Expressing Constraint Satisfaction Problems in Declarative Modeling Using Natural Language and Fuzzy Sets

Emmanuel Desmontils

PERGAMON

Computers & Graphics 24 (2000) 555–568

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Technical Section

Expressing constraint satisfaction problems in declarative modeling using natural language and fuzzy sets

Emmanuel Desmontils

IRIN, Université de Nantes, 2 Rue de la Houssinière, BP 92208, F-44322 Nantes Cedex 3, France

Abstract

The aim of this paper is to provide a new model of fuzzy or vague properties according to linguistic notions within the scope of declarative modeling and constraint satisfaction problems (CSPs). Some generic modifiers and generic fuzzy operators are applied on these properties. This model allows us to build a CSP according a pseudo-natural scene description. Solving this CSP provides possible scenes. Moreover, we propose a linguistic interpretation of negative properties instead of the classical logical negation. This negation process selects plausible properties as linguistic negation. Then, it sorts them to provide first the ones that have the best chance to be selected by the user.

Keywords: Image synthesis; Geometric modeling; Declarative modeling; Constraints; CSP; Fuzzy sets; Linguistics

1. Introduction

The aim of Declarative Modeling in image synthesis [1–3], also called Generative Computer Aided Design [4], Cooperative Computer Aided Design [5], Interactive 3-D Computer Graphics [6] or Scene Description [7], is to provide high-level tools to help a designer to design a scene.[1] Declarative Modeling allows the designer to create his scene just giving a set of properties. The modeler is in charge of exploring the universe of scenes to select the ones that respect the description. The designer has only to choose his favorite scene using appropriate tools. Therefore, the building process is composed of three steps:

- a high level *scene description*, with the appropriate media, without considering any basic geometric primitive or any elementary value;
- an exhaustive *scene generation*;
- an insight of the solution set using high level *understanding tools*, i.e. based not only on the knowledge of the scene solution but also on properties of the description and properties deduced during the generation.

If the designer does not find a solution that suits him, he can change the description and generate a new set of solutions. Consequently, the work done by a declarative modeler is more important than the one done by a traditional one. So, the designer can restrict his work to higher-level tasks. The aim of Declarative Modeling is not necessarily to provide directly the designer the solution he/she wants but to help him/her in the design process. Solutions the designer study help him/her to refine his/her mental representation of the scene. Therefore, Declarative Modeling enriches the design process by proposing solutions the designer did not plan. These solutions can be considered as drafts of the final one. Finally, he/she can use a classical modeler to adjust a good solution. Ref. [3] gives precisions about possible working modes with a Declarative Modeler.

The scene description process is based on several types of media. The most current one is the classical computer dialog using dialog boxes, text fields, radio or check buttons, etc. But a description can also be provided to the

E-mail address: emmanuel.desmontils@irin.univ-nantes.fr (E. Desmontils).

[1] In declarative modeling, the word "scene" is commonly used as "geometric model of the scene" without making assumption about the hardware configuration, the viewpoint, etc.

0097-8493/00/$ - see front matter
PII: S0097-8493(00)00058-3

modeler through the use of natural language (writing or talking), sketches, gestures, etc.

The scene generation process is more or less declarative. It can be:

- a reverse function [4,8,9];
- a database exploration [6,10,11];
- an exploration tree in "GENWIN" [12] and "MegaFormes" [13];
- an inference engine using rules and facts in "PolyFormes" [14];
- a constraint based system using:
  - Snyder's interval analysis [15] in "VoluFormes" [16] and "SKETCH" [17],
  - interval constraint programming [18] in "Virtual Cameraman" [19],
  - Allen's relations on temporal intervals [20] in an architecture project [21],
  - constraint satisfaction techniques [22] in "BatiMan" [23].

The insight of the solutions concerns as well one solution (calculating the best view point in a 3D scene, the best visualization model, etc.) as the set of solutions itself (high-level classification tools, etc.).

In such systems, the main notion is the description. It allows the designer to describe the desired scene through high-level properties without considering any building detail (geometric primitives, precise values or intervals, etc.). Therefore, the description is considered as a set of properties describing the main outlines or features of a scene. The description is used by the generation process to build the corresponding solutions and by understanding tools to highlight solutions.

The formalization of properties used in building a description is fundamental for a powerful interpretation. In common applications (based on imperative modeling), the scene is often described by giving or selecting values using menus, dialogs, etc.

Example 1. For instance, let us suppose that the designer wishes to create a rectangular zone with "a rather important front width and a weak surface". With a classical modeler the designer has to use the function "setRectangle (x, y, w, h)" and to value all parameters: "setRectangle $(10, 20, 30, 25)$" even if the position of this zone does not matter. Moreover he/she must use a calculator to validate this solution. If this solution does not suit him/her, the designer has to "play" with parameters and the calculator to find a good one.

But this type of man–computer dialogue is not always understandable [24]. To describe a scene, words and sentences are sometimes the only way to express what the designer wants to see. The major reason is the vagueness of natural languages.

Example 2. With a declarative modeler, the designer says "The surface is weak. The front width is rather important" and the modeler generates a set of valid solutions by automatically computing values for parameters.[2] The designer has only to select the one he/she prefers.

But, in common Declarative Modelers [16,25], properties associated with each modified words are often classical disjoint intervals. For us, this model (which is not based on linguistic studies) does not correspond to the human thinking and to the natural language. For example, an "important width" is also a "rather important width" but not at the same level. Moreover, it seems to us that it would be easier to declare a few basic properties and to build the other with generic modifiers. Modifier parameters could be adapted to the user's profile.

In this paper, properties are defined with *vague or imprecise linguistic terms* expressed in pseudo-natural language sentences. Therefore, we propose a new model using basic properties modified by generic operators and deals with imprecise or vague properties based on current formalizations [7,16,25] and Zadeh's fuzzy sets [26]. This model allows us to improve the man–computer dialogue using qualitative and quantitative properties formulated with a more natural language. It also allows us to manage negative properties using linguistic studies instead of the logical negation usually used in Fuzzy Logic and Declarative Modeling. This process will try to find an implicit affirmative property equivalent to the negative one.

To propose such a model, a proper classification of properties is needed to make an inventory of all possible properties. We can find some semantic taxonomies (ontology-based approach[3]) on particular fields of knowledge, like taxonomies of spatial relations [6,16,21,27], but not for all ones used in scene description. In [28], we suggest a syntactic classification of these properties in Declarative Modeling considering semantics of properties as an application dependent problem. The major ones are: *predicate properties*, *relative properties* and *connected properties*.

The aim of this paper is to present this model and the negation's management within the scope of a constraint satisfaction problem solver for the scene generation step. So, we successively present in this paper a short introduction to constraint satisfaction problems (Section 2) and,

[2] This very simple "rectangular zone declarative modeler" will be called the "RZ-modeler" in future examples.

[3] The word ontology means the study of existence in philosophy, but in artificial intelligence it usually means the set of most primitive terms or concepts that canonically describe some particular class of concepts like geometric shape, topology, visual sensations, etc.

an overview of basic notions of our model and the first properties we studied called elementary properties (Section 3). Then, we propose new models of properties we take into account: relative properties (Section 3) and connected properties (Section 4). We also present our solution to manage negative properties and new linguistic elements to improve this process (Section 5). Finally, before concluding, we describe a brief application (Section 6).

2. Constraint satisfaction problems

Constraint Satisfaction Techniques have been used several times in computer graphics, especially in geometric modeling [29]. A finite Constraint Satisfaction Problem (CSP) is defined by $P = (X, D, C)$, where $X = \{X_1, \ldots, X_n\}$ is a set of variables, $D = \{D_1, \ldots, D_n\}$ a set of finite domains associated with variables and $C = \{C_1, \ldots, C_m\}$ a set of constraints [22]. A constraint can be:

- a set of tuples of domain's value witch are authorized for some variables;
- a function witch associates each authorized value of domains.

