Design of a New Constraint Solver for 3D Declarative Modelling: JACADI

Olivier Le Roux Véronique Gaildrat René Caubet

Departement of Computer Science I.R.I.T
Université Paul Sabatier; 118 route de Narbonne, 31062 Toulouse Cedex, France
Phone. (33) 05 61 55 83 29 Fax. (33) 05 61 55 62 58
E-mail : { leroux, gaildrat, caubet }@irit.fr

Abstract

Many works have proved interest and importance of constraint solvers in declarative modelling. In order to assist user in three-dimensional scene design, an interactive constraint solver must be robust and efficient in all circumstances. Nevertheless, user's freedom should not be too restricted by limited solver's capabilities. Concerning space planning problems, it seems necessary to take into account objects with various orientations and not only isothetic[1] boxes. In this paper, we consider the design of a new dynamic and hierarchical constraint solver called JACADI[2]. Based on a CSP[3] extension and an adapted filtering technique, this solver will be able to give solutions to complex 3D-positioning problems. JACADI will be the core of a new version of DEM²ONS[4].

Keywords: space planning, constraint solver, geometric constraints, CSP extension, non-isothetic positioning

1 Introduction

To easily produce realistic virtual environments, powerful tools providing geometric and photometric data are necessary. Declarative modelling, modelling with constraints and multimodal interaction are three paradigms that should be used to build tools that are able to assist designers in their tasks. Our works take place after researches concerning the DEM²ONS project: [GCR93, KGC97, Kwa98]. The goal of this project is to propose a tool able to build a 3D scene according to a declarative description. The main component of this system has been a constraint solver called ORANOS [KGC98a, Kwa98]. ORANOS retrieves constraints from declarative layer and try to achieve a 3D scene (figure 1) that satisfy most of the constraints. The user can modify his description, then the system proposes a new scene: this solver is dynamic. If the user gives an over-constrained or inconsistent description then the system proposes an incomplete but consistent scene. It's due to solver capability to manage weighted constraints.

Unfortunately, ORANOS is too limited to produce realistic scenes. In particular, it is not designed to take into account non-isothetic objects.

In this paper we introduce some new concepts for general 3D-positionning problems using geometric constraints.

We also present the basis of a new dynamic and hierarchical constraint solver, called JACADI, designed for realistic declarative space planning.

The paper is organised as follows. Chapter 2 gives general framework and briefly introduces concepts about declarative modelling, geometric constraints and space planning.

[1] An isothetic box is a box with faces parallel to reference axes.
[2] In french: "Jacques à dit..."; in english: "Simon-says"
[3] Constraint Satisfaction Problems
[4] DEclarative Multimodal MOdelliNg System

We also present an overview of DEM²ONS in order to situate the aim of this study in our global project. Chapter 3 introduces fundamental notions and discusses about Kwaiter [Kwa98] and Charman [Cha94, Cha95] works. In Chapter 4, we present a new CSP extension called DHSCSP[5] based on previous analysis. In this section we also give the basic concepts of our new constraint solver JACADI. Solving a DHSCSP is a NP-hard problem, so we explain in Chapter 5 a powerful filtering technique able to transform a DHSCSP in an equivalent but simpler problem. We terminate by giving a crude backtrack algorithm able to achieve solutions.

Example of very simple scene computed by ORANOS from description: "put the table against the left wall"; "put a book on the table"; etc. Note that all objects are isothetic.

Example of scene with non-isothetic objects we expect to produce with our future solver JACADI.

Figure 1: ORANOS vs JACADI

[5] DHSCSP, for Dynamic Hierarchical Spatial CSP

2 Framework and overview of our global project

2.1 Declarative modelling

Declarative modelling allows designers to describe scenes or shapes giving only high level properties. These properties are obtained by expressing the mental image of a scene in an appropriated language. Then a geometric model is built by the system in order to fit in with the description [CDMM97, DH93, KGC97, PRT98]. Unlike classical modellers, declarative modellers manage high-level abstraction concepts. Nevertheless, declarative modelling is not a substitute for classical modelling but must be seen as an extension. Using both the two approaches in the same tool (CCAD[6]) has interested Sellinger [SP96]. Underlying problems are about knowledge keeping and data transmission between the two entities.
Kwaiter [Kwa98] considers declarative modelling as an additional layer for classical modelling tools to free the designer from tedious tasks.
For more information, interested reader can refer to GEODE[7] workgroup that identify most of aspects of declarative modelling.

2.2 Geometric constraints

Modelling with constraints is very often used in declarative modelling. In fact, constraint formalism allows expressing complex design problems as constraint satisfaction problems. For a complete bibliographical survey about modelling with constraints you can refer to [KGC98b]. Following chapters deal with a particular constraint satisfaction problem: space planning and geometric constraints. A geometric constraint is a relation concerning one or more geometric parameters owned by one or more objects [Woo87]. As Charman said, there is no consensual taxonomy about geometric constraints. We use the taxonomy proposed by Charman [Cha95] because in our opinion it seems to be appropriated to our problem. He distinguishes shape constraints (shape of objects)

[6] CCAD : Cooperative Computer-Aided Design

[7] GEODE is a GDR-PRC AMI team. It includes MGII team from IRIN, CERMA, architectural school of Nantes, and images synthesis team from EMN.
www.emn.fr/dept_info/GEODE

and spatial constraints (position of objects). He divided spatial constraints in four subsets:

- topological constraints: "Objects A and B don't overlap"
- distance constraints: "the chair is one meter from the table"
- angular constraints: "the door form a 45° angle with the wall"
- orientation constraints: "the wardrobe is on right of the couch".

JACADI will only accept this four kind of constraints.

2.3 Space planning problem

In this paper we focus on the "sofa problem" introduced by Howden in 1968. Given a bounded planning space (a lounge), objects (furniture), and a set of constraints on these objects ("the chair is in front of the table", "the wardrobe is against the wall", "the light is on the table", etc.) we have to find location and size of objects into the 3D space in order to satisfy constraints. Such a problem is called *space planning problem* [Eas73]. Note that this term also refers to space partitioning, splitting problems, etc. Seeing space planning as a constraint satisfaction problem is very interesting because of the possibility to use numerous techniques developed to solve CSP [BF91]. We can notice two major approaches to compute solutions: complete techniques and stochastic methods. In this paper we only consider complete methods based on constraint graph filtering and backtracking techniques [Dec99]. The main difficulty arises from NP-Complete complexity of algorithms. Another difficulty comes from dynamic characteristics of the problem. To get more interactivity, the solver has to minimise new computations when the designer adds, removes or modifies a constraint.
See [Cha95] for a complete overview about space planning problem modelling and solving.

