GQR: A Fast Solver for Binary Qualitative Constraint Networks

Matthias Westphal and **Stefan Wölfl**
Department of Computer Science,
University of Freiburg,
Georges-Köhler-Allee, 79110 Freiburg, Germany
{westpham, woelfl}@informatik.uni-freiburg.de

Zeno Gantner
Information Systems and Machine Learning Lab,
University of Hildesheim,
Samelsonplatz 1, 31141 Hildesheim, Germany
gantner@ismll.uni-hildesheim.de

Abstract

Qualitative calculi are constraint-based representation formalisms that allow for efficient reasoning about continuous aspects of the world with inherently infinite domains, such as time and space. GQR (Generic Qualitative Reasoner) is a tool that provides reasoning services for arbitrary binary qualitative calculi. Given qualitative information expressible in a qualitative calculus, GQR checks whether this information is consistent w.r.t. the calculus definition. GQR employs state-of-the-art techniques in both qualitative and constraint reasoning, such as heuristic search and usage of known tractable subclasses. In contrast to specialized reasoners, it offers reasoning services for a variety of calculi known in the literature, which can be defined in a simple specification language. The main focus in the design and implementation of GQR is to provide a generic and extensible solver that preserves efficiency and scalability.

Introduction

Qualitative constraint calculi are representation formalisms for efficient reasoning about continuous aspects of the world, such as space and time. Qualitative information, when represented as constraint networks, can be checked for consistency by applying well-known constraint techniques. GQR (Generic Qualitative Reasoner) (Gantner, Westphal, and Wölfl 2008) implements such techniques for qualitative constraint networks in which only binary relations occur. GQR takes a calculus description and one or more constraint networks as input, and processes the networks using the symbolic path consistency method and heuristic backtracking. In contrast to specialized reasoners, it offers reasoning services for arbitrary binary qualitative calculi, which can be defined in a simple specification language.

When the development of GQR started, only reasoners for specific calculi, e.g. (van Beek and Manchak 1996; Nebel 1997), were available. In order to reason in a new calculus, one had to develop a new program, modify an existing one for a similar calculus, or encode it in a different logical framework for which solvers existed. Now similar research efforts exist, e.g., the qualitative algebra toolkit (*QAT*) (Condotta, Saade, and Ligozat 2006) and *SparQ* (Wallgrün et al. 2006). In contrast to those tools, the main focus in the design of GQR is to implement a *fast* and *extensible* generic solver, which preserves the efficiency of calculus-specific solvers as much as possible.

GQR has been applied (a) in a high-level agent control system implementing rule-compliant behavior of agents in sea navigation (Dylla et al. 2007) and (b) in the evaluation of different algorithms for application-specific customizations of qualitative calculi (Renz and Schmid 2007).

Qualitative Reasoning with GQR

A *(binary) constraint network* is defined by a set of variables taking values in a given domain and a family of binary constraint relations between pairs of variables (on this domain). The *constraint satisfaction problem* is to determine for a given constraint network, whether there exists an assignment to its variables such that all constraints of the network become satisfied. Since the domains considered in qualitative reasoning are usually infinite, constraint solving techniques need to be applied on finite, symbolic representations of constraint networks, namely, directed finite constraint graphs, where each edge is labeled by a set of relation symbols (each symbol represents a concrete binary relation on the domain). Sets of relation symbols are read disjunctively, that is, they can be used to express imprecise knowledge about the actual configuration.

