Transformations of CSPs to Regular CSPs

Sven Loeffler, Ke Liu, Petra Hofstedt

Brandenburg University of Technology Cottbus-Senftenberg, Germany

Abstract. With the development of constraint programming (CP), NP-complete problems can be molded and solved in a powerful declarative way. Until today, the most and with biggest success researched problems in constraint programming are rostering, graph coloring and satisfiability (SAT) problems [4].

In an ideal declarative programming world, the program execution speed of a constraint problem does not depend on the syntax of semantically equivalent problem descriptions. However, as far as we know, this is not the reality for general constraint problems.

Generally, there are several possible manners to model a given constraint problem since the problem can be described from different perspectives, therefore, two or more different sets of constraints, which might consist of various constraints, could be used to attack the same problem. The resulting constraint problems can lead to very different execution speed and execution behavior. The main reason for this is that there are different constraints with several propagation algorithms, which can describe the same restrictions. Consequently, the replacement of constraints by constraints of another type (in this case regular constraints) and the combination of multiple constraints to a new constraint can lead to a faster solution finding process and a removement of undesirable redundancy.

In our context, undesirable redundancy means the redundancy which occurs through the use of similar or overlapping constraints which decrease the solution speed. In contrast to this, redundancy is often used in a targeted manner to improve the solving speed of a constraint problem. We call this positive redundancy. The mix of removing undesirable redundancy and adding positive redundancy might also be a topic of future research, but until yet we consider only the removement of undesirable redundancy.

Constraint Satisfaction Problems (CSP). The goal of our work is to improve the solving speed of a Constraint Satisfaction Problem (CSP) which is defined in the following way.

Definition 1. ***CSP*** *[1] A constraint satisfaction problem (CSP) is defined as a 3-tuple* $P = (X, D, C)$ *with*

$X = \{x_1, x_2, \ldots, x_n\}$ *is a set of variables,*
$D = \{D_1, D_2, \ldots, D_n\}$ *is a set of finite domains where* D_i *is the domain of* x_i*,*
$C = \{c_1, c_2, \ldots, c_m\}$ *is a set of primitive or global constraints containing between one and all variables in* X*.*

We transform several kinds of (global and sets of local) constraints into regular membership constraints and replace then the original constraints by the created regular constraints in a CSP. For selected problems (like rostering problems) we increased the solution speed of CSPs, removed undesirable redundancy and decreased the number of unnecessary backtracks by the replacement of some or all constraints of a CSP with regular constraints followed by the combination of such created multiple regular constraints into a new, more precise regular constraint. The regular constraint and its propagation algorithm [6,5,2] provide on the basis of this approach.

Regular transformations. According to our definition, a CSP P has a fixed number n of variables $X = \{x_1, \ldots, x_n\}$ and the domains $D = \{D_1, \ldots, D_n\}$ of these variables are finite, it follows that the potential solution space of the CSP P is limited by l (see Eq.1).

$$l = \prod_{i=1}^{n} |D_i|. \tag{1}$$

A limited number of solutions can be enumerated by a regular language, and based on the Myhill-Nerode theorem [3] every regular language has at least one automaton which describes this language. A consequence of this is that for every CSP P at least one regular CSP P_{reg} exists which is declaratively equivalent to P and contains only one regular constraint. We name such a CSP, which only contains (one or more) regular constraints, a regular CSP.

If all solutions of a CSP P are known, then such a regular CSP $P_{reg} = (X, D, C_{reg})$ can generated by the enumeration of all solutions of P. In practice, however, this is not useful because we intend to use the regular CSP P_{reg} to find one or all solutions of P faster.

This is the reason, why we are interested in direct transformations of constraints into regular constraints. For some global constraints, among them *count*, *global cardinality constraint*, and *table* constraints we defined corresponding efficient transformations. In addition to this we defined a transformation for sub-CSPs (in particular binary ones) over sub-sets of variables $X' \subseteq X$ and constraints $C' \subseteq C$ of a CSP $P = (X, D, C)$.

After the transformation into regular constraints it is possible to use the automaton methods *intersection* and *minimize* to combine different automatons to a new automaton and also different constraints to one new constraint. By doing this, redundancy between to constraints can be removed. This approach allows it, especially, to combine constraints of original different kinds together, for example *count* with *table*. In our test cases we have experienced a significant reduction of the size of the necessary search tree and a great improvement of the solving speed of CSPs.

Summary. We have presented a new approach for the optimization of general CSPs using the regular constraint.

It has shown that a general CSP can be transformed into a regular CSP and some special global constraints can be transformed directly efficiently into regular constraints. By the use of automata intersection and minimization regular constraints can be minimized. The evaluation of this approach by a rostering example has shown a significantly reduction of the size of the search tree and a great improvement of execution time.

Our investigations support that our approach can be applied successfully when considering sub-problems of a potentially large CSP P and, thus, subsets of its variables X. The transformation of only sub-problems (instead of P completely) is, thus, much faster and, nevertheless, leads to a reduction of the number of constraints (also the number of backtracks) and of undesirable redundancy which, altogether, improves the solution speed.

Future work. The objectives of the future work are to find more direct transformations for global constraints, investigating promising variable orderings to optimize the size of the DFAs and, similarly, variable and value orderings for the regular constraints. We wolud also research potential benefits from decomposition of automata, parallelization of the transformations and the solving process. Finally, we want to find out about general (static) criteria to decide when to apply the approach as well as extracting promising application areas in general.

References

1. Dechter, R.: Constraint processing. Elsevier Morgan Kaufmann (2003)
2. Hellsten, L., Pesant, G., van Beek, P.: A domain consistency algorithm for the stretch constraint. In: Wallace, M. (ed.) Principles and Practice of Constraint Programming - CP 2004. Lecture Notes in Computer Science, vol. 3258, pp. 290–304. Springer (2004)
3. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
4. Marriott, K.: Programming with Constraints - An Introduction. MIT Press, Cambridge (1998)
5. Paltzer, N.: Regular Language Membership Constraint. Seminararbeit, Universität des Saarlandes, Deutschland (2008)
6. Pesant, G.: A filtering algorithm for the stretch constraint. In: Walsh, T. (ed.) Principles and Practice of Constraint Programming - CP 2001. Lecture Notes in Computer Science, vol. 2239, pp. 183–195. Springer (2001)