Constraints Satisfaction and Symbolic Reasoning for reactive control systems

Anne Liret

LIP6, pole Artificial Intelligence
University of Paris 6
4 place Jussieu,
75252 Paris, Cedex 05, France
e-mail: Anne.Liret@lip6.fr

Pierre Roy

INRIA, Domaine de Voluceau,
Rocquencourt, France.
CREATE, Department of Music,
UCSB, California, USA
e-mail: pierre@create.ucsb.edu

François Pachet

Sony CSL Labs, France.
e-mail: pachet@csl.sony.fr

Abstract

This paper proposes a an object-oriented framework for solving constraint satisfaction problem, with a collaboration between consistency methods and deductive symbolic manipulation of constraints. Object-oriented approach is adapted to the design of constraint-based control system, (for instance CDL, a control and information system for unmanned vehicles or remote surveillance). The solver is designed as a reactive object which reacts to events on variables, constraints, symbolic manipulation results and authorizes adding or removing a constraint. Combining numerical and symbolical reasoning to handle constraints improve the reaction of the system, in quality and rapidity sometimes. It detects inconsistency earlier and therefore avoids bad answer to the user. But symbolic manipulation should carefully used, in relation to the knowledge of experts and to the solver status. Thus a mechanism of extensible and combinable invariants allows to specify and enforce strategy in a declarative manner, as well as the statement of uncertain strategy thanks to heuristic rules. One uncertain strategy represents one intuitive knowledge which is not always true but may be useful to solve problem with large domains.

1 Introduction

Constraints are intrinsically integrated in dynamic control systems such as crisis planning, management of one environment. We note involved constraints: a) Scheduling constraints, for instance if there are several zones to irrigate and several irrigation systems, the control must schedule which plants to allocate to which irrigation canal, b) Capacity constraints, for instance, water is distributed relating to the type of plants and the capacity of absorption. Capacity constraints can be represented by cardinality constraints, c) polynomial constraints express for instance robot coordinates in action.

Control systems requirements are threefold: First of all constraints are generally global and non-binary. Expressions are complex or constraints relate to composite data structures. Second, the system has to react fast enough to environment events, therefore it is designed a an reactive entity where constraints are dynamically added or removed. Third, experts need to fine-tune strategies for improvement of control. Mathematical models traditionally implement constraint for efficiency and formalism facilities. But when values are too complex, global constraints are necessary. On the other side, consistency methods are not efficient because they are general and perform complete consistency. Object-oriented languages are useful to create constraints on variables with structured objects as values.

We consider applications involving one constraint-based control process. By constraint-based control application, we mean any dynamic problem represented by a set of constraints on the environment objects such that every change in the environment has to satisfy the constraints, for instance, planning management, medical monitoring and geometric figure computer aided engineering. One need a solver that dynamically maintains consistency. For efficiency reasons, local partial consistency is preferred to complete consistency, but since completeness is no more ensured, some inconsistencies may be forgot.

1.1 Proposed solution

Object-oriented approach allows to represent concepts in a manner closer to real-world than mathematical one, to state global constraints and constraints on structured objects. Therefore the problem is understandable to experts and the use of global constraints instead of simple ones limits the total number of constraints and simplify the problem resolution. The constraint satisfaction solver is reactive ; it maintains the CSP consistent and manage addition and retraction of constraint. To improve consistency in global constraints, a symbolic reasoning based on rewriting and redundant constraint introduction may help CSP solver to find the result more directly. Linear composition handles linear and polynomial equations and inequalities, deduction on generic cardinality and functional constraints are general. The goal of combining symbolical and numerical treatment of constraints if to detect inconsistencies earlier, to find variable instantiation without making a choice. The idea of translating the problem into another smaller one exists in dynamic control systems, but it needs numerically difficult calculus [Kang 1994]. In control systems, constraints seem to be originated from experts. Thus it is important to have a mean to express naturally this knowledge. Rules allows the expert to implant both its certain knowledge (formal reasoning, strategy) and its intuitive knowledge (uncertain strategy). Deductive strategies can be stated, redefined and combined together, in order to guide the resolution or dynamically change the problem. Formal reasoning and strategies mean laws on the application domain, constraint type or intuitive resolution tips. They are designed as invariants on rules, solver, or environment objects. We have implemented our approach into the *BackTalk* framework. [Roy and al. June 1998] for constraint satisfaction in finite domain.