Solving a CSP means to find value assignments to variables subject to constraints. CSP solvers are already used in Declarative Modeling for the scene generation step [23]. The aim of this paper is not to study CSP and solvers. Conversely, we propose a model that allows the modeler to build the CSP according to different properties of the description, that is to create: the set of variables, the set of associated domains and the set of constraints. Then we study the syntactic shape of possible properties and the process to deduce variables, domains and constraints. This CSP is a theoretical model and does not imply a specific CSP solver. However, this solver has to be able to provide all solutions. For instance, to provide our final example (Section 6) we used the Java Constraint Library.[4] Now, we will study each type of properties and the associated CSP influence.

3. E-properties and basic notions

3.1. Definition

Among all types of properties, the simplest property of a description is called *elementary property* (*e-property*) (also called *predicate properties* in [28]). This property is built as "C_i of x is $f_\alpha m_\beta P_{ik}$" where C_i is *a concept*, x an object that the property deals with, f_α a *fuzzy operator*, m_β a *fuzzy modifier* and P_{ik} a *basic property*.

Example 3. "The surface of c is more or less weak. The front width of c is rather important" are e-properties. "Surface" and "front" are concepts. "*c*" is the object the designer describe. "Rather" is a fuzzy modifier. "More or less" is a fuzzy operator. "Weak" and "important" are basic properties.

Basic notions of concept, operators and e-properties are studied in [30,31]. We are going to present a short overview of them.

3.2. Concepts, domains and basic properties

A concept is a main outline, a feature of a scene. Note that in the scene generation step, concepts are not used in the same way. Some of them are used to build the scene. We call them *generating concepts*. The others are called *non-generating concepts*. They are redundant with one or more generating concepts. However, the designer can describe them.

A *description domain* (*a domain*) is the set of all numerical or symbolic values of the concept. A domain is associated with each concept C_i and is denoted D_i. D_i is often a finite set denoted $\{V_1, V_2, \ldots, V_n\}$ where V_i is a numeric or a symbolic value. For numerical values, D_i is usually defined as an interval of values denoted $[Min \ldots Max]_{unit}$.[5]

For a given scene, the current domain's value of a concept C_i (often a non-generating one) can be calculated by a concept dependent function m_i called the *measure*.

Example 4. For instance, let us take our RZ-modeler. Three concepts can be used: the "front width" (with a domain $[10..40]_1$), the "depth" (with a domain $[20..60]_1$) and the "surface" (with a domain $[200..2400]_1$). The "surface" is a non-generating concept.[6] It can be calculated using the two generating concepts "front width" and "depth". So, its measure is:

$$m_{Surface} = FrontWidth * Depth.$$

A set P_i of basic properties is also associated with each concept C_i. A basic property P_{ik} in P_i is a fuzzy subset of D_i. Then P_{ik} is defined with a *membership function* that gives the *membership degree* to P_{ik} for each value of D_i. If

[4] The Java Constraint Library (JCL) is a Java library that can be used as well in Java enabled browsers as in standalone Java applications. JCL was developed in the Artificial Intelligence laboratory at the Swiss Federal Institute of Technology in Lausanne by E. Bruchez & M. Torrens in 1996.

[5] This notation is equivalent to: $\{Min, Min + unit, Min + 2*unit, \ldots, Max\}$

[6] This generating/non-generating classification can be different. One can select "surface" and "depth" as generating concepts and "front width" as non-generating concept, etc.

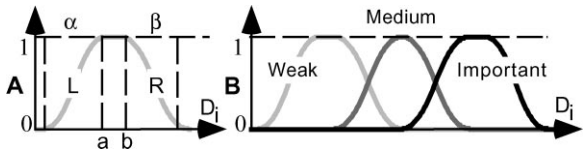

Fig. 1. Basic properties of a concept.

the property is a fuzzy interval, its function can be a LR-function (Fig. 1A) which is defined by two functions $R(x)$ and $L(x)$ (linear functions to make "trapezoidal LR-function" [32]) and a quadruplet $\langle \alpha, a, b, \beta \rangle$ where $[a..b]$ is the set of all values with a maximum membership degree (called *the kernel*) and $[a-\alpha..b+\beta]$ the set of all values with a membership degree greater than 0 (called *the support set*). Properties of Fig. 1B are also defined by LR-functions.

P_i is often composed of three basic properties according to the standard model {*"weak"*, *"medium"*, *"important"*} (Fig. 1B).

From a linguistic point of view, this concept notion can be associated to the *semantic category* notion [33–36]. Semantic categories (concepts) are based on a *maximal category* composed of six *semantic units* (basic properties):

- two unconnected terms, I (*positive*) and F (*negative*), associated to two separate qualities;
- one *neutral* term M giving the absence of I and F (this category is not applied);
- one complex term C, which covers I and F, only indicating that the category is applied;
- two terms IM and FM equivalent to C but that emphasize either I or F.

All categories are built as subsets of the maximal one. The only constraint concerns a symmetry for the signed elements of a category (I, F, IM and IF). This study linguistically validates a concept with three basic properties in the P_i set. In this paper, we use the three basic properties (Fig. 1): "*weak*" (negative), "*medium*" (neutral) and "*important*" (positive).

Basic properties of a concept are modified by fuzzy modifiers and fuzzy operators. These two types of operators are generic: they are not defined for a particular concept or the basic property they are applied on.

3.3. Operators

Fuzzy modifiers are numerous. We can select a finite set with a global order relation denoted: $M_P = \{m_\gamma \mid \gamma \in [1..P]\}$ with $m_\alpha \leqslant m_\beta \Leftrightarrow \alpha \leqslant \beta$. For instance, we can select this following set of modifiers $M_7 = \{$*"extremely little"*, *"very little"*, *"rather little"*, $\emptyset$, *"rather"*, *"very"*, *"extremely"*$\}$.[7] A fuzzy modifier m_α is signed, i.e. $\text{sign}(m_\alpha) < 0$ if $m_\alpha <$ *"*$\emptyset$*"* else $\text{sign}(m_\alpha) > 0$. They transform the membership function of P_{ik} by using a translation and a contraction function. Fig. 2A shows modifications that occur applying all fuzzy modifiers of M_7 on the basic property "*important*".

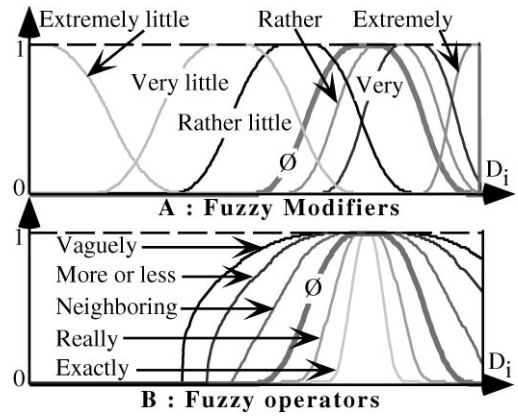

Fig. 2. Sets of (A) fuzzy modifiers and (B) fuzzy operators

In the same way, as fuzzy operators are also numerous, we can select a finite set with a global order relation denoted: $F_Q = \{f_\gamma \mid \gamma \in [1..Q]\}$ with $f_\alpha \leqslant f_\beta \Leftrightarrow \alpha \leqslant \beta$. For instance, we can select this following set of operators $F_6 = \{$*"exactly"*, *"really"*, *"*$\emptyset_f$*"*, "*neighboring*", "*more or less*", "*vaguely*"$\}$. These operators change the membership function of *"*$m_\beta P_{ik}$*"* by using an expansion or a contraction function. Fig. 2B shows the influence of fuzzy operators on the basic property "*important*".

