Towards a Formal Framework for Comparing Constraint Satisfaction Problem Formulations

James E. Borrett & Edward P. K. Tsang

Dept. of Computer Science
University of Essex
Wivenhoe Park
Colchester CO4 3SQ

email: {jborrett, edward}@essex.ac.uk

Technical Report CSM-264, 6th March 1996

Abstract: Current techniques for comparing constraint satisfaction problem formulations have depended to a large degree on the experience of the problem solvers concerned. At the same time there is the potential for spectacular differences in the effectiveness of search algorithms on different formulations of a problem.
In this paper we propose a formal framework for comparing different constraint satisfaction problem formulations. We believe our approach is a significant step towards the goal of improving the reliability of constraint satisfaction problem formulation selection.

1. MOTIVATION

Once the decision has been taken to adopt the constraint satisfaction approach to solving a problem the next step is to formulate it as a constraint satisfaction problem (CSP). A CSP can be defined by the triple <Z, D, C> where Z represents the set of variables in the problem, D maps every variable to a domain of values and C represents a set of constraints on the variables (Tsang 93). The process of formulating the problem then becomes a case of identifying these three component parts of the CSP from the problem description. This is an important stage in the overall process of solving a problem since the choice of formulation can dictate the effectiveness of the subsequent search.

Current methods for generating and then evaluating the merits of different constraint satisfaction problem ZDC formulations[1] tend to be very informal and are somewhat ad-hoc in nature. Some very general guidelines have been suggested such as in (VanHentenryck 89) where the removal of symmetries by the use of redundant constraints is recommended. In (Chamard et al 95) it is suggested that the pruning power of the individual constraints should be as high as possible. However, even the few guidelines that do exist appear far too simplistic.

As a result of this lack of formality at the formulation stage, the process of arriving at the most appropriate ZDC formulation for a given problem can be dependent on the experience of a problem solver, or can be the result of much trial and error and prototyping. Indeed, in (Chamard et al 95) it is suggested that the process of formulating a CSP is an iterative one. If the process can be structured in some way, problem solvers, both experienced and novice, would have the scope for generating and comparing ZDC formulations in a more reliable and reasoned way. In this paper we shall describe a formal framework for comparing ZDC formulations.

The idea of comparing different formulations was considered as early as 1968 (Amarel 68) in the context of search in general. In that paper, Amarel looks at the different formulations of the Missionaries and Cannibals problem and briefly discusses the idea of choosing a formulation based on size of what he calls "*N-State*" space. Similarly, Korf (Korf 80) looks at different formulations of several search problems. He advocates the idea of using "*information content*" as a basis for comparing problem formulations. However, no detail is given in either of these works of how the comparison might be achieved.

A formal framework has many potential benefits which result from structuring the process of comparing ZDC formulations. One of these is that it gives us the potential for the automatic assessment of ZDC formulations. Korf (Korf 80) states that "*to completely automate the process of finding good representations for problems will require a method for evaluating the efficiency of a representation relative to a particular problem solver*", in the context of search in general. We are interested in devising such an approach in the domain of constraint satisfaction problems, and in particular ZDC formulations. We hope that once candidate formulations have been generated the process of evaluating the likely best choice could then be automated using a computer program. It can also be expected that a formal method will produce repeatable conclusions. This is important in developing confidence in the method.

Our aim is to develop a formal framework with which we can select among problem formulations and give the reasons for that selection. These reasons would give problem solver hints as to possible further refinements to the chosen formulation, such that it can be improved. This relates again to (Korf 80) where the idea of automatically refining domain specific search problem formulations is

[1] An instance of a CSP formulation shall be referred to as a ZDC formulation.

considered using transformations. Furthermore, a formal approach which can give reasons for suggesting a preference for a given ZDC formulation has the potential to provide us with ideas for new transformations.

