Toward a consistent spatial model

Toward a Consistent Spatial Model

Vers un modèle spatial cohérent

Auf dem Weg zu einem konsistenten Raummodell

Xiao TANG
Computer Scientist
Univ. of Paris
Paris, France

Xiao Tang, born 1963, received her informatics engineering degree at the University of Chong Qing, China. She obtained her doctor's degree at the University of Paris VI. Since four years, Xiao Tang has studied the problem of consistency.

Khaldoun ZREIK
Prof.
University of Caen
Caen, France

Khaldoun Zreik is professor at the University of Caen, France, and researcher in Artificial Intelligence in design at Lutèce and LAIAC laboratory.

SUMMARY
This paper presents a constraint-based consistency maintenance system for a spatial model simulating a building environment perceived by an Indoor Autonomous Mobile Robot. The constraints define a reference frame for the environment model, representing the fundamental properties concerning the characteristics of spatial entities, topo-geometrical or semantic relations between them. The environment model will be declared consistent if and only if all the constraints are satisfied.

RÉSUMÉ
Un système de maintien de la cohérence est présenté pour un modèle de l'environnement spatial simulé où naviguent des robots mobiles autonomes. Ce système est basé sur des contraintes qui forment une référence de cohérence. Ce sont des propriétés concernant les relations géométriques, topologiques et sémantiques existantes entre les entités spatiales. Le modèle de l'environnement est dit cohérent si, et seulement si toutes ces contraintes sont satisfaites.

ZUSAMMENFASSUNG
Der Beitrag stellt ein auf Nebenbedingungen beruhendes System vor, das die Konsistenz in die Simulierung eines Raummodelles wahrt, wie es von einem autonomen mobilen Roboter für Einsätze im geschlossenen Räumen wahrgenommen wird. Die Nebenbedingungen definieren einen Bezugsrahmen, der die unabdingbaren Eigenschaften räumlicher Grössen und ihren topologischen und semantischen Beziehungen repräsentiert. Das Raummodell wird nur dann als konsistent bezeichnet, wenn alle Nebenbedingungen eingehalten sind.

1. INTRODUCTION

The problem of consistency maintenance in Artificial Intelligence is not a new one. In the domain of knowledge-based systems, the consistency maintenance is a fundamental issue [1] ; in the domain of CSP (Constraints satisfaction problem) [2], the process of solving consists in searching a solution with which all the constraints are consistent ; etc. The notion of consistency depends in fact on the domain in consideration.

The consistency maintenance in our context concerns a model of a simulated spatial environment for IAMRs.

An IAMR can perform intelligent motion planned [7]. The planning of robot's trajectory and the pattern recognition during the motion depend directly on the representation model of its spatial environment. The model must provide necessary information about the real world at different levels : geometric, topological, semantic, etc.

A spatial environment can not always be completely and precisely described, and the change in the environment perceived by an IAMR may be falsified (by the noise of its vision sensors for example). For this raison, the consistency maintenance of the environment model is necessary.

In order to control the environment model's consistency, we develop a consistency maintenance system PROVE which possesses a referential truth model built on three types of constraints : constraints-to-be-propagated (Cp), constraints-to-be-verified (Cv), and constraints-to-be-verified-and-propagated (Cvp). Each constraint is a inviolable numerical or symbolical relation existing between spatial entities such as "each space must have an opening which make it accessible (directly or non) from the exterior".

Two cases are checked. Firstly the initial model will be verified before supplied to the robot : we verify the structure of each spatial object, the relation between different objets, etc.

Secondly, any change in the environment will be controlled by PROVE system : it will be propagated throughout the model by using the constraints to be propagated. If the change and its induced result satisfy all the constraints to be verified, it is considered as consistent and the environment model may be updated. Otherwise, the change will be refused and an analysis procedure, based on a mechanism inspired from ATMS (Assumption-based Truth Maintenance Systems) [4], will be engaged to detect the origins of the anomaly.

2. ENVIRONMENT MODELLING

The IAMR's environment considered is a built space represented on two dimensions and half (as illustrated in the figure 1), which can be decomposed into several levels : <Environment> -> <Spaces, Walls, Openings> -> <Facets> -> <Ridges> -> <Points>. If we consider facets and openings as elementary entities, the structure hierarchy of the environment showed in figure 1 can be represented by the figure 2. Thus several levels of information must be taken into account : numerical at low level to represent the geometry of spatial entities ; symbolical at high level to represent the topology of spatial entities.