In order to simplify, when the concept of P_{ik} is unambiguous, the e-property can be denoted as "x is $f_\alpha m_\beta P_{ik}$". For instance, "The lightness of the object is really very weak" can be written as "The object is really very dark" ("*dark*" $\equiv$ "*weak*") where "*Lightness*" is the concept, "*really*" is the fuzzy operator, "*very*" is the modifier and "*dark*" ("*weak*") is the basic property.

In fuzzy logic, all linguistic words are associated to a fuzzy set. The designer has to describe the membership function for each possible e-property, i.e. all possibilities between operators and basic properties. In contrast, with our new model, the user has only to give basic properties. E-properties are automatically computed using operators and basic properties.[8] Moreover, our model seems to be nearest to natural description (according to linguistic studies) than the other models.

[7] This set can be extended to M_9 adding the operators "*very very little*" between "*extremely little*" and "*very little*", and, "*very very*" between "*very*" and "*extremely*".

[8] Details of the application of these operators are given in [30]. In this paper, we only give the application principle.

3.4. E-properties using domain's values

Sometimes, the designer could want to refer a given value or interval. Therefore, he uses an e-property with another type of basic property. Such property, called *valued e-property*, refers to some of the domain's values like "x is f_α of U", "x is f_α between U and V", "x is f_α at least U", "x is f_α at most U", "x is $f_\alpha m_\beta$ greater than U" or "x is $f_\alpha m_\beta$ lower than U" where U and V are two domain's values. In this case, fuzzy operators and modifiers do not always exist. For instance, we could use a description like "the height of the house is really greater than 4 m", "the park area is between 1000 and 2000 m^2" … but we cannot use "the number of houses is very between 50 and 200".

All but two properties are interpreted as classical intervals $[a..b]$.[9] They can be defined by the LR-function: $\langle 0, a, b, 0\rangle$ and any L and R functions. The two exceptions are "x is $f_\alpha m_\beta$ greater than U" and "x is $f_\alpha m_\beta$ lower than U". They can be interpreted as fuzzy sets (Fig. 3). They are also defined by a LR-function as standard basic properties. This function is built regarding the domain's value, domain's boundaries and operators used.

3.5. Creating a CSP using e-properties

Such a model of e-property is an easy way to build a CSP. A CSP variable is associated to each generating concept. To build domains associated to variables, two ways can be found. In the first way, they are considered as domains of our model. Consequently, e-properties allow us to create unary constraints over CSP domains. The second way is to reduce directly CSP domains to valid values. We prefer the first to the second one because the number of constraints and domains width is lower. As our model uses fuzzy sets, domains are defined by the support set of the e-property (all values with a membership degree greater than 0).

Non-generating concepts allow us to create constraints. The constraint is the function of the concept's measure and its values have to be in a given set of the concept's domain. They are created only if they are used in at least one property of the description.

Example 5. For instance, let us take our RZ-modeler and the description E: "The front width is rather important". The CSP associated with this description, denoted $csp(E)$, is:

$$csp(E) = (X = \{X_1 = FrontWidth, X_2 = Depth\},$$
$$D = \{D_1 = [28..39]_1, D_2 = [20..60]_1\},$$
$$C = \{\}).$$

[9] However, some solutions can be provided to introduce fuzzy values to easily interpret the application of fuzzy modifiers and fuzzy operators [31].

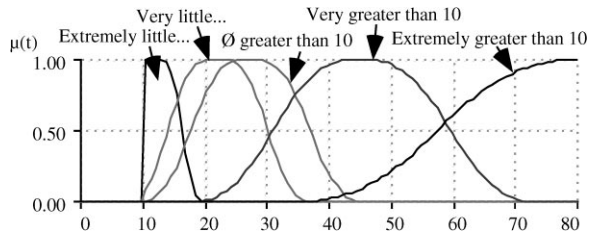

Fig. 3. Examples of fuzzy e-properties using domain's values with some modifiers.

Example 6. If the designer adds "The surface is weak", the CSP becomes:

$$csp(E') = (X = \{X_1 = FrontWidth, X_2 = Depth\},$$
$$D = \{D_1 = [28..39]_1, D_2 = [20..60]_1\},$$
$$C = \{C_1 = \{m_{Surface} = X_1 * X_2 \in [222..1163]_1\}\}).$$

3.6. R-properties

E-properties are based on a single element or an object of the scene (using a unary concept). We can also find *relative properties* (*r-properties*) built as "C_i between $\{x_r\}$ is $f_\alpha m_\beta P_{ik}$" (x_r, $r \in [1..n]$, are n elements). The concept used to build this property is a concept that refers to more than one element (often a binary concept). R-properties are processed as e-properties.

Example 7. "The distance between a and b is really important" is an *r*-property. It refers to the objects "*a*" and "*b*".

4. C-properties

4.1. Definition

E-properties and r-properties refer to only one concept over one or more objects. However, it is also useful to compare two objects according to a concept. Such a property is called *connected property* (*c-property*). This property is used to compare one or more concepts that belong to one or more objects of the scene. Domains of referring concepts need to be semantically equivalent. Most of usual properties of that kind are *binary connected properties*. Such properties can use one or two objects in the scene according to one or two basic properties not necessary from the same concept. Concepts are unary or binary concepts.

This property is built as "C_i {[of X_p] — [between X_p and X_r] — [Ø]} is $f_\alpha k_\beta s_\delta P_{ik}$ {[than] — [as]} C_j {[of X_q] — [between X_q and X_s] — [Ø]}" where i and j such as D_i (of C_i) and D_j (of C_j) can be semantically (and possibly arithmetically) compared, f_α is a fuzzy operator,

k_β is a *comparison operator* that can be fuzzy ("*a few*", "*medium*", "*a lot*", "*extremely*", "2 *times*", "*of V units*", "*between U and V units*", "*at least V units*" …) and s_δ is the *direction operator* ("*less*", "*as*" and "*more*") according to the direction of the *reference property* P_{ik} (X_i is an element of the scene with $i \in \{p,q,r,s\}$).

Example 8. "The front width of A is really more important than the front width of B" and "The front width of A is 3 times more important than the depth" are c-properties.

C-properties can be divided into two types according to the comparison operator used:

- c-properties based on a difference (with operators as "*a lot*", "*a few*", etc.), called *Difference c-properties*;
- c-properties based on a ratio (with operators such as "2 *times*", etc.), called *Proportion c-properties*.

To each type of c-property, we associate a particular concept (with two concepts as parameter). They are denoted "$Diff_{C_i,C_j}$" and "$Prop_{C_i,C_j}$".

4.2. Difference c-properties

When the designer uses such a property, he wants to compare two values of two concepts (sometimes the same). For this concept of difference, the measure is based on the compared concept's ones. For a c-property like "C_i of X_p is $f_\alpha k_\beta s_\delta P_{ik}$ {[than] — [as]} C_j of X_q" and the measures m_i (for C_i) and m_j (for C_j), the measure of the concept of difference is: $m_{Diff_{C_i,C_j}} = m_i - m_j$. The associated domain $D_{Diff_{C_i,C_j}}$ is $[-\delta..\delta]_u$ where $\delta = Max(|Bm_i - BM_j|, |BM_i - Bm_j|)$ and $u = Min(u_i, u_j)$.

Interpreting difference c-properties with a precise comparison operator (like "A is 2m more wide than B") is easy to do. Indeed, the difference is explicitly given by the property. Therefore, we are interested in those with vague operators.