All of these issues give weight to the notion that a formal framework for the comparison of constraint satisfaction problem formulations is useful and worth investigating. Even if the method is only applicable to a selection of problems it can still be useful. The overriding aim of our approach is to ensure that the maximum amount of refinement possible is incorporated into the formulation stage before search begins. In (Amarel 68) it is stated that "*the choice of appropriate representations is capable of having spectacular effects on problem solving efficiency*". An example of this effect with CSPs can be found in (VanHentenryck 89) where redundant constraints are added to a formulation of the Magic Number Series puzzle with dramatic effect. In (Puget 93) the potential for formulation manipulation is demonstrated by the addition of redundant constraints in formulations of a range of small problems to remove symmetry in their search spaces. Thus by studying the ways in which simple problems are formulated we can learn useful ways of selecting the most appropriate formulations when dealing with more complex problems.

A further wish for any formal method is that the final choice of formulation be, as far as possible, algorithm independent. There are many different algorithms and paradigms[2] for solving constraint satisfaction problems and there is the possibility that different formulations could be more favoured by one particular paradigm. However, we believe that it should be possible to find unanimous choices for algorithms within the same paradigm, and it is for this reason that we shall initially restrict our work to selecting the best ZDC formulation of problems for the group of complete backtracking algorithms such as forward checking (Haralick&Elliott 80), back-marking (Gaschnig 77), conflict-directed back-jumping (Prosser 93) and the many hybrids which derive from these (Prosser 93). This restriction does not detract from the usefulness of our work since some of these algorithms are used in commercially available constraint solving packages such as ILOG Solver and CHIP.

Another point to consider in our work is that we shall ignore the cost to generate formulations when comparing them. The cost to generate a formulation is for the problem solver to consider. For example, the application of interchangeability techniques (Freuder 91), (Haselbock 93) to transform a formulation has a processing cost associated with it. We are more concerned with the final effects of the generation process and evaluating whether or not the final output is a useful one.

1.1 Algorithm Dependence

It is clear that, given the large number of algorithms which have been developed for solving constraint satisfaction problems, it can be expected that some will be more effective at exploiting certain features of CSPs than others. In (Tsang et al 95) it was shown that different algorithms have domains of superiority when their performances are ranked, in the domain of binary CSPs. In other work, algorithms have been developed to take advantage of specific features of problems, such as the tree clustering algorithm (Dechter&Pearl 88).

1.2 Algorithm Independence

While different algorithms may be more suited to different formulations it is often the case that, given a domain of ZDC formulations, some of them are clearly bad for all algorithms. Conversely, some of

[2] see (Kumar 92) (Tsang 93) for overviews of some of these.

the formulations may be clearly better for all algorithms being considered. Examples of such instances are given below.

- *Single variable problems* - using the magic number series problem (VanHentenryck 89) as an example we could suggest two possible formulations. The problem is to find a sequence of numbers length N, such that each number in the sequence is equal to the number of times the index of that position occurs in the sequence. The first formulation we propose is to have the individual digit positions as variables, values 0 to N-1 as their domains and a global constraint on their values. The second formulation we propose uses a single variable with a domain of the possible N^N sequences and a unary constraint expressing the sequence requirements. This second formulation is likely to be a bad choice for all constraint satisfaction algorithms since it effectively reduces to generate and test, with few solutions. No constraint processing techniques can be applied efficiently.
- *Node consistent single variable problems* - we can use the magic number series problem once again, with an additional formulation. The third formulation is a node consistent version of the single variable formulation. This is obviously a good formulation for all algorithms, if it can be generated within a reasonable amount of time (which is unlikely to be the case), since it is effectively the set of all solutions.
- *Symmetry* - the well known 8-Queens problem is to place 8 queens on a 8×8 chess board without attacking each other. One possible formulation of this problem is to use 8 variables, the 8 queens, with domains as the 64 squares on the board. This is a symmetrical ZDC Formulation since any solution is still a solution if values are interchanged. An alternative, less symmetrical, formulation is to have variables which represent a queen for each row. Their domains are the column in which that queen is placed. It is clear that the second formulation can be expected to be a much better choice for all algorithms since the effective size of the search space has been reduced considerably while some duplicated solutions have been eliminated.
- *Increasing problem structure* - the SENDMORY problem is a well known crypto-arithmetic puzzle. The solution to the puzzle is to assign a unique digit to each of the letters in the addition "SEND+MORE=MONEY". One formulation for this problem is to have variables which represent the letters, domains representing the digits and a global constraint stating the addition which must hold. An alternative is to introduce redundant carry variables into the equation. This allows us to break down the global constraint and introduces more variation in the structure of the problem. It is quite likely that this second formulation will be the better choice since there is the potential for the algorithms to take advantage of this increased diversity in structure of the formulation.