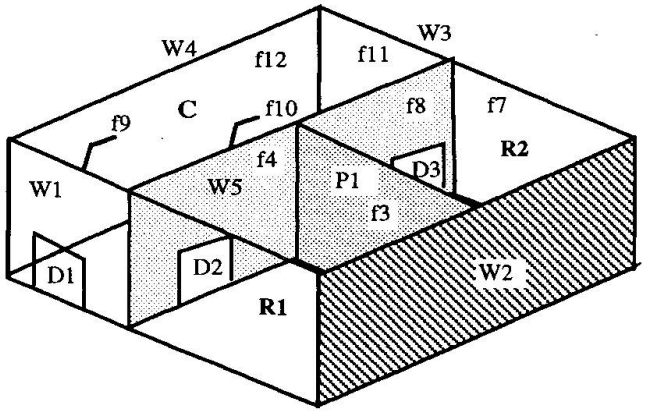

Figure 1. An illustrative building environment

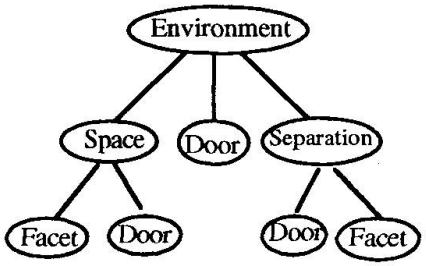

Figure 2. Structure hierarchy

Each spatial entity (facet, door, space or wall) possess some properties like "coordinate", "length" and some topo-geometrical relations with other entities like "adjacent" between two rooms, etc.
We have adopted the object-oriented representation, owing to their encapsulation and inheritance faculties [6, 10], to represent the spatial entities.
Each entity is described by its intrinsic properties, like the width, length of a space, the height of a door ; and by the topo-geometrical relations with other entities, like composition relation between a composite entity and its components, adjacent relation between two spaces, etc.
Each spatial entity is represented by an object with own identifier and attributes. Three categories of attributes have been distinguished :

- terminal attributes being peculiar to an object (size, area of a room, ...) ;
- constituent attributes pointing to the component objects of a composite object (facets and doors of a room, ...) ;
- topological attributes representing the link between objects ("space-linked" of a door, "furniture-contained" of a room, ...).

We define a special attribute "Graph" associated with each composite object in order to link it with its structure graph.
For example the state of Room R2 in the figure 1 can be described as below :

R2 = (<Width 3.0>, <Length 4.0>, <Facets (f1, f2, f3, f4)>, <Doors (D2)>, ..., <Graph Gr2>)

Spatial entities of the environment are described by the following object classes : Facet, Door, Separation, Space. Separation has two sub-classes : Partition and Wall ; Space has also two sub-classes : Room and Corridor.

3. TOWARD A CONSISTENT ENVIRONMENT MODEL

An other type of relations existing between the spatial entities, allow us to define some restrictions on the state or structure of spatial entities, such as for example "the area of a room is equal to the product of its width and its length", "each space must be accessible from the exterior", "the length of the room R2 must be lower than 3 metres", etc. We consider these relations as constraints which may concern, on the one hand, the intrinsic properties of spatial entities, and some arbitrary functional requirements on the other hand.

These constraints constitute a truth model allowing us to control the state consistency or the structure consistency.

3.1 Truth Model Description

Constraints of consistency

In our context, the constraints of consistency concern all the geometrical relations, topological relations and the semantics of the environment's entities, as the examples in the following :

- the constraint of object's position : an object can not be present on the left and on the right of another at the same time, etc. ;
- the constraint of object's structure : a door is always incorporated in a wall ; each space must have a door such that with which this space can be directly or indirectly accessible from the exterior ; etc. ;
- the algebraic relation : the area of a space is the multiplication of its width and its length ; etc. ;
- the deduction : if a door exists between two spaces, then a passage exists between the two spaces, etc. ;
- the exigency : the machines room must have a door such that its width ranks above 2.0 metres ; etc.

Classification and representation of the constraints

The constraints in the truth model differ from those of CSP [11, 3, 2, 9], in that they concern the variables having infinite value domains (the constraint's variables in the CSP have generally the discrete and finite value domains). The constraints can be classified in three types according to the manner in which they are employed : constraints-to-be-propagated to propagate an objects change ; constraints-to-be-verified-and-propagated to refine previous propagation and constraints-to-be-verified to verify objects consistency.