We assume that exist equivalence between the difference c-property and a valued e-property defined in the difference concept's domain. Basically, "C_i of X_p is $f_\alpha k_\beta s_\delta P_{ik}$ {[than] — [as]} C_j of X_q" seems to be equivalent to "$Diff_{C_i,C_j}$ is $f_\alpha m_\beta$ *PP*" where "*PP*" is domain's valued basic property like "*greater than* 0" (a positive property) or "*lower than* 0" (a negative property). This last scheme provides the fuzzy subsets of authorized differences. For instance, "The front width of A is really more important than the front width of B" can be interpreted as "The difference between the front width of A and the front width of B is really greater than 0".

To find a good equivalence, we assume the direction operator is signed such as: sign("*less*") is negative (<0), sign("*as*") is neutral ($=0$) and sign("*more*") is positive (>0). So, The property's sign is the same as the direction of the comparison: $Direction(\text{c-}prop) = Direction(f_\alpha k_\beta s_\delta P_{ik}) = \text{sign}(s_\delta) * \text{sign}(P_{ik})$. Therefore, the valued basic property direction is founded with these relations:

- $\text{sign}(s_\delta) = \text{sign}(P_{ik}) \neq 0$
  $\Rightarrow Direction(f_\alpha k_\beta s_\delta P_{ik}) > 0$
  $\Rightarrow \text{sign}(PP) > 0 \Rightarrow PP =$ "*greater than 0*";
- $\text{sign}(s_\delta) \neq \text{sign}(P_{ik}) \neq 0$
  $\Rightarrow Direction(f_\alpha k_\beta s_\delta P_{ik}) < 0$
  $\Rightarrow \text{sign}(PP) < 0 \Rightarrow PP =$ "*lower than 0*";
- $\text{sign}(s_\delta) = 0$
  $\Rightarrow Direction(f_\alpha k_\beta s_\delta P_{ik}) = 0$
  $\Rightarrow \text{sign}(PP) = 0 \Rightarrow PP =$ "*of 0*".

The last step is to find the value of m_β. Therefore, let us take the set of P modifiers M_P. We can select a set of comparison operators denoted: $K_R = \{k_\gamma | gamma \in [1..Q]\}$.

The translation is given by:

$\forall \alpha \in [1..R], \exists \beta \in [1..P] : k_\alpha \leftrightarrow m_\beta$ with $k_\alpha \in K_R$ and $m_\beta \in M_P$.

We can select this following set: $K_7 = \{$"*very little few*", "*little few*", "*few*", "∅", "*lot*", "*enormously*", "*infinitely*"$\}$ with the relation $k_\alpha \leftrightarrow m_\alpha$. Fig. 4 shows the equivalent valued e-properties in the concept of difference.

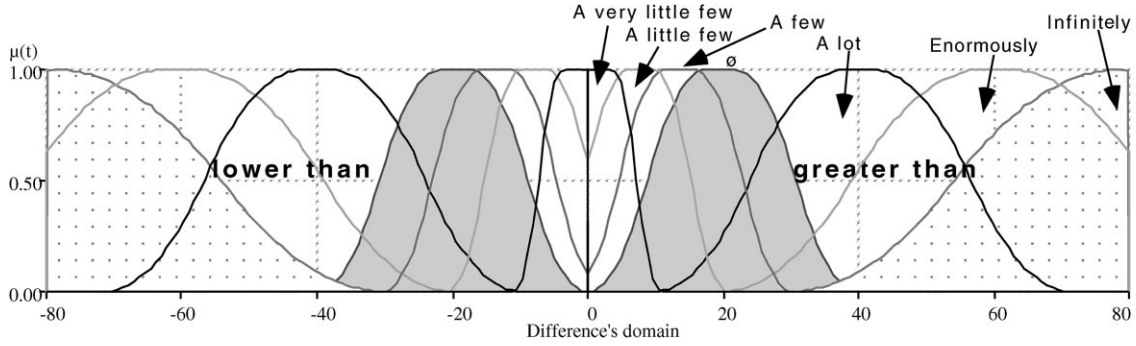

Fig. 4. Difference concept.

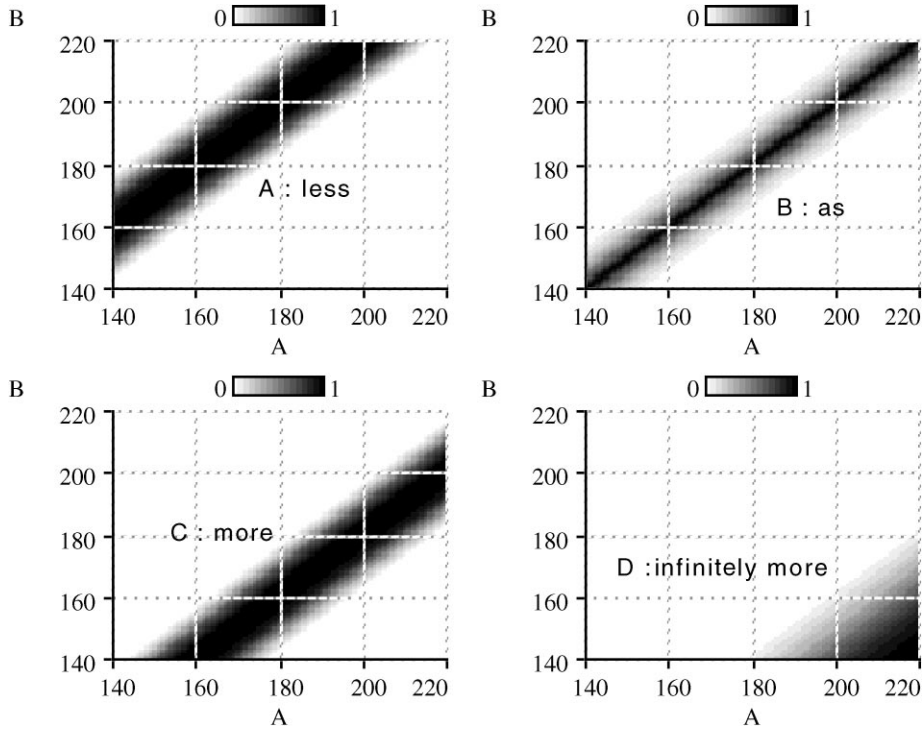

Fig. 5. Examples of object comparison.

Therefore, the property "The depth of A is more important than the depth of B" is translated as "The difference of depth between A and B is greater than 0". Fig. 5 shows valid couple of values according to some comparisons.

4.3. Proportion c-properties

When the designer uses such a proportion c-property, he wants to give a proportion (a ratio) between to values of two concepts (may be the same). For this concept of proportion, the measure is based on the compared concept's measures too. For a c-property like "C_i of X_p is f_α p times $s_\delta P_{ik}$ than C_j of X_q" (with p a numeric value[10]) and the measures m_i (for C_i) and m_j (for C_j), the measure of the concept of proportion is: $m_{Prop_{C_i,C_j}} = (m_i, m_j) \in D_i \times D_j$.

For a given proportion property and a numerical value p, the membership function is defined as:

$\mu_{Prop_{C_i,C_j}}(a,b) = \mu_{\langle\alpha,a,b,\beta\rangle,LR}(|a - p*b|)$ if $Direction(s_\delta P_{ik}) > 0$ and $\mu_{\langle\alpha,a,b,\beta\rangle,LR}(|p*a - b|)$ otherwise (α and β are two floats).

[10] The operator p can be fuzzy ("*a dozen*", etc.) but currently the solution is not acceptable.

The Fig. 6 shows the membership function when p equals 3 for property like "The front width of A is 3 times more important than the depth".

