

Lexical Imprecision in Fuzzy Constraint Networks

James Bowen, Robert Lai, and Dennis Bahler

Department of Computer Science

North Carolina State University

Raleigh, NC 27695-8206

jabowen@adm.csc.ncsu.edu, lai@jim.csc.ncsu.edu, bahler@ncsu.edu

Abstract

We define fuzzy constraint networks and prove a theorem about their relationship to fuzzy logic. Then we introduce Khayyam, a fuzzy constraint-based programming language in which any sentence in the first-order fuzzy predicate calculus is a well-formed constraint statement. Finally, using Khayyam to address an equipment selection application, we illustrate the expressive power of fuzzy constraint-based languages.

1 Introduction

We agree with Zadeh [13] that “the imprecision that is intrinsic in natural languages is, for the most part, possibilistic rather than probabilistic in nature ... the denotation of [an imprecise] word is generally a fuzzy ... subset of a universe of discourse.” We also agree with Nilsson [10] that “For the most versatile machines, the language in which declarative knowledge is represented must be at least as expressive as first-order predicate calculus” (FOPC). So, we are interested in languages which support the expressive power of the full fuzzy FOPC. The inference engines for such languages cannot be sound, complete, and terminating. Although termination is a *sine qua non*, neither of the other two attributes is indispensable. Unsound inference has long been accepted; in crisp logic, it has been codified as various forms of non-monotonic logic. Acceptance of incomplete inference is less widespread, but we argue that there are many situations in which the expressiveness of the full fuzzy FOPC is more important than complete inference.

We have developed a fuzzy constraint-based programming language in which any sentence in fuzzy FOPC about an arbitrary universe of discourse is a well-formed constraint statement. The inference engine for this language, called Khayyam, is sound and

terminating. While not complete in general, it is refutation-complete for many applications.

In Section 2, we introduce semantics in fuzzy logic, with reference to semantics in classical logic. Then, in Section 3, we give the theoretical basis of Khayyam, by presenting the relationship between fuzzy constraint networks and semantic models in fuzzy logic. In Section 4, we introduce Khayyam and give an example program, an expert system for computer selection. We give a comparative review and discussion in Section 5.

2 Semantics in Classical and Fuzzy Logic

A theory Γ in some classical FOPC language $\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$, where \mathcal{P} is the vocabulary of predicate symbols, \mathcal{F} the function symbols, and \mathcal{K} the constant symbols, is satisfiable iff there exists some model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$ of the language \mathcal{L} under which every sentence in Γ is true. In the model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$, \mathcal{U} is a universe of discourse while \mathcal{I} is an interpretation function for the symbols of \mathcal{L} in terms of \mathcal{U} . Using $\mathcal{M}_{\{X \mapsto u\}}$ to denote a model containing the extended interpretation function $\mathcal{I} \cup \{X \mapsto u\}$, the semantic rules for truth are:

$\mathcal{M} \models p(a_1, \dots, a_n)$ iff $\langle \mathcal{I}(a_1), \dots, \mathcal{I}(a_n) \rangle$ is in $\mathcal{I}(p)$.

$\mathcal{M} \models \neg A$ iff $\mathcal{M} \not\models A$.

$\mathcal{M} \models A \wedge B$ iff $\mathcal{M} \models A$ and $\mathcal{M} \models B$.

$\mathcal{M} \models A \vee B$ iff $\mathcal{M} \models A$ or $\mathcal{M} \models B$.

$\mathcal{M} \models A \Rightarrow B$ iff $\mathcal{M} \not\models A$ or $\mathcal{M} \models B$.

$\mathcal{M} \models A \Leftrightarrow B$ iff $(\mathcal{M} \not\models A \text{ and } \mathcal{M} \not\models B)$ or $(\mathcal{M} \models A \text{ and } \mathcal{M} \models B)$.

$\mathcal{M} \models (\forall X)A$ iff $\mathcal{M}_{\{X \mapsto u\}} \models A$ for every $u \in \mathcal{U}$.

$\mathcal{M} \models (\exists X)A$ iff $\mathcal{M}_{\{X \mapsto u\}} \models A$ for some $u \in \mathcal{U}$.