To describe the different constraints, we use a constraint representation formalism, based on the sorted first order predicate logic [12]. The predicates of form "Attribute(entity, value)" are used to present the triplets <entity, attribute, value>.

Constraints-to-be-propagated (Cp)

The Cp constraints are some directed relations between numerical and symbolic data. They allow to add new values to entity's attributes, to remove an object and even lead to the creation of entirely new objects.

A Cp is represented as follows:

$$<Quant_{prem} \mid T_{prem}> \{<Quant_{prem} \mid T_{prem}>\}$$
$$<C_{prem}> => \{<Quant_{conc} \mid T_{conc}>\} <C_{conc}> [\mid \mid R]$$

where:

$<Quant_{prem}>$ represents the quantifier ($\forall$ or $\exists$) of a variable "prem" in a premise ;

$<T_{prem}>$ represents the type of "prem" ;

$<C_{prem}>$ represents the body of constraint premises ;

$<C_{conc}>$ represents the constraint's conclusion ;

<R> represents a formula that solves the constraint on the variables in the premises and the

conclusion.

Examples :

Cp1: $(\forall x \mid Space(x))\ (\forall y \mid Space(y))\ (\forall z \mid Door(z))$

$Bound(z, x,y) \Rightarrow (\exists p \mid Passage(P))\ (Space\text{-}between(P, x,y) \wedge Access(P, z))$

Cp2 : $(\forall x \mid Space(x))\ Length(x, L) \wedge Width(x, w) \Rightarrow Area(x, a) \mid\mid a=l*w$

The constraint Cp1 describes that a passage exists between two spaces if there is an open access between them. The constraint Cp2 says that we can get the space area from it's length and width.

Equivalent constraints

An algebraic relation between n variables can be represented by n calculating formulas, we will define n equivalent constraints Cp to represent all the formulas. However, only one of them can be triggered in a reasoning cycle, to avoid a dead loop. To do this, we introduce a particular predicate : Deduce(O, A, o1, a1, ..., oi, ai) which says that the value of attribute "A" of object "O" is the calculating result of the value of the attribute a1 of object o1, ..., and the value of attribute ai of object oi. Then the relation represented by Cp2 can be defined by three equivalent constraints and one of which can be represented as below :

Cp2' : $(\forall x \mid Space(x))\ Length(x, l) \wedge Width(x, w) \wedge \neg Deduce(x, Length, x, Area, x, Width)$

$\wedge \neg Deduce(x, Width, x, Area, x, Length) \Rightarrow Area(x, a) \wedge Deduce(x, Area, x, Length, x, Width) \mid\mid a=l*w$

Constraints-to-be-verified(Cv)

A Cvi∈ Cv is one or several directionless relations between objects. It gives a Boolean value and is represented as follows:

$$<Quant_{prem} \mid T_{prem}> \{<Quant_{prem} \mid T_{prem}>\} <C_{prem}> \mid\mid <P>$$

where: <P> represents a set of predicates that use the variables from the premises as their arguments, then they will be satisfied by these variables. It may be one or several formulas (linked by "And") of comparison between the variable and its reference (as in Cv1) , or one or several geo-topological relations to be verified (as in Cv2 and Cv3).

Examples:

- the area of a room is equal to the product of its width and its length :

Cv1: $(\forall x \mid Room(x))\ Length(x, l) \wedge Width(x, w) \wedge Area(x, a) \mid\mid a=l*w$

- a rectangular room consists of four closed facets and of a door at least :

Cv2: $\forall x\ Room(x) \mid\mid Facets(x, F) \wedge Doors(x, D) \wedge Cardinality(F, 4) \wedge$

$Length\text{-}higher(D, 1) \wedge Closed(F)$

- each space must be accessible from exterior

Cv3 : $(\forall x \mid Space(x)) \mid\mid Accessible(x)$

The first consists in controlling the attribute "area" of a room, the second consists in controlling the global structure of a room, and the last consist in controlling a topological relation between each space and the exterior which is a special space.

