabla, 1, 2, 3\} \times \{\nabla\})$. $C_2(a, c)$ corresponds to $p(a, c) \Rightarrow q(a, c)$. Since this can be rewritten as $\neg p(a, c) \vee q(a, c)$, $C_2(a, c)$ can be defined as the union of two sets, one for each disjunct. Since $p(a, c) \notin \mathcal{S}$, $p(a, c)$ is false if a does not exist or if c does not exist or if a and c both exist but do not satisfy p . Since $q(a, c) \notin \mathcal{S}$, $q(a, c)$ is true iff a and c both exist and satisfy q . Calculating the union of the corresponding possibility distributions, and simplifying, we get $C_2(a, c) = (\{\nabla, 1, 2, 3\} \times \{\nabla, 1, 2, 3\}) - \{\langle 3, 3 \rangle\}$. Intersecting the cylindrical extensions of these constraints, we get $\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} = (((\{\nabla\} \times \{\nabla, 1, 2, 3\}) \cup (\{\nabla, 1, 2, 3\} \times \{\nabla\})) \times \{\nabla, 1, 2, 3\}) - \{\langle 3, \nabla, 3 \rangle\}$, which means that $\mu\Phi_{\mathbf{D}, \mathbf{X}, \mathbf{C}} = \{\langle \nabla, \nabla, \nabla \rangle\}$. This means that there are 27 different models $\mathcal{M} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S} \rangle, \mathcal{I} \supseteq \mathcal{I}_p$, of the language \mathcal{L} under which Γ is satisfied; there is one minimal model, $\langle \mathcal{U}, \mathcal{I}_p, \mathcal{S} \rangle$, under which neither a, b nor c exist.

6.2 Example 4

To see the impact of the story on the intent of a theory, consider the above situation, modified so that the story $\mathcal{S} = \{\}$. The only difference in the network is that $C_1(a, b) = \{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 3 \rangle\}$. Computing the intent and the minimal intent, we get $\Phi_{\mathbf{D}, \mathbf{x}, \mathbf{c}} = (\{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 3 \rangle\} \times \{\nabla, 1, 2, 3\}) - \{\langle 3, 3, 3 \rangle\}$, and $\mu\Phi_{\mathbf{D}, \mathbf{x}, \mathbf{c}} = \{(1, 2, \nabla), (2, 3, \nabla), (3, 3, \nabla)\}$, so, although c need not exist, both a and b must exist when the story is empty.

7 A Brief Overview of Galileo

A program in Galileo is a declarative specification of a free constraint network, analogous to the problem specifications in Examples 3 and 4. That is, in general, a Galileo program specifies a first-order language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$, a theory Γ containing sentences from that language, a universe of discourse \mathcal{U} , a partial interpretation function \mathcal{I}_p for \mathcal{L} and a story \mathcal{S} . Of these, only the theory Γ must always be specified explicitly. The Galileo run-time system provides a default language $\mathcal{L}_g = \langle \mathcal{P}_g, \mathcal{F}_g, \mathcal{R} \rangle$ in which \mathcal{R} contains the real numeric strings, \mathcal{P}_g contains names of standard predicates ($=, = <, \text{etc.}$), and \mathcal{F}_g contains names of standard functions ($*, +, \text{etc.}$). The run-time system also provides a universe of discourse $\mathcal{U}_g = \mathfrak{R}$, an interpretation function \mathcal{I}_g for \mathcal{L}_g in terms of \mathcal{U}_g and a story $\mathcal{S}_g = \{\}$.

The language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$ defined by a Galileo program has the following components: $\mathcal{P} = \mathcal{P}_g \cup \{\text{predicate symbols defined in the program}\}$; $\mathcal{F} = \mathcal{F}_g \cup \{\text{function symbols defined in the program}\}$; $\mathcal{K} = \mathcal{R} \cup \{\text{object symbols used in the program}\}$. The universe of discourse \mathcal{U} defined by a Galileo program is the union of $\mathcal{U}_g = \mathfrak{R}$ with any application-specific domains that are defined in the program. The partial interpretation function \mathcal{I}_p defined by the program is the union of the set of mappings in \mathcal{I}_g with the mappings provided by any definitions of application-specific domains, relations and functions that are in the program. The story \mathcal{S} defined by a Galileo program is the union of $\mathcal{S}_g = \{\}$ with any story provided in the program.