A fuzzy FOPC language contains one species of symbol not found in classical FOPC, hedge symbols. These are used to capture the meaning of linguistic hedges, such as *very* or *fairly*, which are used in sentences of the form $h A$, where h is a hedge and A is a sentence. In a fuzzy language $\mathcal{L} = \langle \mathcal{H}, \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$, \mathcal{H} is the vocabulary of hedge symbols, while \mathcal{P}, \mathcal{F} , and \mathcal{K} are the

predicate, function, and constant symbol vocabularies as in classical logic.

In fuzzy logic, truth can be either multi-valued, with truth-values in the range $[0,1]$, or linguistic, with truth-values denoting fuzzy subsets of $[0,1]$. Here, we consider multi-valued fuzzy logic. As in classical logic, in a fuzzy model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$, \mathcal{U} is a universe of discourse and \mathcal{I} provides interpretations for each symbol in the various vocabularies of \mathcal{L} . As in classical logic, for every $\kappa \in \mathcal{K}$, $\mathcal{I}(\kappa)$ is an element of \mathcal{U} . However, the interpretations for the predicate and function symbols of \mathcal{L} are fuzzy relations over \mathcal{U} . A symbol h in \mathcal{H} modifies the truth value of the sentence to which it is applied, so these hedge symbols are interpreted as crisp one-to-one relations over the segment $[0,1]$ of the real number line. For example, if $\mathcal{I}(h) = \{ \langle X, Y \rangle \mid \text{LEQ}(0, X) \wedge \text{LEQ}(X, 1) \wedge \text{EQUALS}(Y, \text{TIMES}(X, X)) \}$ and the truth-value of a sentence A is d , then the truth of $h A$ is d^2 . A theory Γ in \mathcal{L} is satisfiable iff there exists some fuzzy model $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$ of \mathcal{L} under which every sentence in Γ has a truth-value greater than zero.

Fuzzy functions generalize crisp functions by being able to map onto fuzzy subsets of \mathcal{U} as well as onto individual members. Consider the interpretation for some n -ary function symbol $f \in \mathcal{F}$. In this interpretation, which is a fuzzy $(n+1)$ -ary relation over \mathcal{U} , there may be several tuples that have the same first n components but different $(n+1)^{\text{th}}$ components. Suppose, for example, that the constant symbols a and b are interpreted as 3 and 4, respectively, while the binary function symbol g has as its interpretation the fuzzy set $\{ \langle 1, 5, 6 \rangle @ 1.0, \langle 3, 4, 7 \rangle @ 0.7, \langle 3, 4, 8 \rangle @ 1.0, \langle 3, 4, 9 \rangle @ 0.8 \}$, where $x @ y$ denotes that x has membership degree y in the fuzzy set. In this situation, the functional expression $g(a, b)$ has three possible interpretations: $7 @ 0.7$, $8 @ 1.0$, and $9 @ 0.8$.

Because of this aspect of fuzzy functions, model theory for fuzzy logic is simplified if functional expressions are de-Skolemized. Taking a language $\mathcal{L} = \langle \mathcal{H}, \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$, where \mathcal{F} is non-empty, we produce a function-free language $\mathcal{L}' = \langle \mathcal{H}, \mathcal{P} \cup \mathcal{F}', \emptyset, \mathcal{K} \rangle$, by replacing each n -ary fuzzy function symbol $f \in \mathcal{F}$ by a new $(n+1)$ -ary predicate symbol $f' \in \mathcal{F}'$ which is given the same interpretation as f had. From a theory Γ in \mathcal{L} we generate a theory Γ' in \mathcal{L}' , by replacing each usage of f by an appropriate usage of f' ; for example, the atomic sentence $p(f(a), b)$ would be replaced by $(\exists X)(f'(a, X) \wedge p(X, b))$.