In the Cv1, when one of the three variables is a calculating result of the two others, by the constraint

Cp, this constraint is automatically satisfied. To avoid the redundancy, we introduce also the predicate Deduce(O, A, o1, a1, ..., oi, ai), the Cv1 is transcribed as :

Cv1': $(\forall x \mid Room(x))\ Length(x, l) \wedge Width(x, w) \wedge Area(x, a) \wedge \neg Deduce(x, Length, x, Area, x, Width) \wedge \neg Deduce(x, Width, x, Area, x, Length) \wedge \neg Deduce(x, Are, x, Width, x, Length) \mid\mid a=l*w$

Constraints-to-be-verified-and-propagated (Cvp)

A Cvpi∈Cvp is a combination of the two preceding types of constraints. It is a constraint to be verified (Cvpvi), associated with other constraints to be locally propagated (Cvppi). The constraints to be propagated (Cvppi) are triggered when the verified constraints (Cvpvi) are unsatisfied.

Examples:

- Cvp1:

Cvpv1: $(\exists R3 \mid Room(R3))\ Area(R3, a) \mid\mid a="Fixed"$

Cvpp1: $(\exists R3 \mid Room(R3))\ Area(R3, a) \wedge Width(R3, w) \Rightarrow Length(R3, l) \mid\mid l=a/w$

$(\exists R3 \mid Room(R3))\ Area(R3, a) \wedge Length(R3, l) \Rightarrow Width(R3, w) \mid\mid w=a/l$

This example indicates that if the surface of Room R3 must remain fixed, its length must be modified if its width is modified, and its width must be modified if its length is modified.

The satisfaction of a Cvp begins with the evaluation of Cvpv ; if the Cvpv is unsatisfied a Cvpp will be chosen. This will be repeated until the Cvpv is satisfied.

3.2 Consistency Maintenance of the Environment Model

Two types of consistency control may be considered : static consistency control and dynamic consistency control.

The static consistency control concerns the checking of a stable environment model with respect to the constraints-to-be-verified Cv. We must control : if each object is a such object designated by the nom of its class (its proprieties and its structure) ; if the geo-topological relations between different objects are right, etc.

The dynamic consistency control takes place when any entity is changed. A change may be a real change or the noise of IAMR's vision sensors. It may be consistent itself, but its induced results may be not consistent. So, to control the consistency of a change, we verify in the first if the initial modification is consistent, and then we propagate the modification by using the constraints-to-be-propagated and verify if the induced results are consistent.

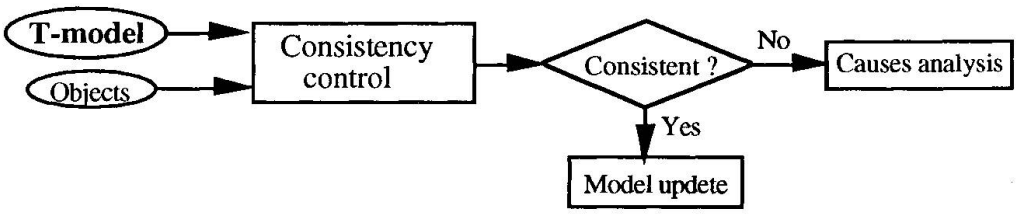

Figure 3. Objects consistency control schema

If the change and the results of its propagation satisfy all the constraints of T-model, we consider that this change is consistent and the environment model may be updated, otherwise we don't accept this change and it triggers its analysis module to detect the anomaly's causes (figure 3).

4. CONSISTENCY MAINTENANCE SYSTEM : PROVE

The consistency maintenance system PROVE which realises the above ideas is implemented on a SUN workstation and developed with the object language AIRELLE (a super-layer on the Le-lisp 15.2 of the INRIA) [8].

4.1 Constraint Representation

Different constraints are represented by three object classes : "Constraint" for Cp, "Constraint-V" for Cv, and "Constraint-VP" for Cvp.

The first and the second classes have three principal attributes : "Premise" representing a constraint's premise (the left part of the connector "=>" of Cp or "| |" of Cv), "Conclusion" representing a constraint's action (the right part of the connector), "Condition" which represent an optional trigger condition.

The third class is a combination of the two first classes, and it has two attributes : "Constraintv" which points to a constraint Cv, "ConstraintP" which points to a list of constraints Cp.

A constraint object may be presented in the form of a list composed of three parts, each of which begins by a reserved key word : "If" for the premise, "Such-as" for the trigger condition, and "Then" for the conclusion.