The Galileo program corresponding to Example 1 above is as follows:

```
area : positive.5
tensile_stress : nonnegative.
load : nonnegative.
load = area * tensile_stress.
tensile_stress = < 200.
```

The language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$ defined by this program has the following components: $\mathcal{P} = \mathcal{P}_g$; $\mathcal{F} = \mathcal{F}_g$; $\mathcal{K} = \mathcal{R} \cup \{area, tensile_stress, load\}$. The universe of discourse $\mathcal{U} = \mathcal{U}_g = \mathfrak{R}$. The partial interpretation function $\mathcal{I}_p = \mathcal{I}_g$. The

⁵A Galileo statement of the form “ $\kappa : p$ ” is merely an elliptical form of the Galileo statement “exists(κ) and $p(\kappa)$,” where “exists(κ)” is Galileo syntax for $(\Omega\kappa)$.

story $\mathcal{S} = \mathcal{S}_g = \{\}$. This program, in effect, defines a classical constraint network because the first three constraints specify that the object symbols *area*, *tensile_stress* and *load* must exist.

However, the following program, which corresponds to Example 2, does define a free constraint network, because the existence of *radius* or *breadth* and *height* is contingent on the interpretation of *shape*:

```

domain form ::= {circular, rectangular}.
area : positive.
tensile_stress : nonnegative.
load : nonnegative.
shape : form.
load = area * tensile_stress.
tensile_stress =< 200.
shape=circular implies exists(radius : positive) and
    area = 3.14159 * radius ^ 2.6
shape=rectangular implies
    exists(breadth : positive) and
    exists(height : positive)
    and area = breadth * height.
exists(radius) equiv
    not exists(breadth) and not exists(height).
exists(height) equiv exists(breadth).

```

The language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$ defined by this program has the following components: $\mathcal{P} = \mathcal{P}_g \cup \{form\}$; $\mathcal{F} = \mathcal{F}_g$; $\mathcal{K} = \mathcal{R} \cup \{circular, rectangular, area, tensile_stress, load, shape, radius, breadth, height\}$. The universe of discourse $\mathcal{U} = \mathcal{U}_g \cup \{circular, rectangular\} = \mathfrak{R} \cup \{circular, rectangular\}$. The partial interpretation function $\mathcal{I}_p = \mathcal{I}_g \cup \{form \rightarrow \{circular, rectangular\}, circular \rightarrow circular, rectangular \rightarrow rectangular\}$. The story $\mathcal{S} = \mathcal{S}_g = \{\}$. The intent of the network would be greatly expanded by the elimination of the last two statements in the program, but the minimal intent would be unaltered.

The following program, which corresponds to Example 3, contains two application-specific relation definitions, as well as an application-specific story:

```

relation p(number,number) ::= {(1,2), (2,3), (3,3)}.
relation q(number,number) ::= {(1,2), (2,3)}.
story ::= {p(a,b)}.
p(a,b).
p(a,c) implies q(a,c).

```

The language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$ defined by this program has the following components: $\mathcal{P} = \mathcal{P}_g \cup \{p, q\}$; $\mathcal{F} = \mathcal{F}_g$; $\mathcal{K} = \mathcal{R} \cup \{a, b, c\}$. The universe of discourse $\mathcal{U} = \mathcal{U}_g = \mathfrak{R}$. The partial interpretation function $\mathcal{I}_p = \mathcal{I}_g \cup \{p \rightarrow \{1, 2\}$,

⁶A Galileo expression of the form “exists($\kappa : p$)” is an elliptical form for the equivalent “exists(κ) and $p(\kappa)$.” The distinction between $\kappa : p$ and exists($\kappa : p$) is made purely for ease in parsing.