2.4 DEM²ONS: system overview

The following synoptic (figure 2) presents the future version of our declarative system DEM²ONS. The core will be the constraint solver JACADI presented in this paper. This solver will have to allow 3D-objects positioning according to a geometric and weighted constraint system. Currently we only work on solving techniques that ensures completeness. In future works we hope to extend our solver by integrating stochastic algorithms (like evolutionist techniques) to propose time-efficient solutions in case of scenes with numerous objects.

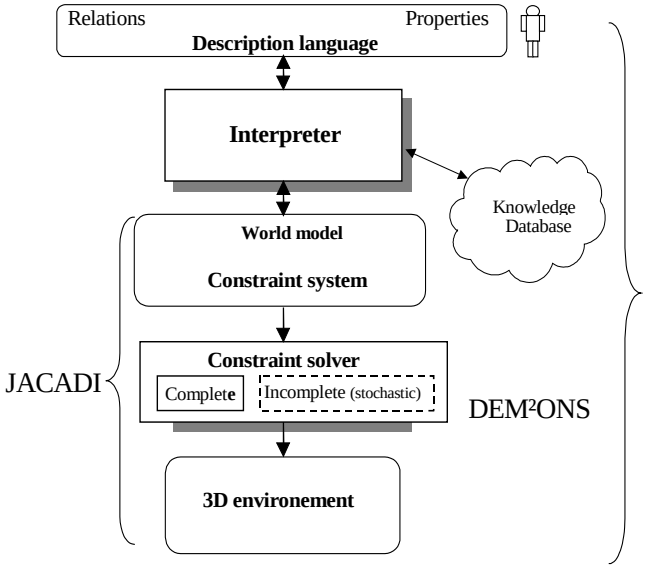

Figure 2: DEM²ONS synoptic

3 Fundamental concepts and analysis of solvers ORANOS and EAAS

3.1 Fundamental notions

Because of JACADI is based on a CSP extension, this chapter briefly discusses about CSP and important associated concepts.

A CSP [Mac77] P = (V, D, C) is defined by:

- a set of variables V = { v_1,…v_n }
- a set of domains D = { D_1,…D_n}, where D_i is the domain associated with v_i.
- a set of constraints C = { C_1,… C_n } where C_i is defined by any relation linking a set of variables.

A solution of a CSP, P = (V, D, C) is an assignment of values from D to the variables of V such that all the constraints of C are satisfied.
Solving a CSP is a NP-hard problem. One approach that has proved its worth is to try to reach partial consistency. A partial consistency technique is a polynomial filtering algorithm that transforms a CSP in a simpler one. To find a solution one approach consists to explore search space using a backtracking algorithm. This backtracking can be improved using:

- look-ahead schemes: a filtering algorithm is applied during backtracking to prune search tree. Forward checking (FC) or real full-look-ahead are the most common algorithms.
- look-back schemes: these algorithms (backmarking, conflict directed backjumping, etc.) uses learning techniques to avoid failures.

For a complete description of all these techniques see [Dec99]. Another way to solve CSP is stochastic algorithms like tabu search or simulated annealing [HP98].

A numeric CSP (NCSP) is a kind of CSP where variables are numeric. Domains are sets of numeric values and constraints are defined by numeric relations [Lho93].

A dynamic CSP (DCSP) [Dec88, SV93] is a sequence of CSPs each resulting from a change in the preceding one. The change may be a relaxation (a constraint is removed) or a restriction (a constraint is added). It is necessary in an interactive application. To solve a DCSP, the strategy consists to try to reuse previous results in order to reduce new computation.

In this study we consider CSP with weighted constraints [Bor92]. Such problems are called hierarchical CSP (HCSP). In interactive applications, hierarchical aspect is used to avoid deadlock. If the problem is inconsistent, the system gives a solution by satisfying constraints with higher priority first.

We now present two solvers used to design JACADI. Advantages and drawbacks of these solvers are briefly discussed relatively to our requirements.

3.2 ORANOS [Kwa98]

Works carried out by G. Kwaiter [KGC97a, KGC98a, and KGC98b] have shown interest and power of constraint solvers for space planning. The ORANOS solver based on DHNCSP[8] formalism is able to solve the "sofa problem" under some particular conditions. Presently ORANOS in unable to give a time-efficient solution when objects are non-isothetic: it is a direct consequence of the chosen domain representation and use of interval algebra [Sny92].

Kwaiter defines a DHNSCP as follows:
A DHNCSP P = (V, D, C_p) is defined by
- a set of numeric variables V = { v_1,…v_n }
- a set of continuous domains D = { D_1,…D_n}, where D_i is a numeric interval.
- a set of weighted-constraints C = { $C_{1,p1}$,… $C_{n,pn}$ } where $C_{i,p}$ is defined by any numeric relation linking a set of variables and where p is the priority associated with the constraint.

To solve such a dynamic problem he uses an incremental filtering algorithm based on interval algebra (arc B-consistency [Lho93]) combined with an improved backtracking (forward-checking).

Advantages:
– Hierarchical constraint management [KGC98a][Bor92]: in order to deal with over-constrained problems. The system can propose objects location even when user's description is inconsistent.
– Dynamic constraint solver [KGC98a] [Dec88]: due to an incremental approach, ORANOS can re-satisfy dynamically the system when a change occurs in the set of constraints. It's not necessary to restart from the beginning. This capability is very useful to improve solver interactivity
– Able to handle cyclic constraint graphs.

Drawbacks:
– Unable to efficiently handle non-isothetic objects because arc B-consistency [Lho93] preserves domain convexity. The filtering method is not adapted to not connected graphs and disjunctive constraints because after filtering numerous inconsistent areas remain. (see figure 3)
– Scene coherency can't be guaranteed between two successive solutions.

ORANOS is in fact an all-purpose dynamic and hierarchical constraint solver that can be used for varied applications. But it is not efficient for planning problems that contains only geometric

[8] Dynamic Hierarchical Numeric Constraint Satisfaction Problems

constraints, especially if there are non-isothetic objects to position.