Constraint = <((If premise1 premise2 ...) (Such-as condition) (Then action1 action2 ...))>

Premise

A premise is represented by a triplet : <Object attribute value> as :

((Class x) Attribute (type y))

Where "Class" is the sort of x which limits the value domain of x, "Attribute" is an attribute of x, "type" is the attribute value type, and y is a variable representing the value of the attribute.

When the object or the attribute value is known, the premise becomes as following :

(O Attribute (type y)) or ((Class x) Attribute V)

Condition

A condition is a LISP-like logical expression such as : (equal x y).

Action

A constraint's actions have different representations as below :

(=> A Attribute V)	for modifying of an attribute's value
(O Remove A)	for removing "A" instance of "O" class
((O Create A) (=> A Attribute V) {(=> A Attribute V)})	for creating a "A" instance of "O" class
(>= X V)	for comparing a "X" variable with its "V" reference

Where : "=>" is the message sending function ; "A" is a known object ; "Attribute" is a real object attribute (but is not a variable) ; "O" is an object class ; "V" is the value of an object attribute, which can be a constant or formula ; "Remove" and "Create" are two reserved words. The first formula assigns "V" value to "Attribute" attribute of "O" object.

Examples :

The Cp1 and Cv1' can be rewritten as :

```
Cp1 :  (      (if      ((Door D) Space-linked (ens S)))
              (Then ((Passage 'Create P) (=> P 'Space-between S) (=> P 'Access D))) )
Cv1 :  (      (If      ((Space s) Length (float l))
                       ((Space s) Width (float w))
                       ((Space s) Area (float a)) )
              (such-as (and (and (not (Obtain s Length s Width s Area)) (not (Obtain s Width s Length s Area)) )
                            (not (Obtain s Area s Width s Length))
              (Then  (= s (* l w))) )
Cv3 :  (      (if      ((Model M) Space (ens S)))
              (Then  ((mapc S) (Verify-Accessible var-inter)
```

Where the "Obtain (O A o1 a1 ... oi ai)" function is a procedure verifying if the value of A attribute of O object is the calculating result of the value of a1 attribute of o1 object, ..., and the value of ai attribute of oi object. "ens" is a data type of AIRELLE. "(mapc S)" showing that the following action will be executed in an ensemble of objects (as a "broadcast") (we do not use the method "applique" which allows to do also a "broadcast", in order to easily recuperate the action result).

The equivalent constraints are marked by an attribute "C-equivalent". For example the "C-equivalent" value of Cp2' is "Cp3, Cp4", the one of Cp3 is "Cp2', Cp4", and the one of Cp4 is "Cp2', Cp3".

The constraints and the corresponding objects are linked by the sort of each constraint which presents in the premise of a constraint.

4.2 PROVE's Running

The PROVE system contains four principal modules as illustrated in figure 4. When an object modification is received, in the first PROVE propagates, using the constraints Cp, this modification throughout the model to induce all its effects, then it refines this propagation by using the constraints Cvp, and finally the consistency of the modification and its induced results will be verified with respect to Cv. Therefore the propagation process of a modification is non-linear.

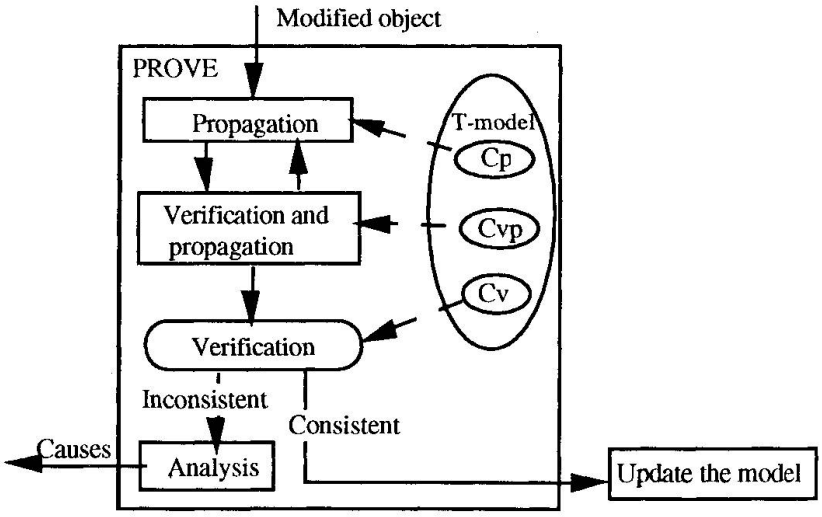

Figure 4. Structure of the consistency maintenance system

The **Propagation** module triggers Cp constraints in the order determined by the priority assigned to each constraint, using the "forward-chaining" control structure with the "width-first" search strategy. The modified objects or the new created objects are stored in a special objects class called DMODIF. This mechanism prevents from operating directly on the model : if the original change is considered unacceptable, the model will remain unaffected.