From these examples we can see that there are formulation decisions which can be considered good and bad, independent of the algorithm to be used. To an experienced problem solver this distinction may be obvious, but to the novice problem solver and for more complex problems, the distinction becomes much less clear.

1.3 Algorithm Independent Measures

In section 1.2 we showed how some formulations can be selected independent of the algorithm to be used to solve the problem. In order to structure the process of making such selections we need to isolate the elements of formulations which reflect this algorithm independence. We believe that this is one of the main tasks which needs to be undertaken in order for the CSP community to develop a formal framework for comparing formulations. More details of this concept are given in section 2.

1.4 Criterion for Success

Having outlined our objective, it is useful to clearly set out the goal of our method. We believe that a formal framework for comparing ZDC formulations should fulfil the following criterion;

> "*A formal framework for comparing ZDC formulations should, for the average case or better, enable us to select the formulation which gives the best algorithm performance*"

Given a domain of ZDC formulations, we wish to be able to identify the ones which algorithms will be able to solve in the shortest time. We suggest that CPU time is a reasonable criterion for defining best since this trend can be evaluated across formulations for any given algorithm. Criteria such as compatibility checks would be less reliable since different formulations can be expected to use different constraints which may not have the same evaluation time.

2. THE FORMAL FRAMEWORK DESCRIBED

In this section we outline a framework for the comparison of ZDC formulations. We believe that this framework provides the CSP community with a series of foci, where certain elements of the framework will need to be further developed.

The primary object of interest in the formal framework is a domain of all possible ZDC formulations, $\mathbb{F}$, for the problem in question. Given a subset of the formulations in $\mathbb{F}$ we want to select the best one according to the criterion given in section 1.4. We see the basic element of this process as a pairwise comparison of the individual formulations. These comparisons can be applied repeatedly across the subset of ZDC formulations until an overall conclusion can be made. The outcome of each pairwise comparison is a conclusion. We therefore define the set of possible pairwise conclusions, $\mathbb{C}$, to be;

$$\mathbb{C} = \{true, false, no_conclusion\}$$

where *true* and *false* give the direction of a decision and *no_conclusion* means that there is no discernible conclusion. An example of conclusions being generated is given later in this section.

In order to draw any conclusions it is necessary to build up *quantitative descriptions* of problem formulations. There are two possible approaches to building quantitative descriptions. The first is to use a single measure. Such a single valued approach would be useful in that it would provide a simple decision for each comparison. However, this approach requires the development of a measure which encapsulates all algorithm independent features of a ZDC formulation in one enumeration and hence provides us with very little scope for explanation of the conclusion made. It is also questionable whether the single value approach is at all feasible.

An alternative to the single value description is to develop a vector of measures which take into account the many different features of a ZDC formulation. By using the multi valued approach we gain many advantages such as the potential for providing a degree of explanation for any successful comparison. The process of development of such a vector is also more flexible and can become an incremental one. Effectively the more elements to the vector we have, the more possibilities there are for describing ZDC formulations. As a result, the resolution of comparison will be improved.

For the reasons outlined above we believe the vector approach to be the most promising. We define a vector of measures, $\mathcal{M}$, as;

$$\mathcal{M} = (m_1, m_2, \ldots, m_k)$$

Each element of the vector can be expressed as;

$$m_i: \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{C} \qquad 1.$$

An example of a measure comparing two formulations, F_1 and F_2 to produce a conclusion could be;

$$m_i(F_1, F_2) = \textit{true}$$

We take this to mean that using measure m_i, formulation F_1 is better than F_2.