To consider the model-theoretic rules for a function-free fuzzy logic language, let $\mathcal{M} \models^d \gamma$ mean that the sentence γ has the truth-value d under the model \mathcal{M} . Then, the model-theoretic rules, where p is a predicate symbol, h is a hedge symbol, and the κ_i are constant

symbols, are as follows:

$$\begin{aligned} \mathcal{M} \models^d p(\kappa_1, \dots, \kappa_n) &\text{ iff } \langle \mathcal{I}(\kappa_1), \dots, \mathcal{I}(\kappa_n) \rangle \text{ has} \\ &\text{ membership degree } d \text{ in } \mathcal{I}(p). \\ \mathcal{M} \models^d hA &\text{ iff } \mathcal{M} \models^e A \text{ and } \langle e, d \rangle \text{ is in } \mathcal{I}(h). \\ \mathcal{M} \models^d \neg A &\text{ iff } \mathcal{M} \models^{1-d} A. \\ \mathcal{M} \models^d A \wedge B &\text{ iff } \mathcal{M} \models^a A \text{ and } \mathcal{M} \models^b B \\ &\text{ and } d \text{ is } \min(\{a, b\}). \\ \mathcal{M} \models^d A \vee B &\text{ iff } \mathcal{M} \models^a A \text{ and } \mathcal{M} \models^b B \\ &\text{ and } d \text{ is } \max(\{a, b\}). \\ \mathcal{M} \models^d A \Rightarrow B &\text{ iff } \mathcal{M} \models^a A \text{ and } \mathcal{M} \models^b B \\ &\text{ and } d \text{ is } \min(\{1, 1 - a + b\}). \\ \mathcal{M} \models^d A \Leftrightarrow B &\text{ iff } \mathcal{M} \models^a A \text{ and } \mathcal{M} \models^b B \\ &\text{ and } d \text{ is } \min(\{1, 1 - a + b, 1 - b + a\}). \\ \mathcal{M} \models^d (\forall X)A, &\text{ where } X \text{ is free in } A, \text{ iff} \\ &d \text{ is } \min(\{e \mid u \in \mathcal{U} \wedge \mathcal{M}_{\{X \mapsto u\}} \models^e A\}). \\ \mathcal{M} \models^d (\exists X)A, &\text{ where } X \text{ is free in } A, \text{ iff} \\ &d \text{ is } \max(\{e \mid u \in \mathcal{U} \wedge \mathcal{M}_{\{X \mapsto u\}} \models^e A\}). \end{aligned}$$

3 Fuzzy Constraint Networks

Formally, a fuzzy constraint network can be defined as:

Fuzzy Constraint Network:

A fuzzy constraint network is a triple $\langle \mathcal{U}, \mathbf{X}, \mathbf{C} \rangle$ where \mathcal{U} is a universe of discourse, \mathbf{X} is a finite tuple of q parameters, and \mathbf{C} is a finite set of r fuzzy constraints. Each constraint $C_k(T_k) \in \mathbf{C}$ imposes a restriction on the allowable values for the a_k parameters in T_k , a sub-tuple of \mathbf{X} , by specifying that some fuzzy subset of \mathcal{U}^{a_k} contains all acceptable combinations of values for the parameters. \square

A fuzzy constraint network is an intensional specification of a joint fuzzy possibility distribution for the values of the parameters in the network:

The Intent of a Fuzzy Constraint Network:

The intent of a fuzzy constraint network $\langle \mathcal{U}, \mathbf{X}, \mathbf{C} \rangle$ is $\Pi_{\mathcal{U}, \mathbf{X}, \mathbf{C}} = \overline{C}_1(\mathbf{X}) \cap \dots \cap \overline{C}_r(\mathbf{X})$, where, for each $C_k(T_k) \in \mathbf{C}$, $\overline{C}_k(\mathbf{X})$ is its cylindrical extension in the Cartesian space defined by \mathbf{X} . \square

The network intent is a fuzzy set of q -tuples, each tuple giving a valuation for the q parameters in \mathbf{X} , the membership of the tuple in the intent being the degree to which the valuation satisfies all the constraints in \mathbf{C} . A fuzzy constraint network is consistent iff the network intent is not the empty set.

It can be shown that finding consistent values for the parameters in a fuzzy constraint network is the same as finding a model of a fuzzy FOPC language under which some theory is satisfied. In what follows, let $\vec{A}\vec{B}$, where A and B are tuples of the same arity, be the mapping function from the components of A onto those of B in which each component of the tuple A is mapped onto the component in the corresponding position of tuple B ; for example, if $A = \langle x, y \rangle$ and $B = \langle 4, 2 \rangle$, then $\vec{A}\vec{B} = \{ x \mapsto 4, y \mapsto 2 \}$.