2 Object and constraints to design real-world problem

Using composite objets allows to make the problem more comprehensive. If the CSP solver accepts general constraints on variables with objects values (the domain is denumerable but the elements are complex objects), then it becomes easier to post general constraints representing a real-world problem. For instance, the domain can be an ordered set of irrigation canals in control of plants management or coordinates of points in control of vehicle movement. In an object-oriented framework for stating and solving objets-CSP like *BackTalk/BackJava*, constraints are organized in a hierarchy of classes. Sub-classing existing classes allows to create a new class of constraint and reuse predefined mechanisms of consistency.

Constraints on objects regroup the power of expression on objet models and the clearness of declarative language to state required relation between objects. Two general object constraints have been designed. Functional constraints (`BTPerformCt`) in the form `F(x, y) = z`, where `x` and `z` are variables, `y` is either a value or a variable and `F` is any operation symbol on every object in `x` domain and `z` domain or value. Such a constraint is created by instantiating the class `BTPerformCt`. Thanks to the constraint language of the framework, the constraint is stated

simply by writing the expression in the host language ; for instance, in a problem of allocating computer service demands, given `X` a client and R a set of resources with energy capacity, compatible with the client `x`, `((x neededEnergy) @= (R selectEnergy) sumOfElements)` creates the general constraint $\sum_{r \in R} availableEnergy(r) = neededEnergy(x)$.

For instance, `isOpen(Canal1) @= Z` means that `Z` is true or false depending on the state of the irrigation canal number 1. General Cardinality constraints (`CardinalityGenCt`) mean that the size of `{X in S / P(x)} is Y,` where `S` in a set of variables, `P` is a property of any value of the variables and `Y` is a variable. For instance, the CSP `{|{X in SetOfCanals / isOpen(X)}| = Y, Y in 1..10}` constrains the number of open irrigation canals to be bounded by 1..10. These constraints have been used for scheduling courses of a class in university (Paris 6), where one task is characterized with teacher, course, duration, preferred dates and hours. For instance `|{p in Teachers / uses(p) = #slides)}| <= 1`, and `(#slides = uses(p))` ➔ `endTime(p) <= 13*60`, since the projector is only available in the morning.
Specialized global constraint with optimized algorithm have been found for global cardinality, capacity, rotating constraints [Régin & Puget 1997] but global object constraints remain difficult to handle when maintaining consistency in general..

3 Reactive solver to handle constraints

The solver handles object and numerical constraint satisfaction problems by combining CSP techniques and symbolic reasoning based on rewriting and redundant constraints deduction. Combination can be seen as a collaboration of solvers on a shared constraint store. Collaboration is closed to an autonomous agents architecture, each one including a solver. This approach by reaction to events differs from traditional blackboard-based control.
BackTalk was designed as a dynamic reactive solver where events are addition or retraction of simple constraint on one variable domain, or addition or retraction of complex constraint. The solver reacts to addition of simple constraints ($X = \alpha$, $X \neq \alpha$, $X \geq \alpha$) by performing local consistency on the whole problem. Every constraint reacts to events on a variable of it by modifying another variable. Variables and constraints were linked by variable-daemons, specific of the variable event. Due to efficiency reasons, partial consistency like Box-consistency [Van Hentenryck and al. 1997] is performed instead of complete consistency. This means that some incompatible values may be forgotten. Nevertheless some control situations enforce the solver to guaranty the absence of inconsistent values.