The vector $\mathcal{M}$ is the first part of our framework which we see as being a focus of research for the CSP community. We need to develop the individual elements of $\mathcal{M}$.

By extending this idea for the entire vector $\mathcal{M}$, the result of a pairwise comparison of ZDC formulations would be a vector of conclusions. We shall describe this as the *enumerated measure description*, E, for a comparison.

$$E = (e_1, e_2, \ldots, e_k) \qquad \text{where} \qquad \forall i : e_i \in \mathbb{C}$$

If $\mathbb{E}$ is the set of all possible enumerated measure description then we have;

$$\mathcal{M}: \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{E} \qquad 2.$$

In other words, given two ZDC formulations, application of the measure vector gives us an enumerated measure description. One final step remains before we can reach an overall conclusion about the comparison. That step is to interpret what a description, E, tells us. To make use of E we need a *resolution function*, $\mathcal{B}$, which maps an enumerated measure description to a conclusion:

$$\mathcal{B}: \mathbb{E} \rightarrow \mathbb{C} \qquad 3.$$

In other words $\mathcal{B}$ resolves the vector E and provides a conclusion about the comparison which has been performed. The function $\mathcal{B}$ is the second of the elements that we identify as a need for the CSP community to develop.

To incorporate the notion of ZDC transformations into the framework we add a further element. A transformation function, $\mathcal{T}$, can generate a new formulation in the domain $\mathbb{F}$. This is expressed as;

$$\mathcal{T}: \mathbb{F} \rightarrow \mathbb{F} \qquad 4.$$

It is obvious that the goal of any function $\mathcal{T}$ should be to produce a new formulation which is an improvement on the original formulation. According to our framework a useful $\mathcal{T}$ should satisfy the following ;

$$\mathcal{B}(\mathcal{M}(\mathcal{T}(\mathrm{F}), \mathrm{F})) = true$$

We have now described the basic elements of our framework for comparing ZDC formulations. As we have indicated, of these basic elements much work needs to be done by the CSP community to find useful instances of $\mathcal{M}$, $\mathcal{B}$ and $\mathcal{T}$.

3. DISCUSSION

Structured techniques for comparing problem formulations have not previously been developed in the field of constraint satisfaction. The importance of introducing structure to the process is evident from the wide range of possibilities that exist for formulating many constraint satisfaction problems. In this paper we have proposed a formal framework which provides a first step towards achieving that aim.

Two elements of the framework are still to be developed. The first of these is a vector of measure functions, $\mathcal{M}$, for describing ZDC formulations. As noted in section 1.3, these measures need to reflect algorithm independent qualities of formulations. One example of a candidate measure is that of T-Factor (Borrett&Tsang 95). T-Factor is a effectively a measure of the expected solution density of a ZDC formulation when it is soluble, and a qualitative measure of the level of insolubility for insoluble problems. It can be expressed as;

$$\text{T-Factor} = \left\{ \begin{array}{ll} \dfrac{E}{S} & (\text{for } E \geq 1) \\ \dfrac{1}{E \times S} & (\text{for } E < 1) \end{array} \right\}$$

Where E is the expected number of solutions for the formulation and S is the size of the search space.

The level of solution density is a contributory factor to the effectiveness CSP solving algorithms, so T-Factor fulfils our criterion for measure functions. It is also a value which can be compared across different classes of CSP. We can incorporate this into a measure vector element, m_j as;

$$m_j(\mathrm{F}_1, \mathrm{F}_2) = \left\{ \begin{array}{ll} true & \text{for TFactor(F1)} > \text{TFactor(F2)} \\ false & \text{for TFactor(F1)} < \text{TFactor(F2)} \\ no_conclusion & \text{otherwise} \end{array} \right\}$$

The other part of the framework which the community needs to develop is vector resolution functions, $\mathcal{B}$. One possible approach to this stems from the observation that elements of $\mathcal{M}$ may all provide the same conclusion. Other possible scenarios arise when only a few of the elements differ. When the conclusions in the enumerated measure description are of varying nature we must apply more complex strategies. An example of this could be to use a hierarchical ordering of the elements.

