Representations in Constraint Programming

Christopher Jefferson

June 1, 2009

Constraint programming is a powerful and general purpose tool which is used to solve a large range of combinatorial and real-world problems. CP solvers combine a number of powerful and generic algorithms, which when used together can often solve problems infeasable in other frameworks.

The major practical issue which stalls adoption of constraint programming by a wider audience is that transforming a problem from a high-level description into a format suitable for a constraint solver, called *modelling*, is more of an art than a science. Hence, modelling can only be accomplished well by expert practitioners. Often small changes, which to the untrained user may appear useless, can lead to huge reductions or increases in the time taken to solve a problem.

The first, and arguably most important, decision when modelling a problem is to choose the type of variables which will be used. Most constraint systems provide only a small number of variable types; usually integers, matrices and possibly sets. Users must transform the high-level abstract types from their problem into these basic types, and then map the constraints down onto these types. All this must be accomplished using only the set of constraints provided by their solver. This thesis provides the first complete, generic method for comparing all the representations of a particular variable. In doing so it provides a number of key insights which improve the state of the art in automating the modelling process.

Choosing the "best" representation of a high-level variable, such as a timetable or partition, in CP is extremely difficult; with many possible conflicting definitions of best. In general keeping the number of variables and constraints used small will allow the solver to work faster. Designing the representation so that propagators for global constraints can be used can also improve performance. Often different representations have different strengths,

so multiple representations are used, and *channelled* together.

Channelling brings its own problems, including how to link together and search over the multiple representations. This thesis provides the first comprehensive study of complex forms of channelling. While all published examples of channelling show decreased search size, this thesis demonstrates how channelling can lead to massive increases in search when bad representations are used, and how to detect such cases automatically.

The biggest problem with many previous methods of comparing representations is that they inevitably become tied up in the global constraints which have been implemented for a particular representation, and their complexity. This ever-changing list makes it hard to build a formalism which does not change between different solvers, and even different versions of the same solver.

This thesis provides a radical and original theory for representations of variables in constraint programming. This theory abstracts away from details of the exact list of constraints implemented in a solver, and instead studies variables in isolation. This allows for new theories and insights which are shown to be of practical importance to modelling and automating the modelling process in particular. While this framework must be used with care, many of the results it generates provide new powerful insights which have surprised many expert constraint modellers into why some variable representations perform well in practice.

Representations can be boardly split into two categories, *internal* and *external* representations. An *external* representation splits one variable in a CP problem into multiple variables, which are then given to the solver. For example, a subset of the integers one to ten might be represented by ten Boolean variables, where the i^{th} boolean is true if and only if i is in the set. *Internal* representations are used by the solver internally and may not be expressible in a traditional CP framework, for example representing an integer during search by storing it's current upper and lower bound.

The line between internal and external representations is often blurred in practice, for example previously external representations for sets have in recent years been moved inside the solver for improved efficiency. In this thesis, both *internal* and *external* representations are defined and compared within one unified framework.

While the theory of this thesis can be, and is, applied to many types, the main types considered in this thesis are sets and multisets. The two main representations of sets and multisets considered, called the *occurrence* and

explicit representations, are similar to the representations of other mathematical types, such as graphs, and so many of the results generalise easily to other types.

The *occurrence* representation of a set or multiset S provides one integer variable for each $s \in S$, denoting how many occurrences there are of s. The *explicit* representation on the other hand provides a (possibly ordered) list, explicitly giving the elements of the set or multiset. These two representations are the fundamental basis of many of the set and multiset representations occurring in the modelling literature.

One of the highlights in this thesis is a framework for finding if a representation is the best possible for a particular constraint. If a representation is *perfect* for all the constraints in a problem, then the search tree generated when solving the problem is minimal. General theorems which show when a representation is perfect provide a firm theoretical basis for comparing representations, and finding when a representation is the best for a particular problem.