4.4. CSP and c-properties

C-properties allow building a set of constraints. For difference c-properties, the associated constraint is built like e-properties with a non-generating concept: the difference of the two measures must be in an interval defined by the corresponding valued e-property.

For a proportion c-property as studied, the generated constraint depends on the sign of the direction of the property. If this direction is positive then the constraint is $X_i = p*X_j$ else the constraint is $X_i*p = X_j$.

Example 9. For instance, always with our RZ-modeler, the designer can say: "The front width is more important than the depth" (i.e., "The difference between front width and the depth of A is greater than 0)". With this c-property, the associated csp is:

$$(X = \{X_1 = FrontWidth, X_2 = Depth\},$$

$$D = \{D_1 = [10..40]_1, D_2 = [20..60]_1\},$$

$$C = \{C_1 = \{X_1 - X_2 \in [10..30]_1\}\}).$$

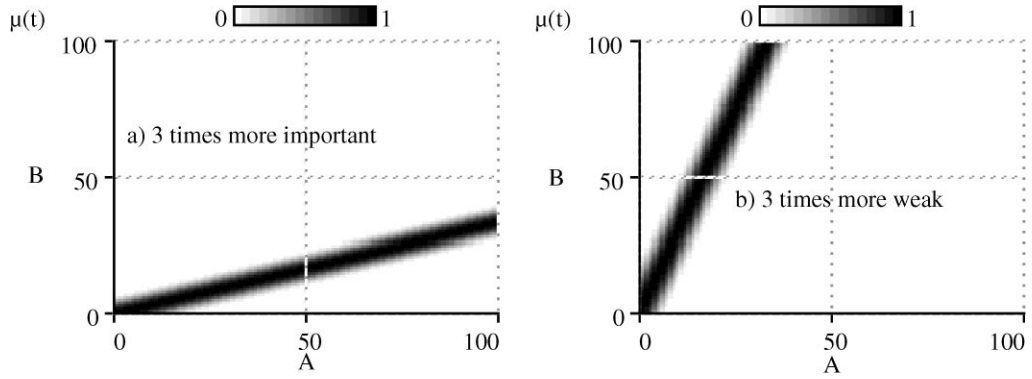

Fig. 6. Examples of object ratio.

Example 10. With a property "The front width is 3 times more important than the depth", the associated csp is:

$(X = \{X_1 = FrontWidth, X_2 = Depth\},$

$D = \{D_1 = [10..40]_1, D_2 = [20..60]_1\},$

$C = \{C_1 = \{X_1 = 3{*}X_2\}\}).$

Remark 11. Sometimes, the designer could want to change his description according to the current solution. Such a property, called *modifying properties* [28], has a scheme: "C_i [*of* X_p] is $f_\alpha k_\beta s_\delta P_{ik}$" with operators as defined for c-properties. For instance, the designer can say "The depth is really more important". These properties are interpreted as e-properties. We assume the designer implicitly adds "… than the current value of this concept" at the end of the property. Therefore, a m-property is easily translated to a valued e-property. For the CSP, this property adds a new unary constraint or directly reduces the concept's domain (as we saw for e-properties).

5. The negation

5.1. Principle

We are interested in the interpretation the designer gives to a property like "The lightness is not very weak": a property denoted as "C_i of x is not A" where A is a property "$f_\alpha m_\beta P_{ik}$". Then, sometimes the designer uses a negative sentence to make a style effect (litotes, etc.), to highlight a property he does not want, to express a lack of information, etc. Linguistically, this property is not equivalent to the logical property "*not-A*" which is defined by the membership function of A according to fuzzy logic. "*not-A*" has no sense in natural language. Indeed, the designer means that the object respects an affirmative e-property P, which is quite different from A, but based on the same concept. For instance, the designer may suggest "The lightness is medium" for "The lightness is not very weak" (Fig. 7). The aim is to search for *this* implicit e-property that is equivalent to the negative one[11] [37,38].

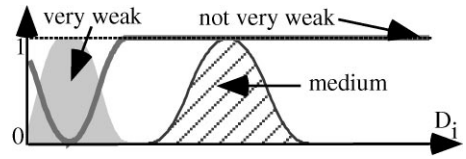

Fig. 7. Interpretations of "x is not A".

For such a process, we use the notion of linguistic interpretation. *The linguistic interpretation* of a sentence "x is A", where A is an e-property, is often considered, in linguistic [39], as a disjunction of properties "x is $\emptyset_f A$", "x is really A" and "x is neighboring A". The fuzzy operator set can be divided into two subsets G_1 (the subset of fuzzy operators understood over the property "x is A") and G_2 (the complementary subset of G_1 in F_Q). We can deduce: "x is A" $\Leftrightarrow$ {"x is $f_\alpha A$" with $f_\alpha \in G_1$}. For instance, in F_6, we have: $G_1 = \{$"*neighboring*", "*really*", "$\emptyset_f$"$\}$ and $G_2 = \{$"*exactly*", "*more or less*", "*vaguely*"$\}$. Therefore, the linguistic interpretation of the property "x is weak" is "x is neighboring weak" or "x is really weak" or "x is $\emptyset_f$ weak".

The linguistic negation of a property as "x is A", in a given domain, denoted "x is not A" means that (1) the designer rejects all linguistic interpretations of "x is A" and (2) he refers to another e-property P in the same domain. As a result, "x is not A" can be equivalent to

[11] Sometimes, the designer may give a negation without the implicit property. "x is not …" is translated as "x is *all but* …": the standard logical process.

"x is P". Using the notions we defined before, the negation of "x is A" in a given domain means that the designer rejects all properties as "x is $f_\alpha A$" with $f_\alpha \in G_1$ and refers to a linguistic negation of "x is not A" *one of these properties*:

- "x is $f_\gamma B$" with B an e-property defined on the same domain than $A, B \neq A$ (with a basic property often different) and $f_\gamma \in F_Q$ *or*
- "x is $f_\beta A$ with $f_\beta \in G_2$ *or*
- "x is $m_\delta A$" with $m_\delta \neq \emptyset$ and usable with modifiers of A.

Possible properties of a concept are defined by all combinations between fuzzy operators, fuzzy modifiers and basic properties of this concept. *Plausible properties* are possible properties that can be selected as linguistic negation of a given property. They are selected using a nuanced similarity of fuzzy sets [38,40]. Properties are selected if their membership function has a *weak agreement or a rather little similarity* to the rejected ones (all linguistic interpretations of the denied property). The number of resulting plausible properties is rather important yet. Therefore, we propose a set of new strategies to help the designer to select quickly the good one.

5.2. Strategies

The main problem is to find the implicit property P of a negation that, by definition, does not point out P. Thus, we propose some linguistic criteria to help for this choice.

5.2.1. Linguistic criteria

In order to help the designer to choose quickly the appropriate implicit property of "x is not A", we propose to order the set of plausible properties. This sorting process is based on two principles:

- *Simplicity*: The designer usually means a property often built with a very simple scheme.[12]
- *Preference*: The designer often means a signed basic property that is different from the A's one.

5.2.2. Linguistic value of a property

We have defined the linguistic complexity of an operator and of an e-property to sort plausible properties.

First, the *linguistic complexity of an operator* is a function denoted cl as $cl(\text{"}\emptyset\text{"}) = 0$ and for each operator "op", cl("op") is the number of word for "op". However, as positive descriptions are easier to formulate, negative operators are penalized in comparison with positive ones. So, we have: $cl_m(m_\alpha) = cl(m_\alpha) + 1$ if $m_\alpha <$ "$\emptyset$" and $cl_m(m_\alpha) = cl(m_\alpha)$ otherwise.