The individual conclusions would then be consulted according to that hierarchy until a reasonable general conclusion can be made.

One aspect of the framework which has already been addressed by researchers in many ways is in the domain of ZDC Transformations, $\mathcal{T}$. Examples include the removal of redundant values in problem reduction(Tsang 93), removal of redundant constraints (Dechter&Dechter 87), the generation of dual problems (Dechter&Pearl 88), consistency techniques (Mackworth 77), (Mohr&Henderson 86), and interchangeability techniques (Freuder 91), (Haselbock 93) (Freuder,Wallace&Sabin 95).

There is much work to be done to refine our proposed formal framework. However, our method, while very simple in its concept, gives the CSP research community a clear direction towards the goal of comparing CSP formulations in a reasoned and effective way.

Acknowledgements

The authors would like to thank Jim Doran and Rachel Cardel-Oliver for their useful comments on this work. We are also grateful to the members of the constraint satisfaction research group for their useful discussions.

References

Amarel, S., On Representations of Problems of Reasoning about Actions, Machine Intelligence, vol 3, 131-171, 1968

Borrett, J. E. & Tsang, E. P. K., 1995, On the Selection of Constraint Satisfaction Problem Formulations, Technical Report CSM254, Dept. of Computer Science, University of Essex, Colchester

Chamard, A., Fischler, A., Guinaudeau, D-B. & Guillard, CHIC Lessons on CLP Methodology, EPSRC Technical Report, 1995.

Dechter, A. & Dechter, R., Removing Redundancies in Constraint Networks, Proceedings 6th National Conference on Artificial Intelligence, 105-109, 1987

Dechter, R. & Pearl, J., Tree-Clustering Schemes for Constraint Processing, Proceedings 7th National Conference on Artificial Intelligence, 150-154, 1988

Freuder, E. C., Eliminating Interchangeable Values in Constraint Satisfaction Problems, Proceedings 9th National Conference on Artificial Intelligence, 227-233, 1991

Freuder, E. C., Wallace, J. & Sabin, D., Generalisation and Abstraction for Constraint Satisfaction, in Proceedings Constraint-95, an International Workshop on Constraint-Based Reasoning, FLAIRS-95, 16-24

Gaschnig, J., A General Backtrack Algorithm that Eliminates most Redundant Tests, Proceedings 5th International Joint Conference on Artificial Intelligence, 457, 1977

Haralick, R. M., and Elliott, G. L., Increasing Tree Search Efficiency for Constraint Satisfaction Problems, Artificial Intelligence, vol 14, 263-313, 1980

Haselbock, A., Exploiting Interchangeabilities in Constraint Satisfaction Problems, Proceedings 13th International Joint Conference on Artificial Intelligence, 282-287, 1993

Korf, R. E., Toward a Model of Representation Changes, Artificial Intelligence, vol 14, 41-78, 1980

Kumar, V., Algorithms for Constraint Satisfaction Problems: A Survey, AI Magazine, vol 13, 32-44, 1992

Mackworth, A. K., Consistency in Networks of Relations, Artificial Intelligence, vol 8, 99-118, 1977

Mohr, R. & Henderson, T. C., Arc and Path Consistency Revisited, Artificial Intelligence, vol 28, 225-233, 1986

Prosser, P., Hybrid Algorithms for the Constraint Satisfaction Problem, Computational Intelligence vol 9, 268-299, 1993

Puget, J-F., On the Satisfiability of Symmetrical Constrained Satisfaction Problems, ISMIS 1993

Tsang, E. P. K., Borrett, J. E. & Kwan, A. C. M., An Attempt to Map a Range of Constraint Satisfaction Algorithms and Heuristics, in proceedings 10th Biennial Conference on AI and Cognitive Science, Society for the Study of Artificial Intelligence and Simulation of Behaviour, 1995, 203-216

Tsang, E. P. K., Foundations of Constraint Satisfaction, Academic Press, London 1993

Van Hentenryck, P., Constraint Satisfaction in Logic Programming, MIT Press, 1989