3.3 EAAS [Cha94, Cha95]

EAAS[9] is a floor plan design tool designed by P. Charman (SECOIA project). He has proposed a CSP extension called SCSP (for Spatial CSP). SCSP deals with geometric characteristics of space planning problems.

Charman defines a SCSP as a triple (O,G,C):

CSP	SCSP
V: set of variables D: set of domains C: set of onstraints	O : set of physic objects G : set of configuration C : set of geometric constraints

Chart 1: CSP vs SCSP

He demonstrates that standard arc-consistency algorithms are inefficient and suggests another partial consistency technique: semi-geometric arc-consistency inspired from AC-3 [Mac77]. To solve a SCSP he combined this algorithm with a backtracking technique and shows that FC is better than RFL.

Advantages:
- Solver suitable for 2D space planning.
- Efficient for geometric constraint management because of the use of SCSP model.
- Good management of non-connected graphs due to explicit presence of non-overlapping constraints. It results from semi-geometric arc-consistency filtering and multidimensional representation of domains. (see figure 3)

Drawbacks:
- SCSP formalism is not incompatible with non-isothetic objects but taking such objects into account is a non-trivial task.
- Not suitable to an interactive environment: over-constrained systems and incremental resolution are not considered.
- Unable to deal with cyclic constraint graphs.

[9] EAAS : Environnement d'Aide à l'Aménagement Spatial (Space planning assisting environment)

3.4 Conclusion

Figure 3 compares ORANOS and EAAS and shows that a multidimensional representation of domains is more suitable for planning problem in order to reduce searching space.

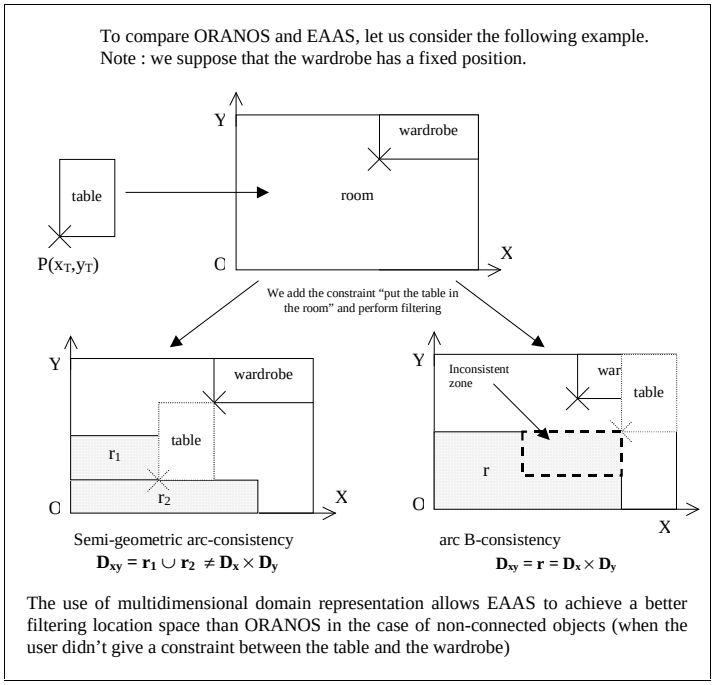

Figure 3: comparison between ORANOS and EAAS

The two solutions presented before have complementary features. The design of JACADI makes use of the most interesting characteristics of these two solvers.

4 Introduction to DHSCSP and base of our constraint solver JACADI

Our goal is to design a solver adapted to interactive graphical applications and efficient for non-isothetic space planning problems. JACADI must implement a filtering method to limit backtracking as possible.

Convention
In the next parts, $\Re$ represents real numbers and $\Re^*$ represents float numbers ($\Re^* \subset \Re$).

4.1 DHSCSP

Previous chapter presents two interesting CSP extensions: DHNCSP [Kwa98] and SCSP [Cha95].

- DHNSCP: dynamic and hierarchical concepts developed in this formalism are essentials in an interactive application. Nevertheless numerical approach does not seem to be the best choice because of the problem nature.
- SCSP: fully designed for space planning, this model works only with static CSP and does not handle over-constrained systems.

That's why, we suggest a new extension of CSP called **DHSCSP**[10]**.** Formally a DHSCSP is defined as follows:

$$\boldsymbol{P = (\Theta, \Delta, \Gamma_i)}$$

- $\Theta = \{\Theta_1, \ldots, \Theta_n\}$ is the set of objects to be positioned. Each object is characterised by a triple (O,P,S) where O is the object's orientation, P is the position of the object's *reference point*, and S is the object's size.
- $\Delta = \{\Delta_{\Theta 1}, \ldots \Delta_{\Theta n}\}$ is the set of configurations. We call object's *configuration* $\Delta_{\Theta i}$, the set of possible positions, orientations, and sizes for one object Θi. An object's configuration is characterised by a triple $(\Delta o_i, \Delta p_i, \Delta s_i)$ where Δo_i is the domain of the orientation parameter, Δp_i is the domain of the position parameter, and Δs_i is the set of possible sizes.
- $\Gamma_p = \{\Gamma_{1,p1,\ldots}, \Gamma_{m,pm}\}$ is a hierarchical set of constraints where $\Gamma_{i,pi}$ is a geometric spatial relation applied on a subset of Θ, i represents the number and p_i the priority of the constraint.

As Kwaiter's DHNCSP and unlike classical CSP, all the constraints must not be consistent to find a solution for a DHSCSP. In fact, constraints are sorted according to their priority (set by user).
So, when the Γ set is over-constrained, it is not necessary to satisfy all the set of constraints. In this case the solver let the constraints with lower priority unsatisfied in order to satisfy the strongest ones. To take into account dynamic and hierarchical aspects in our solver, we choose to use algorithms that Kwaiter has developed for ORANOS [Kwa98].
In the following parts we give a representation for entities used in our solver.