Theorem 1

Given $\mathcal{L} = \langle \mathcal{H}, \mathcal{P}, \mathcal{F}, \mathcal{K}' \cup \mathcal{K}'' \rangle$, \mathcal{U} , \mathcal{I}_p , and Γ , where \mathcal{I}_p interprets, in terms of \mathcal{U} , all and only the symbols in $\mathcal{H} \cup \mathcal{P} \cup \mathcal{F} \cup \mathcal{K}'$, \mathcal{K}'' is finite, each sentence in Γ references at least one symbol in \mathcal{K}'' , and each symbol in \mathcal{K}'' is referenced by at least one sentence $\gamma \in \Gamma$. Let $\langle \mathcal{U}, \mathbf{X}, \mathbf{C} \rangle$ be a fuzzy constraint network such that \mathbf{X} is the lexical ordering of the elements of \mathcal{K}'' and such that $|\mathbf{C}| = |\Gamma|$, with each sentence $\gamma \in \Gamma$ having a corresponding constraint $C(T) \in \mathbf{C}$ such that T is the lexical ordering of those elements of \mathcal{K}'' which appear in γ and $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{T}t \rangle \models^d \gamma$ for each tuple t having membership of degree d in $C(T)$. Then $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^e \Gamma$ for each tuple τ having membership degree e in $\Pi_{\mathcal{U}, \mathbf{X}, \mathbf{C}}$. \square

That is, the set of all $\vec{\mathbf{X}}\tau$ where τ has membership degree d in $\Pi_{\mathcal{U}, \mathbf{X}, \mathbf{C}}$ is the set of all \mathcal{J} such that \mathcal{J} interprets all symbols in \mathcal{K}'' and $\langle \mathcal{U}, \mathcal{I}_p \cup \mathcal{J} \rangle \models^d \Gamma$. Thus, finding consistent values for the parameters in a fuzzy constraint network $\langle \mathcal{U}, \mathbf{X}, \mathbf{C} \rangle$ may be regarded as semantic modeling in a fuzzy logic language $\mathcal{L} = \langle \mathcal{H}, \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$.

The proof of this theorem depends on the following lemma.

Lemma

Given $\mathcal{L} = \langle \mathcal{H}, \mathcal{P}, \mathcal{F}, \mathcal{K}' \cup \mathcal{K}'' \rangle$, \mathcal{U} , \mathcal{I}_p , and γ , where \mathcal{I}_p interprets, in terms of \mathcal{U} , all and only the symbols in $\mathcal{H} \cup \mathcal{P} \cup \mathcal{F} \cup \mathcal{K}'$, \mathcal{K}'' is finite, and the sentence γ references at least one symbol in \mathcal{K}'' . Let \mathbf{X} be the lexical ordering of the elements of \mathcal{K}'' and let $C(T)$ be a constraint such that T is the lexical ordering of those elements of \mathcal{K}'' which appear in γ and such that $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{T}t \rangle \models^d \gamma$ for each tuple t having membership degree d in $C(T)$. Then $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^d \gamma$ for each tuple τ having membership degree d in $\vec{C}(\mathbf{X})$ where $\vec{C}(\mathbf{X})$ is the cylindrical extension of $C(T)$ into the space defined by \mathbf{X} . \square

Proof of Lemma

Take any tuple τ having membership degree d in $\vec{C}(\mathbf{X})$. The projection of the singleton set $\{\tau\}$ on T will be a singleton set containing some tuple τ' ; that is, $proj(\{\tau\}, T) = \{\tau'\}$. Then, by the definition of cylindrical extension, τ' has membership degree d in $C(T)$. The mapping function $\vec{\mathbf{X}}\tau$ provides a mapping for each constant symbol in \mathcal{K}'' . However, the truth of γ depends only on the interpretation of those members of \mathcal{K}'' which appear in γ . Thus, the truth of γ depends only on the mappings provided by $\vec{T}\tau'$, which is a subset of $\vec{\mathbf{X}}\tau$. Since τ' has membership degree d in $C(T)$, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{T}\tau' \rangle \models^d \gamma$, and hence $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^d \gamma$, because the additional mappings in $(\vec{\mathbf{X}}\tau - \vec{T}\tau')$ have no impact on the truth of γ . Since τ is any tuple in $\vec{C}(\mathbf{X})$, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^d \gamma$ for any tuple τ having membership degree d in $\vec{C}(\mathbf{X})$. \square