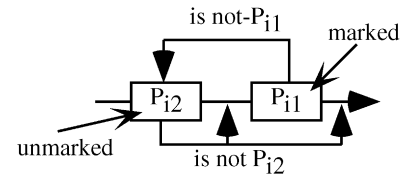

Fig. 8. The negation for a couple of marked terms.

The *linguistic complexity of an e-property* A ("$f_\alpha m_\beta P_{ik}$") is a function CL: $CL(A) = cl_m(m_\beta) + dc{*}cl(f_\alpha)$ where dc is an integer. This definition takes into account the fact the designer easily provides a property with a single operator (simplicity principle). Moreover, with the value dc, we consider that it is more complex to use a fuzzy operator instead of a fuzzy modifier.[13]

Finally, the *linguistic value of an e-property* P is a function using the linguistic complexity of P adjusted according to the preference principle: a strong priority is given to signed basic properties that are different from the denied property one. So, plausible properties are sorted according to their growing linguistic values.

5.2.3. Linguistic mark

Mark and antonym are notions proposed in [36,41,42]. In a semantic category, sometimes one of the two signed terms (often two antonyms) is used as a symbol for the whole category. This term is called *unmarked* because it refers either to the category or to a semantic unit. The other term is called *marked*. The marked term generally implies the absence of the unmarked one. The first one is often the negative unit of the category. However, the unmarked one has a favorable value and is often the positive unit. For instance, the "*lightness*" concept is composed of three basic properties {"*dark*", "*medium*", "*light*"}. The positive basic property "*light*" is unmarked and the negative one "*dark*" is marked.

These notions are interesting to interpret a negative property. In fact, independent of the sorting strategy according to the linguistic value, linguistic studies propose some solutions to calculate the implicit property in simple cases (for precise basic properties in a given context). Let us take a concept C_i with two basic properties $\{P_{i1}, P_{i2}\}$ ordered as the Fig. 8 shows.

The first one is unmarked (the negative antonym) and the second one is marked. So, the negation of P_{i1} is equivalent to P_{i2}. This is not a symmetric rule: the negation of P_{i2} is computed as standard properties. However, the marked notion can be used *to direct* the search, considering the implicit property on the *same side* as the unmarked one [36,42] according to the given

[12] Currently, there is a small set of operators, often including the $\emptyset$ operator.

[13] In practice, 3 seems to be a good value for dc.

order (Fig. 8). Properties which are on the same side than P_{i1} with regard to P_{i2} have therefore a high priority. This solution is valid only if the concept satisfies the *gradation of properties* notion [36] that is a global order relation in P_i: $P_{ia} \leqslant P_{ib} \Leftrightarrow a \leqslant b$. This order can be extended to properties modified by operators.

Sometimes, C_i includes a single basic property $\{P_{i1}\}$. In this case, the negative form can be renamed, adding a prefix to the basic property's name like [42]: *im*, *no-*, *in*, *un*, *not-*, etc. Indeed, the linguistic negation is a new affirmative property that can be added to the concept. This property can be created with the complementary subset of the initial one. P_i may implicitly become $P_i = \{P_{i1}, P_{i2}\}$ with $P_{i2} \equiv Not - P_{i1}$. So, the interpretation of "x is not P_{1i}" gives "x is un-P_{i1}" and conversely. A concept with only one basic property is actually a concept with a couple of basic properties.

Furthermore, C_i may include three properties $\{P_{i1}, P_{i2}, P_{i3}\}$. If a couple of antonyms can be found, they are treated as before. The third property is interpreted in a standard way. For instance, for the concept of "*lightness*", we can easily propose three basic properties like {"*weak*", "*medium*", "*important*"}. If "*weak*" is marked, "x is not important" is equivalent to "x is weak". Nevertheless, "x is not weak" is translated by a property, maybe different from "x is important", but on the same side regarding "*weak*".

Conversely, our interpretation does not *reject* properties but *put them at the end* of the set of properties that are not really plausible. The order is partially broken if the basic property of the denied property is a member of a marked couple. Therefore, when the basic property of a negation is an unmarked term, the plausible property that has default operators and a marked basic property is placed at the first place. However, when the basic property of a negation is a marked term, plausible properties with a unmarked basic property are proposed at the end (Fig. 9).

5.2.4. Deleting complex sentences

As properties are automatically created, some of them may have no meaning in natural language. They are too *complex* to be selected by the designer. For instance, a property like "The hue is vaguely very very little weak" could be generated. These are often properties with a lot of operators and a high linguistic value. It could be interesting to propose a filtering method to delete these properties. A possible solution consists in avoiding some operators that are too complex or not often used. In particular, these operators lead to too complex properties: "*very very*", "*very very little*", "*extremely little*", "*more or less*" … Very often, they are used alone (with the default operator "∅"). So, they must be proposed only at the end.

Moreover, according to the principle of simplicity, the number of words used to build a property can be limited.

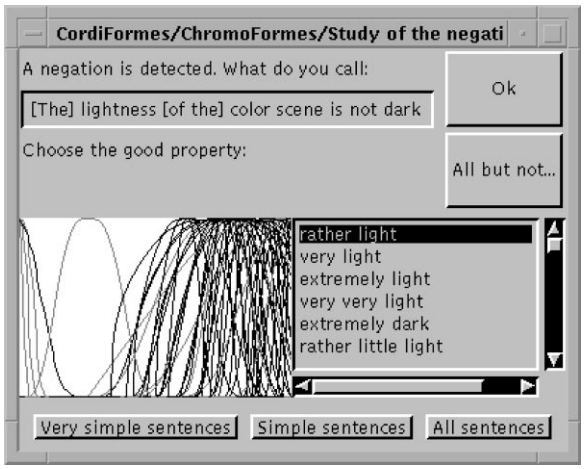

Fig. 9. Plausible properties for the negation of "*dark*".

More than two words produce a too complex property to be easily manipulated by the designer, especially when it is implicit. For instance, the negation of "x is weak" (Fig. 9) is probably "x is very important" but not "x is more or less very very little important"

In order to improve the search, the designer must have to choose between only a few property classes according to the principle of simplicity. We divided plausible properties into three levels:

(1) the *global selection* is made up of all plausible properties;
(2) the *large simplicity* includes only the plausible properties that have one default operator: properties like "x is $\emptyset_f \emptyset P_{ik}$", "$x$ is $f_\alpha \emptyset P_{ik}$" or "$x$ is $\emptyset_f m_\beta P_{ik}$" ($cl_m(m_\beta) * cl(f_\alpha) = 0$);
(3) the *strict simplicity* includes only the plausible properties selected in (2) and having one or two words for the couple of operators used ($cl_m(m_\beta) + cl(f_\alpha) < 2$).

The number of properties at level 3 (Fig. 10) is clearly less important. So, the choice is easier.

5.3. Negation of valued e-properties

The negation interpretation can be extended to valued e-properties. However, the number of this kind of property is infinite. So, the interpretation must be adapted to take this problem into account during the creation of plausible solutions. Some heuristics help us to select the more significant ones. Actually, plausible valued e-properties are based on basic properties and on values used for the denied property or to define the concept's domain (domain boundaries, medium value, etc.). Sometimes, the opposite property is selected. For instance, the negation of "*greater than U*" will produce plausible properties as

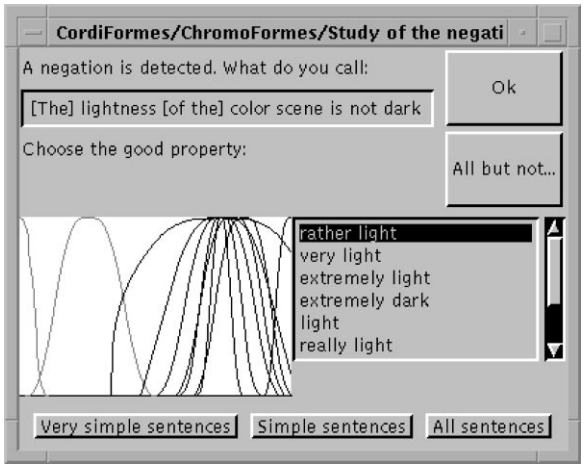

Fig. 10. Very simple interpretations for "*dark*".