[10] for Dynamic Hierarchical Spatial CSP

4.2 Objects representation (see chart 2)

JACADI is designed for a 3D modelling application. The handled objects are also in 3D. Because of complexity of algorithms, we have chosen to only handle the bounding boxes of the objects. To offer further flexibility during the description step, a more precise representation with hierarchical bounding boxes might be used. Every bounding box is associated with three parameters (O,P,S) :

Parameter	Meaning	Representation
O	Orientation of the bounding box in the 3D space	Using Euler 3 angles $O=(\alpha_X, \alpha_Y, \alpha_Z)$
P	Position of the reference point in the scene reference	Using a translation vector $P=(P_X, P_Y, P_Z)$
S	Size of the bounding box along the 3 axes of the object local reference (given by O and P)	Using a scale vector $S=(S_X, S_Y, S_Z)$

Chart 2: Representation of a 3D object in JACADI

An object is entirely determined when the nine parameters have a value.

4.3 Domains representation (see chart 3)

In a DHSCSP we have seen that variables are 3D objects given by three parameters (O,P,S). Every object has its domain $\Delta = (\Delta_O, \Delta_P, \Delta_S)$. The following filtering and resolution techniques depend on the implementation chosen for Δ_O, Δ_P, and Δ_S. We propose the following choices:

Variables	Theoretical domains	Domains representation
Orientation $O = (\alpha_X, \alpha_Y, \alpha_Z)$	$\Delta_O \subset ([0;360[)^3$	3 discretized domains 1D: $\Delta_{\alpha x}, \Delta_{\alpha y}, \Delta_{\alpha z}$, where $\Delta_{\alpha I} \subset [0;360[/ \Pi_i$ where Π_i is the α_i discretization step.
Position $P = (P_X, P_Y, P_Z)$	The modelling space is continuous, so : $\Delta_P \subset \Re^3$	A 3D pseudo-continuous domain : $\Delta_P \subset (\Re^*)^3$ $\Leftrightarrow$ convex polyhedrons union.
Size $S = (S_X, S_Y, S_Z)$	$\Delta_S \subset \Re^3$	3 pseudo-continuous intervals : $\Delta_{Sx}, \Delta_{Sy}, \Delta_{Sz}$, where $\Delta_{Si} = I \subset \Re^*$

Chart 3: Domains representation for a 3D object

4.4 Constraints representation

As in ORANOS, we use a two-level constraint system. *Primitive constraints* are the basic elements used to build high level constraints called *symbolic constraints*. We choose to take into account the four kinds of geometric primitive constraints presented in chart 4.

Building a complete primitive constraint system and analysing symbolic constraint decomposition in primitive constraints is a complex linguistic task that we don't consider in this study (what exactly mean spatial prepositions «on », « to the right », « to the left », « before », « against », « between», etc.).

Primitive Constraints	Meaning	Parameters		
		P	O	T
Non-overlapping(*object$_1$*,*object$_2$*)	Means that two objects must never overlap. It's the default constraint between to objects that are not connected explicitly.	×	F	F
Area Constraints: **Area**(*object$_1$*, *object$_2$*,, *Boundaries_list*)	The position of the object$_1$ is given relatively to the object$_2$. To describe this position a set of oriented boundaries are used: *boundaries_list*. Those boundaries are defined from object$_2$. Area *primitive constraints* are used to implement metric and topologic constraints.	×	F	F
Size Constraint: **Size**(*object$_1$*, …, *object$_n$*, *equation/inequation*)	Relations between the sizes of a group of objects are given using numerical equations or inequations. (« the table is twice longer than the chair »)			×
Orientation Constraints: (1) **Orientation**(*object$_1$*, *object$_2$*, *list*) (2) **Orientation**(*object*, *list*)	The relation (1) expresses relative orientations of two objects (« the table is in front of the window »). As we have chosen to sample the possible orientations, the *list* parameter contains the set of allowed values for this relation. The relation (2) constrains the orientation for only one object (« according to the scene reference, the table is oriented with $\alpha_x = 45°$ »)		×	

P = Position parameter; O = Orientation parameter; T = Size parameter;
× = Parameter concerned by the reduction method of the primitive constraint associated domains.
F (for fixed) = Indicates that this parameter must be fixed to apply the filtering algorithm.

Chart 4 : the four kinds of geometric primitive constraints that will be accepted by JACADI.

4.5 Boundary definition

Boundaries are used to build the area where an object can be positioned. A *boundary* is a frontier used to split the working space in two parts. We choose to implement a boundary by a linear equation (a straight line in a 2D space, a plane in a 3D space, etc.). A plane splits the space in two half-spaces. Let *B* be a boundary defined in a space E. We note B^+ the positive half-space and B^- the negative half-space.

Example: when E is the 3D space ($\Re^3$), we get:

B : f(P) = a.x + b.y + c.z + d = 0
where (a, b, c, d) $\in \Re^4$, with
$B^+ \equiv \{ (x, y, z) \mid f(P) >= 0 \}$and
$B^- \equiv \{ (x, y, z) \mid f(P) < 0 \}$ = the complementary of B^+ in E.

A convex polyhedral area (D_{convex}) can be delimited by the intersection of half-spaces (a set of oriented boundaries). This representation is equivalent to a linear equations system:

$$\mathbf{D_{convex} = \bigcap B_i^{+/-}} \quad \mathbf{i=1..n}$$

To give the area where an object can be located we have to:

- Define a set of planes (set of boundaries).
- Specify for every boundary whether the object is located at the positive or negative side.

Some boundaries are fixed: for example walls of a room. Others boundaries are given according to constrained objects (see figure 4).

4.6 Primitive constraints

The different kinds of primitive constraints accepted by our solver are specified in chart 4. Primitive constraint definition arises from domain modelling given before.