Proof of Theorem 1

Since $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{T}_1 t \rangle \models^a \gamma_1$ for each tuple t with membership degree a in $C_1(T_1)$ then, by the lemma, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^a \gamma_1$ for each tuple τ with membership degree a in $\vec{C}_1(\mathbf{X})$. Similarly, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^b \gamma_2$ for each tuple τ with membership degree b in $\vec{C}_2(\mathbf{X})$. Therefore, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^d \{\gamma_1, \gamma_2\}$ for each tuple τ with membership degree d in $\vec{C}_1(\mathbf{X}) \cap \vec{C}_2(\mathbf{X})$, since the membership degree d of τ in $\vec{C}_1(\mathbf{X}) \cap \vec{C}_2(\mathbf{X})$ is the minimum $\min(\{a, b\})$ of its membership degrees a and b in $\vec{C}_1(\mathbf{X})$ and $\vec{C}_2(\mathbf{X})$, respectively. Similarly, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^d \{\gamma_1, \gamma_2, \dots, \gamma_r\}$ for each tuple τ with membership degree d in $\vec{C}_1(\mathbf{X}) \cap \vec{C}_2(\mathbf{X}) \cap \dots \cap \vec{C}_r(\mathbf{X})$. That is, $\langle \mathcal{U}, \mathcal{I}_p \cup \vec{\mathbf{X}}\tau \rangle \models^d \Gamma$ for each tuple τ with membership degree d in $\Pi_{\mathcal{U}, \mathbf{X}, \mathbf{C}}$. \square

4 The Khayyam language

Khayyam is a programming language based on the the relationship, discussed above, between fuzzy constraint networks and possibility distributions for semantic models of fuzzy logic theories. A program in Khayyam provides a declarative specification of a constraint network by specifying a fuzzy language $\mathcal{L} = \langle \mathcal{H}, \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$, a theory Γ written in \mathcal{L} , a universe of discourse \mathcal{U} , and a partial interpretation function \mathcal{I}_p for \mathcal{L} . Of these, only Γ must always be specified explicitly. The Khayyam run-time system provides a default language $\mathcal{L}_g = \langle \mathcal{H}_g, \mathcal{P}_g, \mathcal{F}_g, \mathcal{K}_g \rangle$ in which \mathcal{K}_g contains the rational decimal numeric strings, \mathcal{P}_g contains names of standard predicates ($=$, $=<$, etc.), \mathcal{F}_g contains names of standard functions ($*$, $+$, etc.), and \mathcal{H}_g contains names of standard hedges (*very*, *fairly*, etc.). The run-time system also provides a default universe of discourse $\mathcal{U}_g = \mathfrak{R}$ and a total interpretation function \mathcal{I}_g for \mathcal{L}_g . This interpretation includes a bijection from the numeric strings \mathcal{K}_g onto the finitely expressible rationals $Q_f \subset Q \subset \mathfrak{R}$, providing the obvious mappings, such as $1.5 \rightarrow 1.5$; note the distinction between 1.5, which is a symbol in \mathcal{K}_g and 1.5 which is a real number in \mathfrak{R} .

The Khayyam run-time system computes the marginal possibility distribution [13] for each parameter in the network defined by a program. To do so, it transforms the language \mathcal{L} defined by a program into an internal function-free language \mathcal{L}' and de-Skolemizes all function-referring sentences in Γ to produce Γ' . The result is subjected to an inference algorithm called Fuzzy Compound Propagation (FCP). Although full details of the algorithm are beyond the scope of this paper, FCP involves interleaved application of three inference techniques: LP, local propagation of known states [7]; FAC, a fuzzy version of

arc consistency [8], generalized to infinite domains and constraints of arbitrary arity, and based on particularization and projection [13]; GPC, a form of path consistency [8], generalized to infinite domains and constraints of arbitrary arity, which is only applied to small portions of the network in certain very specific circumstances.