$f_\alpha m_\beta$ lower than U" and conversely. Note that the designer usually uses a negation because he has not got a precise idea about the implicit property. So, properties based on basic properties and properties like "*greater than*" and like "*lower than*" are better, because they are naturally fuzzy.

When the designer selects an interval for a negation of an non-valued e-property A, a new property definition can be deduced for this concept. A learning process must be used to add this new property in the set of basic properties. Then the users have to give the property's name (often using the basic property of the denied one and adding a negative prefix as un-P_{ik}, not-P_{ik}, etc.).

6. Implementation

In order to validate our model, we wrote a small declarative modeler program (illustrated in Fig. 11). We used the kernel of the Cordiformes Project [31,43] (a set of tools to build declarative modeling programs) to write this program. Our purpose is not to write a full modeler, but only to check if, when using our model, we build the scenes we described.

Our example is a 3D elevation of our RZ-modeler. It is based on a box description:[14] "There are three boxes. Define a box A. A is green. A is tall and not deep. Define a box B. B is red. The width of B is very important. B is not tall. Define a box C. C is like gold. C is behind A. C is on the right. C is taller than B."

[14] Our program currently deals with boxes, but we are working on implementing the whole model and using these boxes to generate houses or buildings.

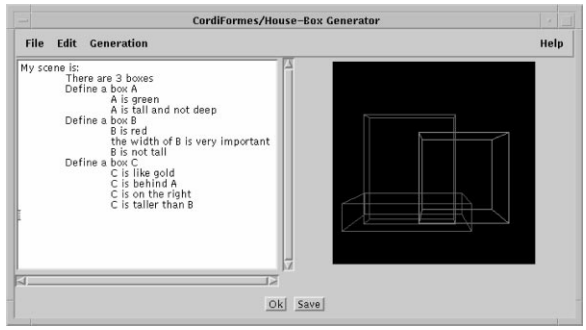

Fig. 11. The generation window.

The designer gives his/her text description according to the pseudo-natural language we propose. Before the generation, the two negations are treated as we saw in Section 5 using dialogs to highlight the two corresponding implicit properties. The modeler shows a draft image of the scene. If the user finds a scene he wants to keep, he can ask the modeler to generate a more realistic image. If he does not, he can ask the modeler for another picture. Pictures in Fig. 12 are six solutions we chose among a large set of solutions to the description given before.

This example is very simple but it is a good validation of our model covering previous properties and the negation process. For more complex objects, the generated CSP will be composed of all generating concepts of the objects [43]. Indeed, we assume that complex objects can be always divided into basic generating concept (besides, objects with a non-basic concept could not be described because linguistic studies highlight that human thinking prefers simple concepts).

7. Conclusion

In order to model properties of a scene description, we propose a new independent domain model. This model is based on the notions of concept, fuzzy properties and generic operators. It concerns several types of properties and allows us to build a CSP in order to produce solutions to the designer's description.

Instead of standard interpretation of negative properties in declarative modeling and fuzzy logic, the interpretation of the negation is not computed by the logical negation without meanings in natural language. We propose a new interpretation of the negation based on linguistic studies. For us, denying an e-property means understanding another implicit e-property in the same concept. We propose a set of tools to build and to sort plausible properties that can be these implicit properties. The designer has only to choose his favorite one. If the implicit property is not generated, the designer can:

- select another property that is not very different,

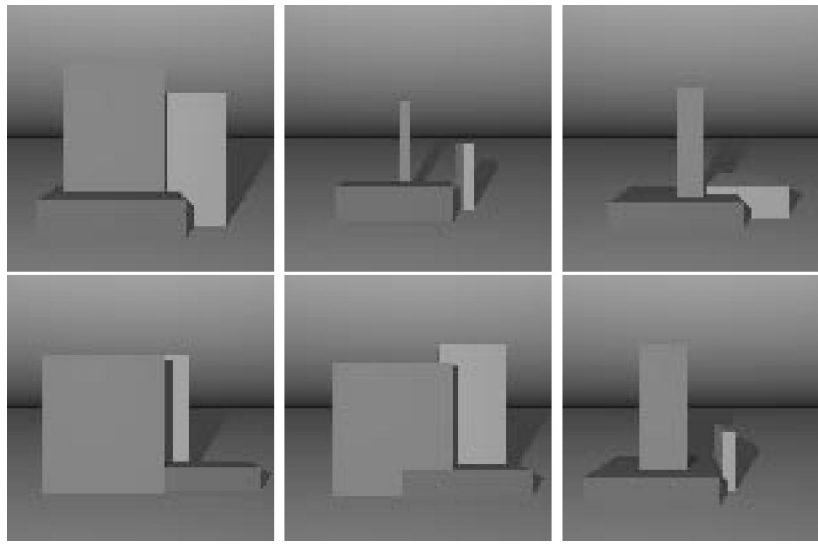

Fig. 12. Solutions to the description.

- change the description (change the denied property),
- decide that the negation is equivalent to "*all but not*".

Nevertheless, we help the designer to formulate his/her description, even if he/she does not find the property solution. This linguistic interpretation of the negation is an important step in the design process and allows the designer to go thoroughly into his/her idea and particularly to refine properties he/she uses in his description.

This linguistic negation's study provides some good results in the design process with declarative modelers. Moreover, the determination of an implicit e-property in a negation improves the generation step regarding to the standard negation. This solution allows to reduce the universe of scene solutions and to select only the solutions that are really close to the designer's idea.

One can argue that the interpretation as presented in this paper does not need fuzzy sets. However, this model is first a good interpretation of natural language and modifications provided by the operators. Second, some works [44] study the use of fuzzy sets by CSP. They call such constraint problem *fuzzy-CSP* (*FCSP*). According this work, fuzzy CSP allows us to introduced new operators on properties to build description like "The width may be wide", i.e. properties with different priority. However, FCSP has to be improved to allow complex 3D geometric properties, very complex concepts and redundancy. Finally, fuzzy sets allow the modeler to value a solution. This value helps the designer to understand his solution. Moreover, this valuation can be used by understanding tools to explore the solution set classifying the elements [45] and to install learning processes to find implicit properties of solution the designer likes.

Our model allows us a better interpretation of properties and a natural approach for operators' applications. Moreover, it gives good results in pseudo-English and in pseudo-French but we have not studied other languages. We suppose that all Latin languages would provide similar results. This model can be also used in a more general point of view as a request language (for example to use in Data Bases as an SQL-like language). It also allows building powerful generating tools to produce solutions. However, the model we present in this paper has to be improved. First, the interpretation of proportion c-properties must be updated because we do not take into account fuzzy operators like "*a dozen*" etc. Then, this model does not process quantifiers. Some works exist [46] but solutions are not easy to manage. Finally, the linguistic negation has to be improved to process not only e-properties but also the other properties like c-properties.

Acknowledgements

We are grateful to J.-Y. Martin, E. Languénou and, especially, Pr. M. Lucas for providing a number of useful comments on earlier versions of this paper.

References

[1] Lucas M. Equivalence classes in object shape modeling. In: editor. Kunii TL Working Conference on Modeling in Computer Graghics. Tokyo, Japan: Springer, 1991, IFIP TC5/WG 5.10, p. 17–34.