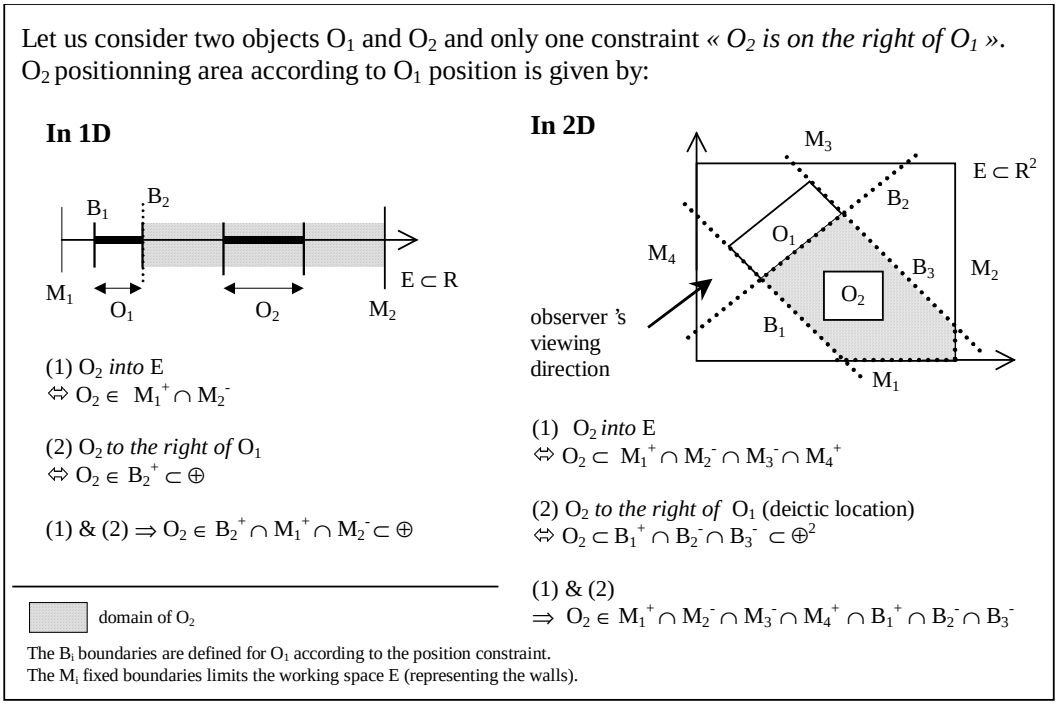

Figure 4: illustration of boundaries definition

4.7 Symbolic constraints

Symbolic constraints are used to implement a high level relation between objects (examples "the book is on the table", "the sofa is against the wall",..). A symbolic constraint has generally the next structure:

"(object$_{target}$) *spatial preposition* (object$_{landmark}$)"

- The target object is the object concerned and modified by the constraint.
- The landmark object is the reference object of the constraint. It is used in order to define boundaries that constrain target object position.

Notes: a symbolic constraint is expressed as a conjunction of primitive constraints.

4.8 Working space

The working space is a finite and connected part of the 3D Euclidean space. It represents the area to fit out (for example a lounge or a bedroom in a house). It is bounded by a set of boundaries. In our approach, we only consider convex polyhedral working spaces (see figure 5).

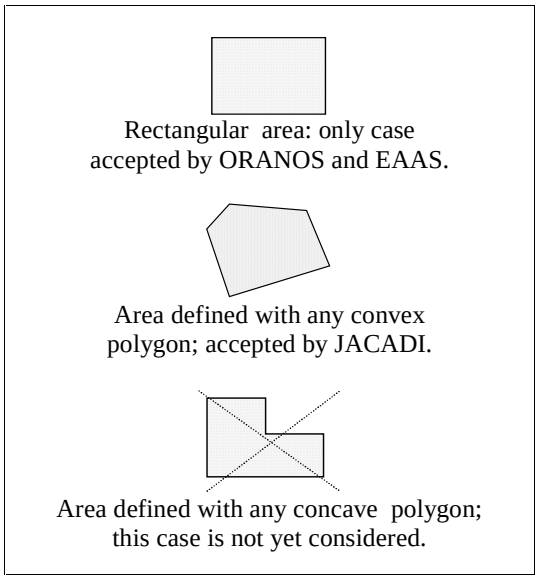

Figure 5: Examples of positioning area in 2D

5 Filtering of the constraint graph

Filtering the constraint system consists in reducing the positioning area of every object by removing most of inconsistent configurations. This step is essential, because the efficiency of the resolution algorithm depends on the efficiency of filtering. In our approach, we separate filtering of three kinds of parameters (orientation, position and size).

– To filter an orientation constraint graph, we have chosen to discretize the orientation parameters: the task is then equivalent to a classical CSP filtering. An arc-consistency algorithm is used [Dec99, Mac77].

– The size of an object is characterised by three parameters. The domain of such parameter is an interval in $\Re^*$. This problem is a NCSP: we use arc B-consistency algorithms [Lho93].

– To filter the last kind of parameters we need to develop new algorithms. We use two kinds of primitive constraints concerning position parameters: a*rea constraints* and *non-overlapping constraints* (chart 4). We thereafter suggest two algorithms to reduce domain of position parameters according to applied constraints. Note that we have chosen a multidimensional representation of the domain of position parameters (union of convex polyhedra). This choice allows a more precise definition of positioning area of objects. So possible collisions can be detected sooner during the filtering phase.

Reduction algorithm in case of non-overlapping constraint:

Function NOReduction (C : NonOverlappingConstraintPrimitive)

Preconditions: *C is a non-overlapping constraint between two objects Θ_1 and Θ_2; Size and orientation of Θ_1 and Θ_2 must be fixed.*

Postconditions: *Returns Inconsistency if the positioning domain of Θ_1 ($\Delta_p(\Theta_1)$) is empty (means that C is inconsistent) else returns true after domain reduction of Θ_1 according to Θ_2.*

Note: *$\Delta_p(\Theta)$ is a list of convex polyhedra*

Begin
  If $\Delta_p(\Theta_1) = \emptyset$ **return** Inconsistency
  Else
    $\Delta'_p(\Theta_1) = \emptyset$
    For every *polyhedron* $\Pi \in \Delta_p(\Theta_2)$
      $\pi_{min} \leftarrow$ intersection of all positions of Θ_2 when its reference point is moved into Π
      If $\pi_{min} = \emptyset$ **return** true
      Else
      For every *polyhedron* $\pi \in \Delta_p(\Theta_1)$
        $\pi_{reduc} \leftarrow$ area of π obtained when the reference point of Θ_1 is moved into π and when Θ_1 collides with π_{min}
        $\pi_{reduit} \leftarrow \pi - (\pi \cap \pi_{reduc})$
      If $\pi_{reduit} = \emptyset$, remove π from $\Delta_p(\Theta_1)$
      Else
        $L_\pi \leftarrow$ Splitting of π_{reduit} to give a list of convex polyhedra
        $\Delta'_p(\Theta_1) = \Delta'_p(\Theta_1) \cup L_\pi$
    If $\Delta_p(\Theta_1) = \emptyset$ **return** Inconsistency
    Else
      $\Delta_p(\Theta_1) = \Delta'_p(\Theta_1)$
      return true
End

Filtering algorithm in case of an Area Constraint:

Function AReduction (C : AreaConstraintPrimitive)
; reduces the positioning area of the target object according to the landmark object

Preconditions: *C is an area primitive constraint. Its parameters are:*
- *Θ_1 = target object;*
- *Θ_2 = landmark object;*
- *L_B = a list of oriented boundaries related to Θ_2.*

C gives the relative position of Θ_1 according to Θ_2.