The algorithm is sound, in that, at any time, the real intensional marginal possibility distribution for a parameter is contained within whatever distribution the algorithm computes for that parameter. If the algorithm were complete, the intensional distribution for a parameter would be exactly that computed. But the algorithm is not guaranteed to be complete, so the intensional distribution for a parameter may sometimes be a proper fuzzy subset of the computed distribution. However, given the meaning of the term “possibility,” a computed distribution which subsumes the intensional is still correct; we merely lack full knowledge of the intensional possibility distribution function.

Khayyam programs are interactive. The user is allowed to augment the theory defined in a program with additional sentences and, whenever he does so, the Khayyam run-time system uses FCP to compute revised marginal possibility distributions. Whenever the intent of the network that results from a sequence of user assertions is the empty set, the system reports a constraint violation. If this happens, the user can recover by retracting one of his earlier assertions. The system maintains a set of dependency records and, at any time, the user can request a justification for the current marginal possibility distribution of a parameter in the network. By making assertions and retractions in any order he pleases, the user can explore various “what if” scenarios.

4.1 A Fuzzy Expert System

The following program defines a fuzzy language $\mathcal{L} = \langle \mathcal{H}_g, \mathcal{P}, \mathcal{F}, \mathcal{K} \rangle$ in which $\mathcal{P} = \mathcal{P}_g \cup \{big, cheap, fast\}$, $\mathcal{F} = \mathcal{F}_g \cup \{price, slots, space, speed, s_func\}$, and $\mathcal{K} = \mathcal{K}_g \cup \{chosen_model, reqd_disk, reqd_slots, reqd_speed\}$.

```

/*Function and Predicate Definitions*/
function speed ::= datafile(qdb,$SPEED$).
function price ::= datafile(qdb,$PRICES$).
function space ::= datafile(qdb,$DISKCPTY$).
function slots ::= datafile(qdb,$NUMSLOTS$).
function s_func ::=
  {(X,A,B,G) -> 0 : X < A} union
  {(X,A,B,G) -> Y : A =< X < B and
    Y = 2*((X-A)/(G-A)) ^ 2} union
  {(X,A,B,G) -> Y : B =< X < G and
    Y = 1-2*((X-G)/(G-A)) ^ 2} union
  {(X,A,B,G) -> 1 : G =< X}.
predicate big ::= {X@Y: Y = s_func(X,100,150,200)}.
predicate fast ::= {X@Y: Y = s_func(X,16,20,24)}.
predicate cheap ::=
  {X@Y: Y = 1 - s_func(X,1000,2000,3000)}.

/*Theory*/
space(chosen_model) >= reqd_disk.
slots(chosen_model) >= reqd_slots.
speed(chosen_model) >= reqd_speed.
not(exists X : space(X) >= reqd_disk and
  slots(X) >= reqd_slots and
  speed(X) >= reqd_speed and
  price(X) < price(chosen_model)).

```

The program specifies the following theory Γ written in the language \mathcal{L} : $\{slots(chosen_model) \geq reqd_slots, speed(chosen_model) \geq reqd_speed, space(chosen_model) \geq reqd_disk, \neg(\exists X)(slots(X) \geq reqd_slots \wedge speed(X) \geq reqd_speed \wedge space(X) \geq reqd_disk \wedge price(X) < price(chosen_model))\}$.

This program is an expert system which interacts with a user and selects a computer to provide, at a minimum price, whatever functionality the user specifies. The user inputs his specification by augmenting the theory Γ given in the program with additional assertions about the symbols *reqd_disk* (representing the amount of disk space he needs), *reqd_slots* (representing the number of slots he needs for attaching peripheral devices), and *reqd_speed* (representing the CPU speed he requires). The program reads the external files PRICES, DISKCPTY, NUMSLOTS, and SPEED to get the extensions for the function symbols *price*, *space*, *slots*, and *speed*. Thus, when selecting a computer, the program chooses from among those machines whose characteristics are described in the external database files.

The program defines a constraint network whose intent is a joint possibility distribution for the values of *chosen_model*, *reqd_disk*, *reqd_slots*, and *reqd_speed*.