$\langle 2, 3 \rangle, \langle 3, 3 \rangle \}$, $q \rightarrow \{ \langle 1, 2 \rangle, \langle 2, 3 \rangle \}$. The story $\mathcal{S} = \mathcal{S}_g \cup \{p(a, b)\} = \{p(a, b)\}$. The language \mathcal{L} defined by this program is larger than the language in Example 3, because of the presence of the default language \mathcal{L}_g provided by the Galileo run-time system; however, the free constraint network defined by this program is identical to that in Example 3.

A full description of the Galileo language is beyond the scope of this paper. However, it should be noted that the language supports the full FOFL. Thus, for example, although all sentences in the example theories considered above were ground, Galileo supports theories which contain arbitrarily nested quantified sentences. Also, although none of the programs considered here involved application-specific functions, the definition and use of such functions is supported.

Since FOFL subsumes FOCL, satisfiability for FOFL is undecidable. There cannot exist, therefore, an algorithm capable of performing automatic constraint satisfaction on an arbitrary network specified in Galileo. However, various forms of CSP can be solved for a wide variety of networks, by the Galileo run-time system, which runs in either of two modes.

In autonomous mode, it is capable of solving the Minimal Enumeration CSP (and also, therefore, the Minimal Exemplification CSP and the decision CSP), for those free networks in which all variables have finite domains. It is capable of autonomously solving the Minimal Exemplification CSP (and, therefore, the decision CSP) for networks in which some or all variables have infinite domains, provided the networks are decidable by backtrack-free search.

In interactive mode, the run-time system can solve the Minimal Exemplification CSP (and, therefore, the decision CSP) for any network which the user, by non-monotonically augmenting the theory with additional assertions, renders decidable by backtrack-free search. Since any model of an augmented theory is also a model of the original theory, a solution to the Minimal Exemplification CSP for the augmented network is a solution for the original network.

Full details of constraint satisfaction in Galileo are beyond the scope of this paper. However, an inference algorithm called Compound Propagation [3] is a central part of both the backtracking searcher and the non-backtracking searcher. The algorithm, whose top-level is shown in Figure 1, involves interleaved application of three inference techniques:

- a version of local propagation of known states, extended to assimilate conditional existence; this is performed by invoking a lower-level procedure called LP;
- a version of arc consistency, generalized to infinite domains and constraints of arbitrary arity; this is performed by invoking procedure AC;
- a form of path consistency, generalized to infinite domains and constraints of arbitrary arity; this operation, which is performed by invoking proce-

cedure PC, is only applied to small portions of the network in certain very specific circumstances.

```

procedure CP(Assertions)
localvar  $Q_{lp}, Q_{ac}, Q_{pc}$ 
begin
   $Q_{lp} \leftarrow \text{Assertions}; Q_{ac} \leftarrow \{\}; Q_{pc} \leftarrow \{\};$ 
  repeat
    LP( $Q_{lp}, Q_{ac}, Q_{pc}$ );
    if  $Q_{ac} \neq \{\} \vee Q_{pc} \neq \{\}$ 
      then repeat
        if  $Q_{ac} \neq \{\}$ 
          then AC( $Q_{lp}, Q_{ac}, Q_{pc}$ );
        if  $Q_{lp} = \{\} \wedge Q_{pc} \neq \{\}$ 
          then PC( $Q_{ac}, Q_{pc}$ )
        until  $Q_{lp} \neq \{\} \vee Q_{ac} = \{\}$ 
      until  $Q_{lp} = \{\};$ 
  end

```

Figure 1: Top-level of the CP algorithm.

8 Example Free Logic Application

We have built several CAD applications using Galileo. Here, both to illustrate the utility in real-world applications of free logic⁷ and to touch, at least briefly, on quantification in Galileo, we provide one universally quantified constraint⁸ from an expert system on Design for Testability (DFT) [4]:

all X : crystal(X) implies
 (all Y : tester(Y) and Y.maxfreq < X.freq
 implies exists(X.ancillary_circuit : divider))

This constraint represents the following information about the design of printed wiring boards provided by our domain experts: “Each crystal on a board must have its own associated divider circuit if the crystal’s oscillation frequency exceeds the maximum oscillation frequency that can be handled by any of the pieces of test equipment that will be used to analyze the board.”