Postconditions: *Returns Inconsistency if the position domain of Θ_1 ($\Delta_p(\Theta_1)$) is empty (means that C is inconsistent), else returns true after reduction of ($\Delta_p(\Theta_1)$ according to C.*

Note: *Let the object Θ.*
. $\Delta_p(\Theta)$ is a list of convex polyhedra used to describe the positioning area of the reference point of Θ.
. A boundary is a plane that splits the space in two parts.

Notation: *(B,i) with i=1 where –1 represents the oriented boundary $B^{+/-}$: (B,1) = B^+ and (B,-1) = B^-.*

Begin
  STEP 1: Computation of the effective boundaries related to Θ_1
    $L_B' = \emptyset$
    For every oriented boundary (B,i) $\in L_B$
      B' $\leftarrow$ plane parallel with B going by the reference point of Θ_1, when $\Theta_1 \in$ (B,i) and reaches B
      $L_B' \leftarrow L_B' \cup$ (B',i)
    The list L_B' is defined related to Θ_2.
    Given a position e ($\in \Delta_p(\Theta_2)$) of Θ_2, L_B' delimits a space area of: R_e.

  STEP 2: reduction of $\Delta_p(\Theta_1)$ *according to C*
    If $\Delta_p(\Theta_1) = \emptyset$ **return** Inconsistency
    If $R_e = \emptyset$ (for any $e \in \Delta_p(\Theta_2)$ R_e)
      return Inconsistency
    Else
      $Z = \emptyset$
    For every *polyhedron* $\Pi \in \Delta_p(\Theta_2)$
      $\pi_{allowed} \leftarrow \cup R_i$ with $i \in \Pi$
      $Z = Z \cup \pi_{allowed}$
    $\Delta_p'(\Theta_1) = \emptyset$
    For every polyhedron $\pi \in \Delta_p(\Theta_1)$
      $\pi_{reduit} \leftarrow \pi \cap Z$
      Remove π from $\Delta_p(\Theta_1)$
      If $\pi_{reduit} = \emptyset$ **Then** nope
      If π_{reduit} is a convex polyhedron **then**
        add π_{reduit} to $\Delta_p'(\Theta_1)$
      else
        $L_\pi \leftarrow$ Splitting of π_{reduit} into a list of convex polyhedra
        $\Delta_p'(\Theta_1) = \Delta_p'(\Theta_1) \cup L_\pi$
    $\Delta_p(\Theta_1) = \Delta_p'(\Theta_1)$
    If $\Delta_p(\Theta_1) = \emptyset$ **return** Inconsistency
    Else return true
End

The two following pages (figure 6 and 7) present illustration of these filtering algorithms.

The AReduction algorithm treats the constraint “the chair is in front of the table”. This algorithm computes the new positioning area of the chair reference point to ensure that all positions for the table and the chair match.