Suppose that the contents of the external files PRICES, DISKCPTY, NUMSLOTS, and SPEED are

as shown in Table 1. Then the joint possibility distribution is the relation shown in Table 2.

Table 1

model1	4000	model1	628
model2	3500	model2	314
model3	2500	model3	192
model4	1500	model4	96
model5	900	model5	40
model6	500	model6	10

(a) PRICES

(b) DISKCPTY

model1	7
model2	7
model3	5
model4	3
model5	3
model6	3

(c) NUMSLOTS

model1	25
model2	23
model3	19
model4	16
model5	12
model6	12

(d) SPEED

A user interacting with this program can specify his requirements, either crisply, or fuzzily, using one of the fuzzy predicates, *big*, *fast* or *cheap*, defined in the program. For example, in specifying speed, he could be crisp, as in $reqd_speed > 18.7$, or imprecise, as in $very\ fast(reqd_speed)$, which uses a predefined hedge *very* and one of the application-specific fuzzy predicates whose interpretations are defined intensionally in the program. Alternatively, in expressing a budgetary limitation, he could be crisp, as in $price(chosen_model) \leq 2000$, or fuzzy, as in $cheap(price(chosen_model))$. Note that the fuzzy predicates are defined using a crisp application-specific function s_func , the definition of which shows it to be the standard piecewise quadratic function found in the literature [13].

Suppose, for example, that the user asserts

$$fast(reqd_speed).$$

The definition of the fuzzy predicate *fast* shows that any speed equal to or less than 16 MHz is definitely not fast. Thus, model4, model5, and model6 are not possible values for *chosen_model* and the corresponding tuples are removed from the resultant joint possibility distribution.

Table 2

<i>chosen_model</i>	<i>reqd_disk</i>	<i>reqd_slots</i>	<i>reqd_speed</i>
model1	{D 314 < D ≤ 628}	{L L ≤ 7}	{P P ≤ 25}
model1	{D D ≤ 628}	{L L ≤ 7}	{P 23 < P ≤ 25}
model2	{D 192 < D ≤ 314}	{L L ≤ 7}	{P P ≤ 23}
model2	{D D ≤ 314}	{L 5 < L ≤ 7}	{P P ≤ 23}
model2	{D D ≤ 192}	{L L ≤ 7}	{P 19 < P ≤ 23}
model3	{D 96 < D ≤ 192}	{L L ≤ 5}	{P P ≤ 19}
model3	{D D ≤ 192}	{L 3 < L ≤ 5}	{P P ≤ 19}
model3	{D D ≤ 192}	{L L ≤ 5}	{P 16 < P ≤ 19}
model4	{D 40 < D ≤ 96}	{L L ≤ 3}	{P P ≤ 16}
model4	{D D ≤ 96}	{L L ≤ 3}	{P 12 < P ≤ 16}
model5	{D 10 < D ≤ 40}	{L L ≤ 3}	{P P ≤ 12}
model6	{D D ≤ 10}	{L L ≤ 3}	{P P ≤ 12}

The run-time system will compute and report the marginal possibility distributions for the network parameters *chosen_model*, *reqd_disk*, *reqd_slots*, and *reqd_speed* that result from intersecting the joint possibility distribution specified by the program with the cylindrical extensions of the distributions specified by the user's assertions.

Suppose that the user also asserts

$$cheap(price(chosen_model)).$$

The definition of the fuzzy predicate *cheap* indicates that any price equal to or greater than \$3000 is definitely not cheap; thus, model1 and model2 are removed from the joint possibility distribution. The only computer still possible is the model3 which only partially meets the specification: the truth-value is 0.125, which is the minimum of 0.28 (the extent to which 19 MHz is *fast*) and 0.125 (the extent to which \$2500 is *cheap*).

If the user is content with this level of truth, he can accept the model3. Alternatively, he can try other scenarios by asserting and retracting different specifications. All inferences can be explained by the run-time system.