9 Comparative Discussion

By basing our work on free logic, we have been able to give our approach what seems to be a more developed theoretical underpinning than the only other

⁷Most practical applications involve the empty story.

⁸In Galileo, “ $(\forall X)(p(X))$ ” is written “all X : p(X).” This constraint uses Ω inside two levels of \forall . Also, as indicated by the dot notation in “Y.maxfreq,” it uses structured domains, another feature of Galileo which is beyond the scope of this paper.

known approach to conditional existence of network variables, that of Mittal and Falkenhainer [9]. In addition, our approach is more general. Their notion of a dynamic CSP is a special case of our concept of a minimal free enumeration CSP, as follows: all domains are finite; the number of potentially-existent variables is finite and a non-empty subset must be existent initially; quantified constraints are not allowed; the notion of conditional existence can be used in only a few different ways, whereas, in Galileo, an expression of the form *exists(object)* can appear anywhere that an atom may appear in a free logic theory.

In [9], the distinction between “active” and “inactive” variables refers only to denotation. Our approach, by contrast, maintains two distinctions: denotation versus non-denotation and existence versus non-existence in the name space. This difference has an important pragmatic consequence: although space limitations prevent us from showing it here, our approach can handle problems with an infinite number of potentially-existent variables, because storage space is not used by network variables until such time as their actual existence is proven or disproven. Indeed, in most cases even the *names* of potentially-existent variables are generated only when the variables’ existence is proven; these names can be regarded as equivalent to functional expressions involving the successor function in finite object vocabulary meta-theories of first-order languages. In practical applications, this happens for only a finite subset of the infinity of variables. Therefore no storage burden is imposed by the infinity of other variables whose possible existence is irrelevant to the particular minimal model constructed.

Free logic is a principled attempt to remedy an ontological inadequacy of classical logic. This logic has long been studied by philosophers and has recently [6] attracted some attention in the Knowledge Representation literature. Nevertheless, although there are several computer languages which are explicitly based on first-order classical logic, our language, Galileo, seems to be the first based on free logic.

References

- [1] Bowen J, and Bahler D, 1990, “Improving Ontological Expressiveness in Constraint Processing,” Technical Report, Department of Computer Science, North Carolina State University.
- [2] Bowen J, O’Grady P and Smith L, 1990, “A Constraint Programming Language for Life-Cycle Engineering,” *International Journal for Artificial Intelligence in Engineering*, 5(4), 206-220.
- [3] Bowen J, Bahler D and Paramasivam M, 1991, “Compound Propagation: A Constraint Monitoring Algorithm,” Technical Report TR-91-11, Department of Computer Science, North Carolina State University.

- [4] Dholakia A, Bowen J and Bahler D, 1990, "Rick: A DFT Advisor for Digital Circuit Design," Technical Report, Department of Computer Science, North Carolina State University.
- [5] Friedman G and Leondes C, 1969, "Constraint Theory, Part I: Fundamentals," *IEEE Transactions on Systems Science and Cybernetics*, ssc-5, 1, 48-56.
- [6] Hirst G, 1989, "Ontological assumptions in knowledge representation," *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 157-169.
- [7] Lambert K and van Fraassen B, 1972, *Derivation and Counterexample: An Introduction to Philosophical Logic*, Enrico, CA: Dickenson Publishing Company.
- [8] Mackworth A, 1987. "Constraint Satisfaction," in S. Shapiro (ed.), *The Encyclopedia of Artificial Intelligence*, New York: Wiley, 205-211.
- [9] Mittal S and Falkenhainer B, 1990, "Dynamic Constraint Satisfaction Problems," *Proceedings of the Eighth National Conference on Artificial Intelligence*, 25-32.
- [10] Mulder J, Mackworth A and Havens W, 1988, "Knowledge Structuring and Constraint Satisfaction: The Mapsee Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6), 866-879.