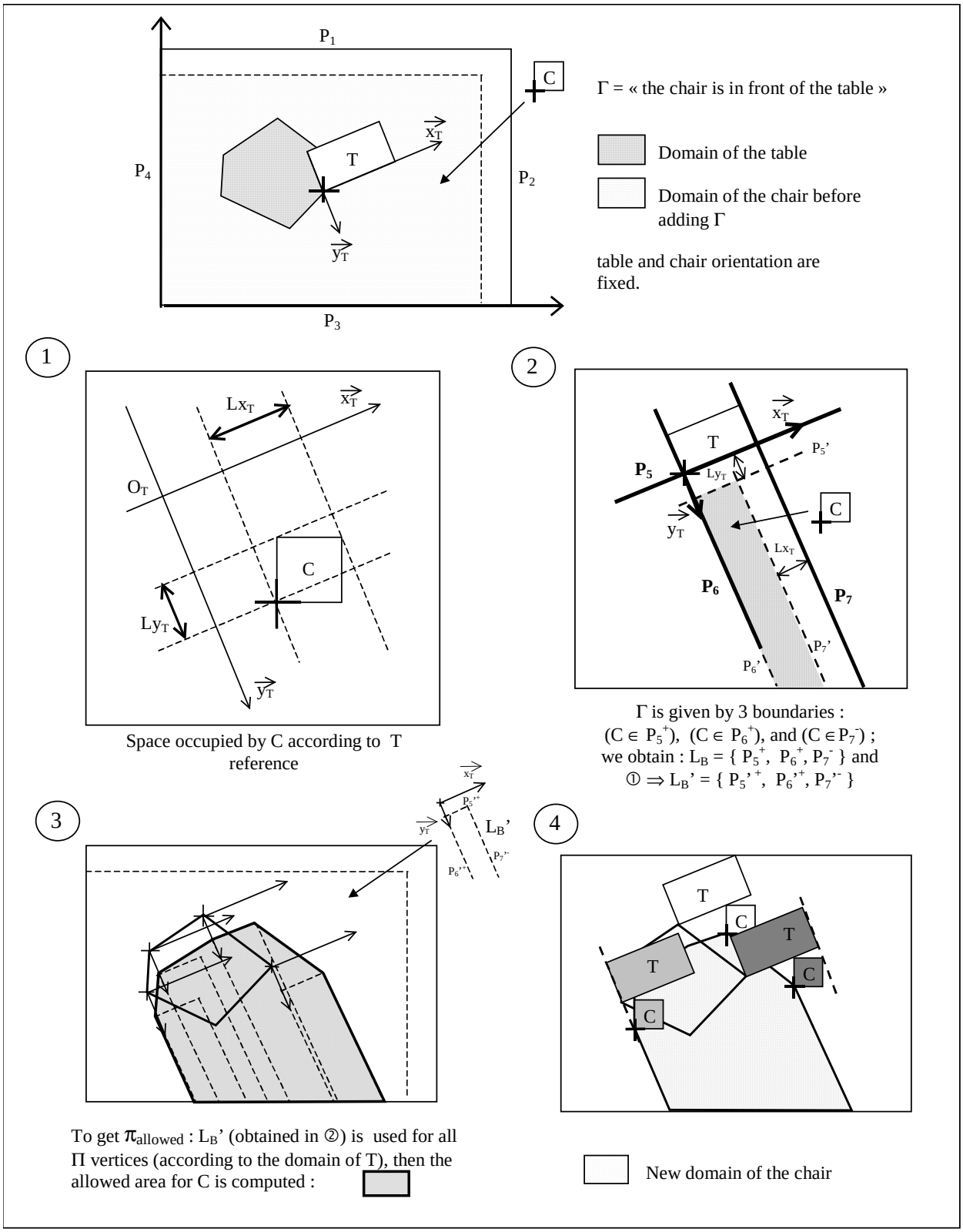

Figure 6: Illustration of *AReduction* algorithm

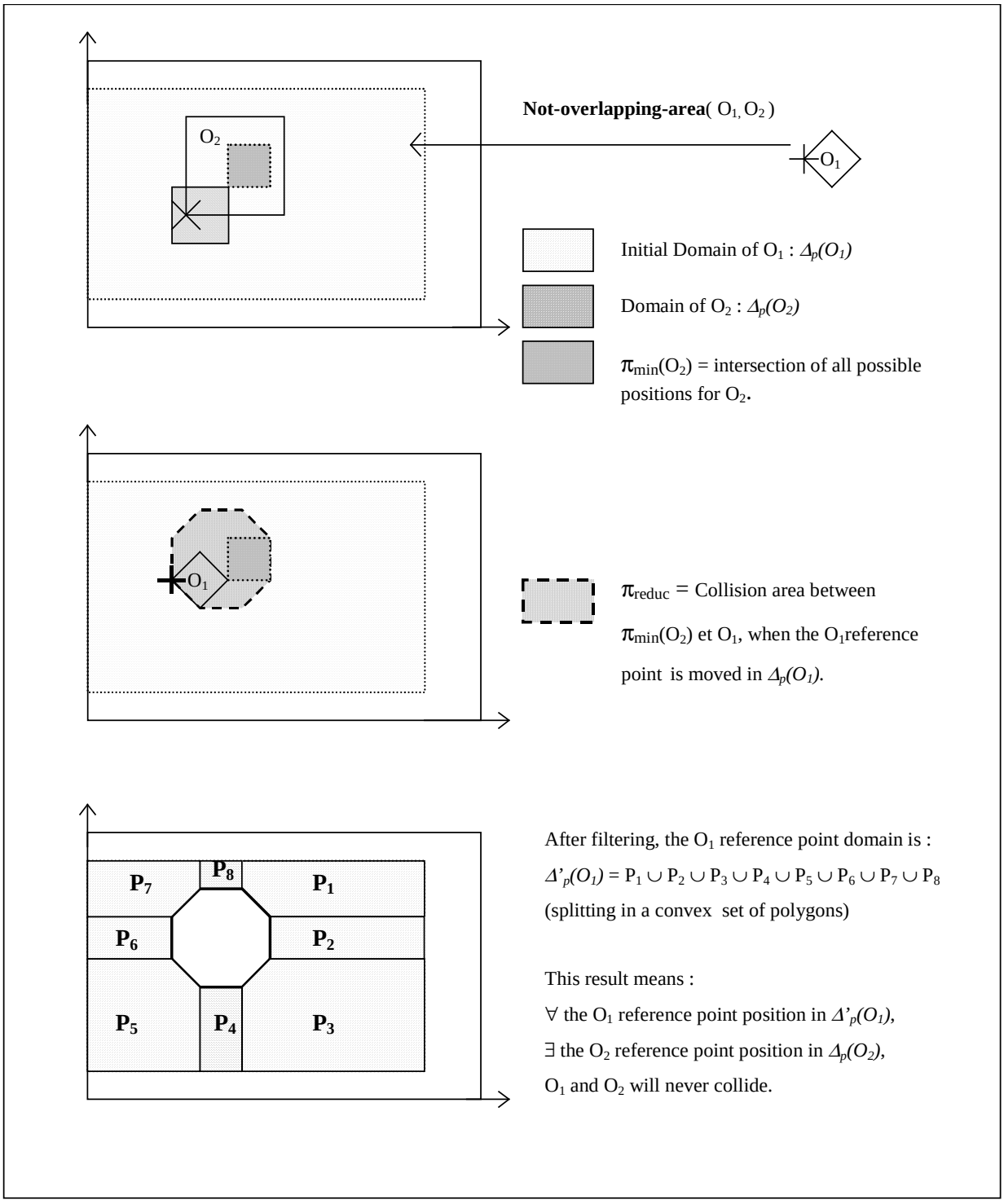

Figure 7: Illustration of *NOReduction* algorithm

5.1 Solving strategy

The aim of JACADI is to solve a hierarchical and dynamic 3D geometric constraint system.
Each constraint has an associated weight given by the user during description phase. This weight corresponds to constraint priority and is used by solver to determine which constraints must be satisfied first.
For dynamic and hierarchical aspects of the constraint system we use the same algorithms as ORANOS (refer to [Kwa98] for complete algorithms). As said before, we handle every related parameter (orientation, position and size) with one appropriated algorithm. In the first implementation of JACADI we plan to use the following strategy:

- **STEP 1** : Orientation processing
  - All the orientation constraints are collected to form a classical CSP. Remember that orientation parameters are discretized.
  - A filtering algorithm (arc-consistency [Mac77]) is applied on the obtained CSP.
  - An improved backtracking with forward checking (see [Dec99]) is computed to set objects' orientation.
- **STEP 2** : Size processing
  - All the size constraints are collected to form a numeric CSP.
  - A filtering algorithm is applied (arc B-consistency [Lho93]) on the obtained NCSP.
  - As for step 1, an improved backtracking with a look-ahead technique (FC or RFL) is performed to set objects' size.
- **STEP 3** : Position processing
  - In this step we collect all others constraints to form a spatial CSP.
  - Knowing size and orientation of objects, we use domain reduction algorithms presented in the previous part (figure 6, 7) to determine where objects can be placed. We obtain for each object a convex polyhedra list that describes potential positions for this object.
  - Finally, to produce a scene, resulting polyhedral domains are sampled into a set of positions. As for step 1 and 2, an improved backtracking algorithm with a look-ahead technique is used to test these sets of location in order to determine a valid position for each object.