The *occurrence* representation is perfect for a large variety of constraints, including in particular any constraints built with the operators $\in, \subseteq, =, \cup$ and $\cap$. The constraint which occurs most frequently missing from this list is $|S|$, the size of a set or multiset. It is *not* sufficient to simply add a variable to the occurrence representation which represents the size of the set, as many solvers do, to make a representation which is perfect over a large range of set constraints. It is proved in this thesis that for many lists of constraints which include $|S| = c$, any representation which is perfect on the full list of constraints must have an NP-hard propagator on at least one of the constraints.

While there may exist better representations than the *occurrence* in practice, this means that for the majority of problems on which the occurrence representation is not perfect, no other representation which can be implemented in polynomial time will be. This places a strong practical and theoretical limit on the performance of set and multiset representations.

One important and well researched area of constraint programming is symmetry breaking. Despite the apparent maturity of this field, it became clear during the writing of this thesis that various basic definitions, and how they interacted with representations and constraint problems, were not properly understood. This lead into my collaborative research into symmetry definitions [1], which lead to a best paper award at CP 2005.

This thesis goes further into symmetry, showing that a large proportion,

and in many problems all, of the symmetry is introduced in the modelling process. This thesis provides new, powerful symmetry breaking methods which can be applied directly to such symmetries. These representation-aware symmetry breaking methods outperform traditional generic methods. This work was published at ECAI [4].

This thesis draws special attention to channelling, which involves joining together multiple representations of the same high-level variable. Previously, channelling has been used to great success when multiple models of a problem exist. This thesis was the first to produce an in-depth study of the theory of channelling, showing that channelling arbitrary models together does not help performance, and often performs much worse than any of the models in isolation. Many existing channelling results fall into a special very limited category, where search is not increased in size, but requires very similar models to operate correctly.

One important feature of the work in this thesis is that it provides a well-defined system to compare and prove results about representations, and constraints imposed on them. This understanding of high level types contributed substantially to the design of the Essence language [2], a powerful and expressive language for defining problems. During my PhD the work in this thesis has been a major influence on the design of Conjure [3], a rule system for correctly transforming problems in Essence into a format suitable for constraint solvers. Without a solid understanding and theory of representations, neither of these systems could exist in their current form.

The theory of representations in this thesis was used to design the Minion constraint solver, which I am the primary architect of. Minion is discussed in [5], which was one of the 10 best papers at ECAI 2006. The original ECAI paper for Minion was the first paper in CP which discussed in depth the issues of implementing apparently simple types, such as integers and booleans, and Minion was unusual when released for providing to users and automated modelling systems an explicit choice in how variables are represented, demonstrating that even for integers there can be no one best representation to all situations.

This thesis provides the first unified general framework for defining and comparing representations in constraint programming. It provides wide-ranging insights and theory, effecting many different areas of constraint programming. As well as directly providing results in the areas of automated modelling and solver design, the general principles of this thesis can be applied to many disparate areas of research, as shown by the range of topics I

have considered during my thesis.

References

[1] David A. Cohen, Peter Jeavons, Christopher Jefferson, Karen E. Petrie, and Barbara M. Smith. Symmetry definitions for constraint satisfaction problems. *Constraints*, 11(2-3):115–137, 2006.

[2] Alan M. Frisch, Warwick Harvey, Christopher Jefferson, Bernadette Martínez Hernández, and Ian Miguel. Essence : A constraint language for specifying combinatorial problems. *Constraints*, 13(3):268–306, 2008.

[3] Alan M. Frisch, Chris Jefferson, Bernadette Martínez Hernández, and Ian Miguel. The rules of constraint modelling. In *Proceedings of IJCAI 2005*, pages 109–116, 2005.

[4] Alan M. Frisch, Christopher Jefferson, and Ian Miguel. Symmetry-breaking as a prelude to implied constraints: A constraint modelling pattern. In *Proceedings of ECAI 2004*, pages 171–175, 2004.

[5] Ian P. Gent, Christopher Jefferson, and Ian Miguel. Minion: A fast scalable constraint solver. In *Proceedings of ECAI 2006*, pages 98–102. IOS Press, 2006.