Example

> In a medical monitoring for on-line diagnostic of patients hospitalized in care units [Ramaux and al. 1997], the goal is to foresee the future evolution of a patient's health, considering its current evolution and some predefined possible future evolution. With this aim, the system inspects the evolution of a patient (the session) and compares it with medical expert knowledge, represented by temporal constraints graphs, called scenarios. A scenario describes one possible and supposed evolution of health, in function of medical events (increase of temperature, decrease of pressure decrease of oxygen rate). A session describes the actual evolution of health. The events are represented by dates while the interval of time variation between two dates is represented by one constraint. From a predefined base, the collects scenarios such as they correspond to the session. More precisely it merges every scenario with

the session, which results in fusion graphs. Then it check any inconsistency into them since one scenario is incompatible with the session if and only if their fusion graph is inconsistent. This phase must run as fast as possible, due to the real-time nature of monitoring. Since a patient's health is managed, one want to guaranty the compatibility between scenario and session.

For another point, several situations have been extracted where a symbolic reasoning on structures of constraint improve CSP techniques [Liret and al. 1997]. Since it adds redundant constraints, the problem's density increases, therefore the inconsistency is found earlier. We have extended our CSP solver with symbolic reasoning method like constraint simplification and constraints linear composition. The extension of *BackTalk* is also designed to be reactive to events on solver state, result of symbolical or numerical manipulation. Constraints-daemons relate constraints to Formal reasoning rules ; they trigger whenever a complex constraint is rewritten or added. Strategy of FR-rules is enforced by simply posting an invariant, that is instantiating a class. An invariant is updated, via rules-daemons, whenever its rules are to apply or have finished to trigger.

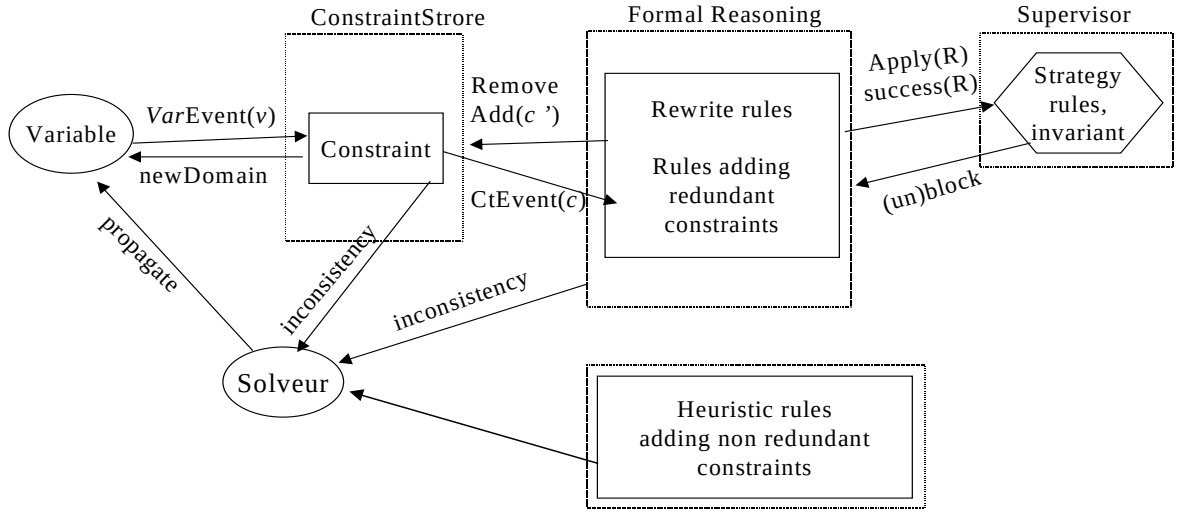

Figure 1: Levels of the extension of BackTalk

The system is made up of three levels of knowledge: Constraints and variables (CSP), symbolic reasoning method encapsulate in rules, and control strategies. When one constraint is added or involved is one FR-rule, it becomes candidate for symbolic treatment. the problem is treated by local partial consistency, then formal propagation is processed. If the latter changes the problem, one achieves bounds-consistency, otherwise one divide the search space.