Notes:
- Inconsistency of the constraint graph can be detected at each step.
- The solver stops when he finds a solution. The 3D-scene is then built and presented. If the user wishes it, the solver can produce another scene. Because of exhaustive search techniques employed, JACADI can theoretically give all scenes that match user's description.
- To manipulate polyhedra, we project to use algorithms from a library called Polylib [Wil93]. This tool operates on objects called domains made of unions of polyhedra. It allows boolean operations (intersection, difference, etc.) on these objects.

6 Conclusion

To design our constraint solver JACADI, we present a model able to formalise our problem: non-isothetic 3D-objects positioning using geometric constraints. This model, called DHSCSP is an extended form of CSP, issued from hybridisation of SCSP and DHNCSP formalisms.

The main difficulty to solve a DHSCSP comes from the huge complexity of the problem. To limit the search space we have proposed an adapted representation of domains combined with a powerful filtering technique of position parameters.

We are currently programming this new solver using both dynamic and hierarchical ORANOS algorithms and new ideas presented in this article. A lot of improvements will be necessary to get an efficient and powerful tool. In particular, we project to study:
- Constraint graph pre-processing: by using hierarchical description of the scene, we think that most of constraint graphs could be split in a set of disconnected parts. Each part could be solved by an independent thread. In this case multiprocessing or distributed architectures could be used.
- Improvement of filtering techniques: currently we don't know how to filter simultaneously orientation and position parameters.

- Determining exact complexity of whole solving algorithm.
- Benching our solver to know maximum problem size tractable using this approach. How many objects JACADI will be able to handle?
- Testing evolutionist strategies (stochastic techniques).

References

[Bor92] A. Borning, B. Freeman-Benson, M.Wilson. Constraint hierarchies. Lisp and Functional Programming. Vol. 5:pages 223-270, 1992.

[BF91] C. Baykan, M. Fox. Constraint satisfaction techniques for spatial planning. In Intelligent CAD Systems III - Pratical experience and Evaluation, pages 187-204, 1991.

[Cha94] P. Charman. A constraint-based approach for the generation of floor plans. In 6th IEEE International Conference Tools with Artificial Intelligence, November 1994.

[Cha95] P. Charman. Gestion des contraintes géométriques pour l'aide à l'aménagement spatial. Thèse de Doctorat, Ecole nationale des Ponts et Chaussées – Novembre 1995.

[CDMM97] C. Colin, E. Desmontils, J. Martin, J.P. Mounier. Working Modes with a Declarative Modeller. In proceedings of the Compugraphics'97, Portugal, December 1997.

[Dec88]. R. Dechter, A. Dechter. Belief maintenance in dynamic constraint networks. In proceedings of the 6th National conference on Artificial Intelligence (AAAI 88), St. Paul, MN, pages 337-342, 1988.

[Dec99] R. Dechter, D. Frost. Backtracking algorithms for constraint satisfaction problems – An ICS technical report, September 1999.

[DH93] S. Donikian, G. Hégron. A Declarative Design Method for 3D Scene Sketch Modeling. In Eurographics'93, Vol.12 (33):pages 223-236, 1993.

[Eas73] C. Eastman. Automated space planning. Artificial intelligence, Vol. 4, pages 41-65, 1973.

[GCR93] V. Gaildrat, R. Caubet, F. Rubio. Declarative scene modelling with dynamic links and decision rules distributed among the objects. International Conference on Computer Graphics, ICCG 93, Bombay, February 1993.

[HP98] J-K Hao, J. Pannier. Simulated annealing and Tabu search for constraint solving. 5th international symposium on IA and mathematics (AIM'98). Fort Lauderdale, Florida, USA, January 1998.

[KGC97] G. Kwaiter, V. Gaildrat, R. Caubet. DEM²ONS: A High Level Declarative Modeller for 3D Graphics Applications. In Proceedings of the International Conference on Imaging Science Systems and Technology, CISST'97, pages 149-154, Las Vegas, June 30-July 3, 1997.

[KGC98a] G. Kwaiter, V. Gaildrat, R. Caubet. Controlling Objects Natural Behaviours with 3D Declarative Modeller. In Proceeding of Computer Graphics International, CGI'98, Hanover, Germany, pages 248-253, 24-26 June 1998.

[KGC98b] G. Kwaiter, V. Gaildrat, R. Caubet. Modelling with Constraints : A Bibliographical Survey. In Proceeding of International Conference on Information Visualisation, IV'98, London UK, 29-31 July, 1998.

[Kwa98] G. Kwaiter. Modélisation déclarative de scènes : étude et réalisation de solveurs de contraintes. Thèse de Doctorat, Université Paul Sabatier – Toulouse, Décembre 1998.

[Lho93] O. Lhomme. Consistency techniques for numeric CSPs. In Proceedings of the 13th International Joint Conference On Artificial Intelligence (IJCAI-93), pages 232-238, 1993.

[Mac77] Mackworth A. K. Consistency in Networks of Relations. Artificial intelligence, Vol. 8 (1):99-118, 1977.

[PRT98] D. Plemenos, W. Ruchaud, K. Tamine, Iterative techniques for declarative modelling, International Conference 3IA'98, Limoges (France), April 1998.

[SP96] D. Sellinger, D. Plemenos; Integrated Geometric and Declarative Modeling Using Cooperative Computer-Aided Design. 3IA'96, Limoges (France), April 1996.

[Sny92] John M. Synder. Interval Analysis for Computer Graphics. Siggraph'92. Computer Graphics, 26, 2 July 1992.

[SV93] T. Schiex et G. Verfaillie. Two Approaches to the Solution Maintenance Problem in Dynamic Constraint Satisfaction Problems. Proc. of the IJCAI-93/SIGMAN Workshop on Knowledge-based Production Planning, Scheduling and Control, p.305-316, Chambery, France, August 1993.

[Wil93]. D. K. Wilde. A library for doing polyhedral operations - IRISA - PI 786 - December 1993. Note: the rapport and the library are available on ftp server: ftp.irisa.fr/local/API/polylib

[Woo87] R. Woodbury. The Knowledge Based Representation and Manipulation of Geometry. Ph. D. Thesis, Carnegie-Mellon University, 1987.