The Use and Interpretation of Meta Level Constraints

Pierre Berlandier
SECOIA Project, INRIA-CERMICS,
2004, Route des Lucioles, B.P. 93, 06902 Sophia-Antipolis Cedex

Abstract. This paper introduces a model for dynamic constraint problems in which constraints and variables are comparable entities. This model provides a natural way to represent configuration or design problems wherein the set of objects and their constraints are bound to evolve during the solving process. Metaconstraints, i.e. constraints on constraint descriptions, are the central contribution of the model. Depending on which part of the description (the set of constrained variables or the relation that links them) they apply to, metaconstraints can be used to monitor the evolution of the problem’s variables or constraints. The implications of metaconstraints on the consistency maintenance process are studied and an implementation within the PROSE constraint language is briefly described.

1 Introduction

Constraint languages are still young programming systems. So far, the major concerns in their design have been the improvement of the solvers efficiency and the cooperation of various solvers in a single interpreter. Now that techniques are known for handling a wide range of constraints with fair efficiency, more attention is paid to the expressive power of those languages. In particular, topics like partial constraint satisfaction [6, 5], integration into an object world [1] or the use of constraint combinators [9] are being investigated.

The work reported in this paper is born of the observation that constraint expression is usually restricted to a fixed set of attributes taken from a flat set of variables. Yet, there are domains such as computer aided design where:

- constrained variables follow a hierarchical decomposition (an artefact is recursively composed of subparts),
- constraint problems are subject to dynamic evolvement as the search progresses (parts are added or removed).

To support these needs, we propose a new model for constraint problems, namely dynamic constraint problems. In the remainder of the paper, we first present the regular and then the dynamic constraint problem model. We show that the latter allows the expression of metaconstraints which fulfill our needs as they enable the construction of both hierarchical and evolving constraint
networks. Then, we present a method for interpreting these constraints for consistency maintenance and we briefly describe an implementation. Finally, we conclude with a comparison between our model and a closely-related work.

2 Constraint Problems

The regular constraint formalism derives from [15]. It presents a constraint problem as a tuple $(X, D, C, R)$ where:

- $X = \{x_1, \ldots, x_n\}$ is the finite set of variables of the problem.
- $D$ is a finite set of domains and $D$ is a bijection from $X$ to $D$ such that $D(x_i)$ is the domain attached to $x_i$. A domain may be finite or infinite, discrete or continuous.
- $C = \{c_1, \ldots, c_m\}$ is the finite set of constraints where $c_i$ is a tuple of variables $(x_{i_1}, \ldots, x_{i_p})$ from $X^p$. The $x_{ij}$ are called the attributes of the constraint. The set of constraints forms a hypergraph, usually termed the constraint network.
- $R$ is the finite set of relations and $R$ is a bijection from $C$ to $R$ such that $R(c_i)$ is the relation attached to $c_i$ defining (in extension or intention) the set of $p$-tuples allowed by this constraint.

The constraint satisfaction problem consists in finding one or more instanciations $V$ of all the variables in $X$ so that:

\[
\begin{align*}
\forall x_i \in X, \; & V(x_i) \in D(x_i) \\
\forall c_i \in C, \; & (V(x_{i_1}), \ldots, V(x_{i_p})) \in R(c_i)
\end{align*}
\]

The search for solutions can be achieved (quite efficiently in spite of the NP-completeness of the problem) for different types of domains such as discrete sets, real numbers, booleans or trees, using appropriate techniques [13, 4].

Another important problem is consistency maintenance where, given a solution $V$, a partial instanciation $\delta_V$ of a subset $\delta_X$ of $X$ (which stands for the modifications made to the given solution) and/or a set of added constraints $\delta_C$, we want to find another solution $V'$ so that:

\[
\begin{align*}
\forall x_i \in \delta_X, \; & V'(x_i) = \delta_V(x_i) \\
\forall c_i \in \delta_C, \; & (V'(x_{i_1}), \ldots, V'(x_{i_p})) \in R(c_i)
\end{align*}
\]

Moreover, it is generally desirable for $V'$ to be close to $V$ according to some semantics (for example, we may wish that the number of changed variables is as small as possible). This problem is usually solved by an incremental revision of the initial solution using techniques such as the propagation of known values [16]. Those techniques do not ensure completeness: there is no guarantee that a solution, if it exists, will be found. Nevertheless, they remain attractive because of their complexity which is linear in the number of variables.