4 Helping CSP techniques with symbolic reasoning

We model a symbolic technique as a solver transforming a set of constraints into a other one, without changing the solutions set of the problem. We distinguish two ways of handling a constraints store, rewriting and redundant information introducing. These ways regroup a great number of symbolic techniques, like constraints simplification, Gauss elimination method, Simplex algorithm on inequalities, linear composition of equations. Indeed, a solver is defined as an algorithmic entity taking data in a certain form as input and returning a result (a set of new

data) as output. Due to the similitude between the rule concept and solver definition, we use the general formalism of rewriting conditional rules to express symbolic transformation techniques.

```
Exists(i1,…,in)
➔ condition(i1,…,in) | SolverCalculus(i1,…, in),
                              add(r1,…, rj), remove(p1,…, pk).
{ p1,…, pk } ⊆ { i1,…, in }
{ r1, …, rj } ⊄ { i1,…, in }
```

If the problem contains n constraints satisfying the conditions of the solver, then the solver is executed and, as a result, new (redundant) constraints are added and a subset of the input constraints are removed.
RF-rules are organized in a hierarchy of classes, distinguished by the premises number and their nature. Each class represents a generic rule whose instance is linked to the set of candidates constraints. A rewrite system simplifies constraints (including the commutative associative ring of arithmetical ((+,0)X(*,1)) expressions, augmented with functional terms), while formal deductive manipulation of existing simplified constraints add new redundant constraints.

4.1 Linear composition

Linear constraints composition and object constraint composition are general rules adding new redundant constraints, while combination of *AllDifferent* and equality detects inconsistency, as well as the substitution of equal variables. Linear constraints composition is an hybridization between Gauss elimination method and Simplex. Since constraints are decomposed, this rule applies on polynomial constraints. From two linear constraints C_1 and C_2, with common variables, we can find a coefficient k such as C_1 - k*C_2 removes the maximum of common variables. There may be several k ; in general C_1-k*C_2 isn't smaller than both C_1 and C_2 ; it often occurs that the result is smaller than either C_1 or C_2.

C_1, C_2 ➔ S = ComputeRatio(C_1,C_2), For all k in S, add(C_1- k*C_2).
(k must be negative if C_2 is an inequality)

Figure 2. FR-rule deducing linear redundant constraint from two linear constraints

We validated this rule on 10 series of randomly generated problems of the same type (non linear quadratic equalities and inequalities on finite domains), by varying in variables number, constraints number and problem density. Most of these problems have no solution. The question was: could the cost of the introduction of FR be balanced by the gain of quality (reduction of the backtrack number) ? We find that on non-linear problems on finite domains, CSP have some difficulties whereas for 25% of the problems set, the use of FR reduces the backtracks number and it increases the time for less than a factor 2 (for 10%, it reduces the time).

4.2 Object functional constraints composition

We defined a general F.R.-rule that allows to express a deduction in the form:

$Y_1 = \Phi_1(X_1, Z_1), \ldots, Y_n = \Phi_n(X_n, Z_n)$

➔ add($\bigwedge_1^p (Y_{\alpha_i} = \Gamma_i(X_{\beta_i}, Z_{\gamma_i}))$, remove($\bigwedge_1^q (Y_{\alpha j} = \Gamma_j(X_{\beta_j}, Z_{\gamma_j}))$)

This rule is expensive and is rather used with n = 2.

Example

Let consider a geometric constraint solver in computer aided engineering. One must place geometric objects on a screen with the aim to build a mechanical tool figure. Constraints are on the length of one arc, the angle between two arcs, intersection points position, tangency to a circle,…. When the user state a new constraint, the current figure is updated in consequence. New position of the figure elements result from the resolution of the geometric constraints problem (variables stand for the elements' coordinates). Geometric constraints can be viewed as sort of functional constraints, for instance, d1,d2 and d3 being lines, `(d1 isPerpendicularTo: d2)` is the (written as is) constraint to say that line d1 is perpendicular to line d2. The FR-rule allows to express geometric theorems, since hypothesis are constraint in condition part and conclusion is the constraint in consequence part. For instance, the rule `[d1 isPerpendicularTo: d2 AND d1 isPerpendicularTo: d3 ➔ d2 is parallelTo: d3]` improved the resolution of the geometric CSP by reducing the space size of the coordinates before le first choice.