[2] Lucas M, Desmontils E. Les Modeleurs Déclaratifs. Revue Internationale de CFAO et d'Infographie 1995;10(6): 559–85.
[3] Colin C, Desmontils E, Martin J-Y, Mounier J-P. Working with declarative modeler. Computer Networks and ISDN Systems 1998;30(20–21):1875–86.
[4] Woodbury RF. Searching for designs: paradigm and practice. Building and Environment 1991;26(1):61–73.
[5] Kochhar S. CCAD: a paradigm for human-computer cooperation in Design. IEEE Computer Graphics and Applications 1994;14(3):54–65.
[6] Tuerino YA, Mochizuki K, Kishino F. Interactive 3D computer graphics driven through verbal instruction: previous and current activities at ATR. Computers & Graphics 1994;18(5):621–31.
[7] Giunchiglia E, Armando A, Traverso P, Cimatti A. Visual representation of natural language scene description. IEEE Transaction on Systems, Man and Cybernetics 1996;26(4):575–89.
[8] Schoeneman C, Dorsey J, Smits B, Arvo J, Greenberg D. Painting with light. In: Siggraph '93, Computer Graphics. Proceedings, Annual Conference Series 1993. CA, Anaheim, 1993, p. 143–6.
[9] Poulin P, Ratib K, Jacques M. Sketching shadows and highlights to position lights. In: Computer Graphics International '97. 1997, p. 56–63.
[10] Rau-Chaplin A, MacKay-Lyons B, Spierenburg PF. The LaHave house project: towards an automated architectural design service. In: Cadex '96 1996, p. 24–31.
[11] Rau-Chaplin A, MacKay-Lyons B, Doucette T, Gajweski J, Hu X, Spierenburg PF. Graphics support for a world-wide-web based architectural design service. In: Compugraphics '96. 1996, p. 83–92.
[12] Khemlani L. GENWIN: a generative computer tool for window design in energy-conscious architecture. Building and Environment 1995;30(1):73–81.
[13] Poulet F, Lucas M. Modelling megalithic sites. In: Eurographics '96. Poitiers, France, 1996, p. 279–88.
[14] Martin P, Martin D. PolyFormes: software for the declarative modelling of polyhedra. The Visual Computer 1999;15:55–76.
[15] Snyder JM. Interval analysis for computer graphics. In: Siggraph '92, Computer Graphics Proceedings, Annual Conference Series 1992. Chicago, USA, 1992, p. 121–9.
[16] Chauvat D, Martin P, Lucas M. Modélisation déclarative et contrôle spacial. Revue Internationale de CFAO et d'Infographie, French International Journal of CADCAM and Computer Graphics 1993;8(4):411–36.
[17] Zeleznik RC, Herndon KP, Hughes JF. SKETCH: an interface for sketching 3D scene. In: Siggraph '96, Computer Graphics Proceedings, Annual Conference Series 1996, 1996, p. 163–70.
[18] Benhamou F. Interval constraint logic programming. In: Podelski A. editor. Constraint programming: basics and trends, vol. 910, LCNS, Berlin: Springer 1995: p. 1–21.
[19] Jardillier F, Languénou E. Screen-space constraints for camera movements: the virtual cameraman. In: Eurographics '98. Lisbon, Portugal, 1998, p. 175–86.
[20] Allen JF. Maintaining knowledge about temporal intervals. Communication of the ACM 1983;26(11):832–43.
[21] Donikian S, Hégron G. Constraint management in a declarative design method for 3D scene sketch modeling. In: Workshop on Practice of Constraint Programming. Newport, USA, 1993.
[22] Tsang E. Foundations of constraint satisfaction. London: Academic Press, 1993.
[23] Champciaux L. Declarative modelling: speeding up the generation. In: CISST '97, Las Vegas, USA, 1997, p. 120–29.
[24] Jones PF. Four principles of man-computer dialogue, In: editor. Booth KS Tutorial: computer graphics. Silver Spring, MD: IEEE Computer Society Press 1979. p. 260–5.
[25] Colin C. Les propriétés dans le cadre d'une modélisation géométrique déclarative. In: MICAD '92. Paris, France, 1992, p. 75–94.
[26] Zadeh LA. Fuzzy sets. Information and Control 1965;8:338–53.
[27] Cohn A, Randell DA, Cui Z. Taxonomies of logically defined qualitative spatial relations. International Journal of Human-Computer Studies 1995;10:831–46.
[28] Desmontils E, Martin J-Y. Properties taxonomy in declarative modeling. In: CISST '97. Las Vegas, USA, 1997, p. 130–8.
[29] Dohmen M. A survey of constraint satisfaction techniques for geometric modeling. Computer & Graphics 1995;19(6): 831–45.
[30] Desmontils E, Pacholczyk D. Vers un traitement linguistique des propriétés en Modélisation Déclarative. Revue Internationale de CFAO et d'Infographie 1997;12(4):351–71.
[31] Desmontils E. Le projet CordiFormes: une plateforme pour la construction de modeleurs déclaratifs. Thèse de Doctorat, Univ. de Nantes, France, 1998.
[32] Dubois D, Prade H. Fuzzy sets and systems – theory and applications. New York, USA: Academic Press, 1980.
[33] Hjelmslev L. La catégorie des cas (1 et 2). Acta Jutlandica, 1935–37.
[34] Brondal V. Les parties du discours: étude sur les catégories linguistiques, Copenhague, 1948.
[35] Brondal V. Essais de linguistique générale, Copenhague 1943.
[36] Ducrot O, Schaeffer J-M. Nouveau dictionnaire encyclopédique des sciences du langage. Editions du Seuil, Paris, France, 1995.
[37] Pacholczyk D, Desmontils E. A qualitative approach to fuzzy properties in scene description, In: CISST '97, Las Vegas, USA, 1997, p. 139–48.
[38] Pacholczyk D, Desmontils E. A linguistic interpretation of affirmative or negative information in declarative modeling in image synthesis. In: IIS '98. IPI. Pan, Editor Malbork, Poland, 1998, p. 150–59.
[39] Scheffe P. On foundations of reasoning with uncertain facts and vague concepts. In: Fuzzy reasoning and its applications. 1981, p. 189–216.
[40] Dubois D. Prade H. Unifying view of comparison indices in a fuzzy set-theoretic framework. Fuzzy Set and Possibility Theory 1980, p. 3–13.
[41] Horn LR. A natural history of negation. Chicago: The University of Chicago Press, 1989.
[42] Muller C. La négation en français, Publications romanes et françaises, Genève, Suisse, 1991.

[43] Desmontils E, Martin J-Y. Vers la Conception Assistée de Modeleurs déclaratifs. Revue Internationale de CFAO et d'informatique graphique 1999;14(2):133–52.
[44] Fargier H, Dubois D, Prade H. Problèmes de satisfaction de contraintes flexibles : une approche égalitariste. Revue d'intelligence artificielle 1995;9(3):312–53.
[45] Champciaux L. Classification: a basis for understanding tools in declarative modeling. In: Compugraphics '97. Villamoura, Portugal, 1997, p. 106–16.
[46] Bosc P, Liétard L, Prade H. On fuzzy queries involving fuzzy quantifiers. In: ECAI '96. 1996.

Dr. E. Desmontils

CURRICULUM VITAE

Emmanuel Desmontils received his Ph.D. in computer science from the University of Nantes (France) in 1998. He teaches computer science at the University of Nantes (France). He is a researcher at the "Institut de Recherche en Informatique de Nantes (IRIN)" and works on Declarative Modeling and Artificial Intelligence (Knowledge Engineering).