Non-binary Constraints

Christian Bessière

LIRMM-CNRS
161 rue Ada, 34392 Montpellier, France
bessiere@lirmm.fr

Abstract
Since the origins of the constraint satisfaction paradigm, its restriction to binary constraints has concentrated a significant part of the work. This is understandable because new ideas/techniques are usually much simpler to present/ elaborate by first restricting them to the binary case. (See for example the arc consistency algorithms, such as AC-3 or AC-4, which have been presented first in their binary version \cite{10,12}, before being extended to non-binary constraints \cite{11,13}.) But this inclination has highly increased in the early nineties. Authors indeed justified this restriction by the fact that any non-binary constraint network can polynomially be converted into an equivalent binary one \cite{6,8,5,19}. And, in most cases, they never extended their work to non-binary constraints.

Up to now, constraint reasoning has generated robust formal definitions (local consistencies, etc.), original resolution methods (filtering, look-back schemes, decomposition techniques, heuristics, etc.), and theoretical results on tractable classes. They were proved useful on many academic problems, and are probably at the origin of the success of this area. But now, constraint reasoning is going towards its maturity, and should then be able to propose efficient resolution techniques for real-world problems, for which the binary conversion is sometimes/often impracticable. This observation, stressed by van Beek in its CP’96 invocation, has started having its effect since papers especially devoted to non-binary constraints or to their binary conversion begin to appear in this end of decade \cite{4,1,22,2}.

Dealing with non-binary constraints for real-world problems raises questions that do not appear as crucial on binary constraints. Applying filtering techniques during search is central in the efficiency of constraint satisfaction algorithms. But applying filtering on non-binary constraints is much more expensive than on binary ones. The weaknesses of the algorithms appear more accurately. Does this mean that we must circumscribe filtering to its weakest form, as in the classical extension to non-binary constraints of the forward checking algorithm? The answer is no. This algorithm is much more subject to thrashing than on binary networks \cite{1}. Local consistencies, which are one of the strengths of constraint reasoning, must also be applied in the non-binary case. But they have perhaps to be seen in a new way.
Perhaps we should accept the idea that the constraint solving tool of the next years will apply different levels of local consistency on different constraints at each node of the search tree, and more, a different algorithm for the same level of consistency depending on the constraint on which it applies. In the first case, the differences can come from the position of the constraint w.r.t. the current node.\footnote{This is already done in the search algorithm forward checking \cite{7}, which does not apply arc consistency on all the constraints, as opposed to algorithms such as really full look ahead \cite{15} or MAC \cite{20}.} or from the constraint type itself.\footnote{In \cite{18}, a filtering algorithm is proposed for a particular type of constraint. It achieves something which is stronger than arc consistency but not equivalent to any level of local consistency already known.} In the second case, specific constraint semantics can make efficient adapted algorithms applicable.\footnote{For the binary case, this approach has already been addressed in extensions of AC-4 \cite{14}, in AC-5 \cite{23}, and in AC-Inference \cite{3}.}

Because of this already mentioned high complexity of applying local consistencies on non-binary constraints, the theoretically wide expressive power of constraint reasoning is perhaps not as wide as that in practice. If applying local consistency on a given constraint is extremely expensive even for low levels (arc consistency is usually the standard level), it is as if this constraint was not allowed by the model. This leads us to the modelling aspects of constraint reasoning. Modelling a problem as a constraint network requires finding the “best” representation of the original knowledge. The criterion under which to decide which representation is better than the others is the efficiency of the available resolution technique to solve it \cite{16,21}. If no acceptable representation is found, this can be because for some of the original constraints, neither their decomposition in simple sub-constraints nor their direct integration were satisfactory. The former reason is usually due to a loss of globality in the decomposition. (Reasoning locally on the sub-constraints of the decomposition cannot detect inconsistencies.) The latter comes from the fact that no efficient filtering algorithm dedicated to this constraint exists while generic filtering algorithms are very time-consuming when applied to it. If we want to solve the original problem, an alternative is to write a dedicated filtering algorithm efficient enough to be incorporated in the solving engine. This has the clear advantage of saving the semantics of the constraint, and thus the potential ability to deal globally with it. We know indeed that the more globally we deal with locality, the more we are able to detect inconsistencies. This approach has been initiated with success on the very common alldiff constraint \cite{17} for which the decomposition in a clique of binary “\(\neq\)” constraints was almost unaffected by arc consistency processing. Even in paradigms such as SAT, where the simplicity and genericity of the model is claimed as its strength, recent works have shown the advantage of making global inferences specific to the problem type before losing all the structure in the SAT encoding \cite{9}. After a few years, that approach will of course have to deal with the issue of the size of the constraint solving engine. The number of different constraints added with their associated filtering algorithm will probably