5 Comparative Discussion

The notion of a constraint permeates the fuzzy logic literature. Indeed, in fuzzy logic, "inference is viewed as a process of propagation of elastic constraints" [14]. In matrix-based fuzzy production systems [12], the matrix represents the intent of a constraint network, albeit of a restricted kind. In matrix-based fuzzy process controllers, for example, the matrix is a joint possibility distribution for the sensor readings and control settings. The large and growing literature on fuzzy mathematical programming can also be viewed as concerned with constraint processing. There is also a smaller body of literature, see for example [11], on consistent labeling, which involves networks where the universe of discourse is finite and the constraints correspond to ground atomic sentences involving unary or binary predicates.

Khayyam provides richer expressive power than any other constraint-based programming language (all of which handle only crisp constraints) because it allows the theory which is used to specify a constraint network to contain arbitrary first-order fuzzy logic sentences about a many-sorted universe of discourse that subsumes \mathfrak{R} . Most constraint-based programming languages restrict the theory used for defining a network to ground sentences. In the crisp Prolog-based constraint languages (Prolog III [4], CLP(\mathfrak{R}) [6] and CHIP [5]), for example, a constraint network is treated as a compound top-level goal; all logic variables in a goal

are existentially quantified, which makes them effectively, for query purposes, the same as uninterpreted constant symbols. CONSUL [3], the only other constraint language which does support arbitrarily quantified constraints, restricts the universe of discourse to the integers.

Although the expressive power of Khayyam means that well-formed programs can be beyond the inferential competence of the run-time system, this is true even of less expressive languages; for example, it is possible to express undecidable diophantine equations in CONSUL.

Khayyam differs from languages based on a fuzzy version of Prolog, such as Fril [1] and FProlog [9], in that these languages are based on the Horn clause subset of fuzzy logic. Khayyam, on the other hand, admits as well-formed syntactically, arbitrary sentences from the first-order fuzzy logic; this is based on a decision to trade inferential completeness for expressiveness. For further details on Khayyam, see [2].

References

- [1] Baldwin J, and Martin T, 1992, "Fast Operations on Fuzzy Sets in the Abstract Fril Machine," *Proc. IEEE Intl. Conf. on Fuzzy Systems*, 803-810.
- [2] Bowen J, Lai R and Bahler D, 1992, "Fuzzy Semantics and Fuzzy Constraint Networks," *Proc. IEEE Intl. Conf. on Fuzzy Systems*, 1009-1016.
- [3] Chronaki C and Baldwin D, 1990, "A Front End for CONSUL," Tech. Report CS-TR-319, Dept. of Computer Science, Univ. of Rochester.
- [4] Colmerauer A, 1987, "An Introduction to Prolog III," Tech. Report, Univ. Aix-Marseille II.
- [5] Dincbas M, van Hentenryck P, Simonis H, Aggoun A, Graf T and Berthier F, 1988, "The Constraint Logic Programming Language CHIP," *Proc. Fifth Gen. Comp. Sys. '88*.
- [6] Jaffar J and Michaylov S, 1986, "Methodology and Implementation of a CLP System," *Proc. ICLP-4*.
- [7] Leler W, 1988, *Constraint Programming Languages*, Reading, MA: Addison-Wesley.
- [8] Mackworth A, 1977, "Consistency in Networks of Relations," *Artif. Intel.*, 8, 99-118.
- [9] Martin T, Baldwin J and Pilsworth P, 1987, "The implementation of FProlog - A fuzzy Prolog interpreter," *Fuzzy Sets and Systems*, 23, 119-129.
- [10] Nilsson N, 1991, "Logic and artificial intelligence," *Artif. Intel.*, 47, 31-56.
- [11] Snow P and Freuder E, 1990, "Improved Relaxation and Search Methods for Approximate Constraint Satisfaction with a Maximin Criterion," *Proc. CSCSI-8*.
- [12] Whalen T, and Schott B, 1983, "Issues in Fuzzy Production Systems," *Int. J. Man-Machine Stud.*, 19, 57-71.
- [13] Zadeh L A, 1978, "PRUF - A meaning representation language for natural languages," *Int. J. Man-Machine Studies*, 10, 395-460.
- [14] Zadeh L, 1989, "Knowledge Representation in Fuzzy Logic," *IEEE Trans. on Knowledge and Data Engineering*, 1(1), 89-100.