The **Verification and propagation** module proceeds in two stages. It begins by verifying the Cvpv of each Cvp constraints; if they are satisfied, the module pass to the next one.

If a Cvpi∈Cvp (its Cvpvi) is not satisfied the module immediately propagates a corresponding Cvppi constraint. This propagation may modify some data that retriggers the "Propagation" module and starts so a new operating loop. The looping continues until either the Cvp constraints are satisfied or some stop conditions are verified (for example when there are Cvp constraints that cannot be satisfied by the model during its processing state).

The **Verification** module consist in verify the Cv constraints. If all the constraints are satisfied the change signalled by the original message is considered acceptable and the model will be updated.

If a Cvi∈Cv constraint is unsatisfied, the change will be not accepted and an anomaly will be declared.

The **Analysis** module makes use of a mechanism inspired from ATMS [4], to identify the set of primary causes (objects initially modified) of each anomaly. The "Analysis" module treats as hypotheses the modifications contained in the message that triggers PROVE. It receives couples of the form (O, J) from the "Propagation" module, where O is a modified object and J the justifications for the modification. Since the object modifications in the original message are treated as hypotheses, the objects justify themselves. For each modified object the module computes a label L representing the hypotheses that have brought about the modification, by constructing a derivation tree using the concerned justifications. A hypothesis is self-labelled by definition and the modified objects are noted as combinations of (O, J, L).

5. CONCLUSION

We have presented a consistency maintenance system PROVE for a spatial object model in the context of our research project : IAMR environment simulation.

We qualify this system as a constraint-based system. Three types of constraints have been classified for the propagation (Cp, Cvp) of an objects change as well as for the objects verification (Cv). However, the constraints in our context are different from the ones in CSP. On the one hand, the domain of variable value of our constraints is rather infinite than finite and discrete. On the other hand, the Cp constraints don't exist explicitly in CSP. Generally the constraints treated in CSP may be classified as Cv and Cvp constraints (for example, in the geometric constraint engine [5], only the Cvp constraints have been treated).

The integration of hypothetical reasoning, to analyse the origins of unsatisfied constraint provides the possibility of a collaboration with IAMR's module vision to identify an object perceived by its sensors.

REFERENCES

[1] M. AYEL et M.C. ROUSSET : La cohérence dans les bases de connaissances. Cepad, 1990.

[2] A. DECHTER, R. DECHTER : Belief maintenance in dynamic constraint networks. Proc. AAAI 88.

[3] J. JATTAR et J.L. LASSEZ : Constraint logic proramming. Proc of the 14th ACM POPL Symposium, Munich, West Germany, 1987

[4] J. de KLEER : An assumption-based TMS. Artificial intelligence. vol. 28, 1986.

[5] G. A. KRAMER : A Geometric Constraint Engine. Artificial intelligence. vol. 58, 1992.

[6] G. MASINI et Al. : Les langages à objets : langages de classes, langages de frames, langages d'acteurs. InterEdition, 1989.

[7] A. MEYSTEL : Autonomous Mobile Robots - Vehicles with Cognitive Control. World Scientific, 1991

[8] A. NICOLLE-ADAM : Le métalangage AIRELLE. Revue d'Intelligence Artificielle, vol 4, n°1, Hermes, 1990.

[9] F. ROSSI : Constraint satisfaction Problems in logic programming. SIGART news letter, Oct. 1988.

[10] M. STEFIK et D. G. BOBROW : "Object-Oriented Programming : Themes and Variations". The AI magazine, Vol. 6.1, 1986.

[11] G.J. SUSSMAN et G.L. STEELE : CONSTRAINTS - A language for expressing almost hierarchical descriptions, Artificial Intelligence, 14, 1980, P.1-39

[12] A. THAYSE, A. BRUFFAERTS etc, Approche logique de l'intelligence artificielle. vol.3, Dunod, 1990.