5 Specifying strategies

Strategies are defined by rules which correspond to the knowledge natural representation by experts. Strategies are modeled as rules reacting to events on constraints, solver situation(number of constraints, number of rule failures) or application of symbolical or numerical techniques. The set of events can be redefined thanks to demons (section 3). Strategies are a simple way of expressing constraint dedicated to update of a dynamic application ; it avoids to use global constraints in the case where they seem to be heavy to handle regarding to the update to perform.

5.1 Certain strategies

In order to limit the cost of symbolic reasoning, strategies are posted one FR-rules. For instance, one can abandon a FR-rule after it has failed 5 times, one can choose different kinds of combination of linear constraints or enforce once rule application. They express laws of the application domain, thus they called "certain strategies". Certain strategies are closed to the notion of invariant defined in *Localizer* [Michel & Van Hentenryck 1997] since they update the rules status (blocked, applicable) incrementally during the resolution. An invariant is viewed as a global constraint reacting to FR-rules status change event. Every event on an object O is associated to a method of strategy class, with predefined title, which implements the reaction of the invariant (invariant is the concrete representation of certain strategy). They use global information and influence a set of entities (FR-rules, object of the environment like figure position, constraints store). Thanks to its declarative status, invariants fine-tune the collaboration between heterogeneous solvers, in a simple manner.

```
LinearCompositionRule. setExecutionControl
    (ItComposition on: self).
    (ItStopIfUseless on: self nbFailure: 5).
```

Figure 3. Declaration of control in the class of linear composition.

- `ItComposition` strategy controls the way of composing two linear constraints. If the arity of the active constraint C is 2 then compute all the possible compositions with C and the linear constraints of the problem ; otherwise compute only the compositions which produce a smaller constraint than one of the initial constraints.
- `ItStopIfUseless` strategy authorizes a linear composition with a given active constraints, while less than N applications have been useless (N given at creation). It is implemented by a counter, incremented whenever the composition is declared useless.

```
ItStopIfUseless(mainRule = r1, counter = 0, n = 5):
    updateRuleIsUseless: r       mainRule = r ifTrue: [
                                        counter := counter + 1.
                                        counter >= n ifTrue: [mainRule block]]

    testRuleBeforeTriggering: r         ^true
```

Figure 4. code Smalltalk for ItStopIfUseless.

A control invariant reacts to any status change of its rule (rule is going to trigger or has just applied, result is useless or useful). Two basic actions has to be implemented: test before triggering if the rule is authorized to run, and update the information of the control invariant after triggering. If an invariant reacts to a particular rule event (useless event for instance), then it must implement the corresponding method.

5.2 Uncertain strategies

An other use of strategy is to define heuristics that divide the space according to a non redundant constraint, instead of a choice variable-value ($x=\alpha$, $x>=\alpha$). This kind of strategy reacts to an event generated by the solver (making a choice). Such heuristic rules is used in particular to deal with float (exact value does not exist) and with disjunctive constraints. Thanks to heuristic rules, control experts can model their intuitive knowledge, their resolution empirical tips. For instance in a scheduling problem, such a strategy may define the task allocation order, forbid the sue of particular symbolic deduction rule due to its cost, and maintain a general state of a control system and of its controlled environment. For instance, in medical monitoring, uncertain strategy models the fact that, whenever the oxygen rate in blood decreases, the control system must activate the oxygen pump.

6 Related works

CLP systems include constraints in inference rules in order to control the triggering of the rules [Marriott & Stuckey 1998]. In CLP systems, constraints are global and persistent till the problem is solved. *Constraints Handling Rules* (CHR) extend CLP rules with the possibility to modify the constraints store. But the control of CHR remains a problem. The idea of heterogeneous modular CSP system is not new since the *Alice* system [Laurière 1976] solved well-known difficult problems by combining automatic constraint deduction on general constraints and the enumeration of the domains with arc-consistency (AC-4 [Mohr & Henderson 1986]). Alice's idea means applying methods with high cost but good qualitative result, on one part of the problem and fast but less precise methods on another part of the problem. Recent works deal with the

same subject in the range of the control of a robot behavior [Benhamou & Granvilliers 1996], or scheduling [Chabrier 1999], [Laburthe 1998], with polynomial non linear constraints.

7 Conclusion

Representing and dealing with design knowledge in a declarative manner is promising for designing control systems and in general systems that dynamically maintain the consistency of an environment. Representation with constraints and rules is straightforward and understandable for experts. The resolution process is automatically performed, which avoids experts to consider the process in all and helps them to focus their interest on the strategies. Moreover the merits to use an object-oriented approach are threefold: 1) Application concepts and global constraints are defined and represented at high-level, 2) subclassing make simple to integrate a declarative architecture for specifying any invariant of strategy, 3) events propagation between autonomous solvers allows to combine constraints symbolic manipulation techniques with existing consistency methods and also to fine-tune resolution strategies into a reactive constraint-based control application. The collaboration between constraints and Formal Reasoning (CSP+FR) detects inconsistency earlier, solves the problem more directly and sometimes faster. It is useful on general object constraints because no optimized consistency algorithm can be used. Due to the cost of symbolic computation, one should involve it into an *off-line* process. One make a trade-off between time optimization and user satisfaction. Finally certain and uncertain strategies provide a high-level way of adding constraints to guide the resolution.

8 References

[Benhamou & Granvilliers 1996] F. Benhamou & L. Granvilliers. Combining Local Consistency, Symbolic Rewriting and Interval Methods. *Proceedings of Artificial Intelligence and Symbolic Mathematical Computation, International Conference (AISMC'96), Steyr, Austria*, LNCS, Springer-Verlag, vol. 1138, pp. 144-159, September 23-25 1996.

[Chabrier 1999] A. Chabrier. A Cooperative CP and LP Optimizer Approach for the Pairing Generation Problem. *CP-AI-OR'99, Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems, University of Ferrara, Italy*, AI*IA & AIRO, 25 - 26 February 1999.

[Kang 1994] Wei. Kang. Extended Controller Form and Invariants of Nonlinear Control Systems with a SIngle Input. *Journal of Mathematical Systems, Estimation, and Control*, vol. 4(2), pp. 1-25, 1994.

[Laburthe 1998] F. Laburthe. Contraintes et Algorithmes en Optimisation Combinatoire. Informatique, Université Paris VII-Denis Diderot, Paris, France, 1998.

[Laurière 1976] J.L. Laurière. Un langage et un programme pour énoncer et résoudre des problèmes combinatoires. Ph. D., University Pierre et Marie Curie, Paris, 1976.

[Liret and al. 1997] A. Liret, P. Roy and F. Pachet. Combining Formal Reasoning Techniques and CSP. *ERCIM Workshop Constraint Programming and Processing (en conjonction avec CP'97), Linz, Autriche*, octobre,27, 28,29 1997.

[Marriott & Stuckey 1998] Kim. Marriott & Peter J. Stuckey. Programming with Constraints, An Introduction. The MIT Press, Cambridge, Massachusetts, 1998.

[Michel & Van Hentenryck 1997] L. Michel & P. Van Hentenryck. Localizer: A Modeling Language for Local Search. *Proceedings of the Conference on Constraint Programming (CP'97), Linz, Austria*, LNCS, Springer-Verlag, vol. 1330, pp. 237-251, 1997.

[Mohr & Henderson 1986] R. Mohr & T. Henderson. Arc and Path Consistency revisited. *Artificial Intelligence*, vol. 28, pp. 225-233, 1986.

[Ramaux and al. 1997] N. Ramaux, M. Dojat and D. Fontaine. Temporal Scenario Recognition for Intelligent Patient Monitoring. *Artificial Intelligence in Medicine Europe, Grenoble*, 1997.

[Régin & Puget 1997] J.-Ch. Régin & J.-F. Puget. A Filtering Algorithm for Global Sequencing Constraints. *Proceedings of the Conference on Constraint Programming (CP'97), Linz, Austria*, LNCS, Springer-Verlag, vol. 1330, pp. 32-46, october 1997.

[Roy and al. June 1998] P. Roy, A. Liret and F. Pachet. Constraint Satisfaction Problems Framework. in Implementing Application Frameworks: Object-Oriented Frameworks at Work.edited by D. Schmidt and R. Johnson M. Fayad. Wiley & sons, vol. 2(17), June 1998.