For reasoning in GQR, we assume that the symbol set is equipped with an algebraic structure: A *(binary) qualitative calculus* is defined by a non-empty finite set B of symbols (referred to as *base relations*), a unary function ${}^{\smile} : B \to B$ (assigning to each base relation its converse), a binary function $\circ : B \times B \to 2^B$ (assigning to each pair of base relations their composition), and a distinguished element $\mathrm{id} \in B$ (the *identity relation*) such that some minimal requirements are met (e. g., $(a^{\smile})^{\smile} = a$, $\mathrm{id} \circ a = a \circ \mathrm{id} = a$; etc.). Formally, the Boolean algebra 2^B defines a non-associative relation algebra if the functions ${}^{\smile}$ and $\circ$ are extended to functions ${}^{\smile} : 2^B \to 2^B$ and $\circ : 2^B \times 2^B \to 2^B$ by: $r^{\smile} := \{b^{\smile} : b \in r\}$ and $r \circ r' := \bigcup_{b \in r, b' \in r'} b \circ b'$.

In GQR the constraint satisfaction problem is then solved on the symbolic level, that is, GQR checks whether the constraint graph is *consistent* in the sense that there exists a path-consistent refinement (cf. below) of the constraint graph in which each edge is a base relation. For many calculi this is sufficient to solve the satisfiability problem.

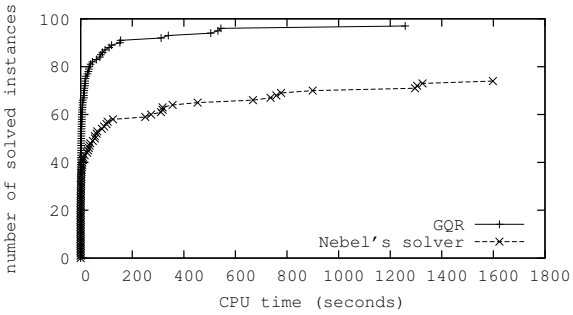

Figure 1: Comparing GQR to a solver for Allen's interval calculus. Instances were taken from the phase transition with 100 nodes each and an average label size of 6.5.

Implementation Details

GQR utilizes the symbolic path consistency algorithm and implements techniques that enhance backtracking search.

Path Consistency and Search. The (symbolic) path consistency algorithm manipulates a given constraint graph by successively refining the labels $r_{x,y}$ (on the edge from node x to node y) via the operation $r_{x,y} \leftarrow r_{x,y} \cap (r_{x,z} \circ r_{z,y})$, in which z is any third variable occurring in the network. In GQR Mackworth's variant of the path consistency algorithm is implemented, e.g. (van Beek and Manchak 1996), which runs in cubic time in the size of the constraint network. Since, in general, the path consistency method is not sufficient to decide consistency of constraint networks, GQR uses chronological backtracking with 2-way or d-way branching, such trying out different instantiations of the constraints containing disjunctions of base relations (cf. Ladkin and Reinefeld 1997; Nebel 1997; van Beek and Manchak 1996). Moreover, by using known tractable subclasses of a calculus (i.e., sets of relations, for which the path consistency method decides satisfiability) one can speed up the reasoning time: instead of splitting a constraint during backtracking into base relations, one can split it into relations belonging to a tractable subclass, which reduces the branching factor of the search tree considerably.

Heuristics. Backtracking search may benefit from heuristics about which part of the constraint network is to be processed next. Currently, the classic weight and cardinality heuristics (van Beek and Manchak 1996) are implemented. Further, a heuristic combining (static) weights with dynamic weights as a boost (Boussemart et al. 2004) is available, which also makes use of eligible constraints (Condotta, Ligozat, and Saade 2007). The selection of sub-relations is based on weights.

Performance. Even though GQR works with arbitrary qualitative calculi, it is competitive with calculus-specific solvers as e.g., Nebel's solver for Allen's interval calculus (Nebel 1997) (see Figure 1).

Extensibility. GQR is written in C++ with an object-oriented design, in which users may add their own heuristics quite easily. New qualitative calculi are defined by writing simple text files, which are then read and processed by the reasoner. Moreover, GQR allows for checking algebraic properties of specified calculi and supports precomputations of composition and converse operations either in part or completely, such that the path consistency algorithm receives a significant speedup (Ladkin and Reinefeld 1997).

GQR is freely usable and distributable under the terms of the GNU General Public License. The software can be downloaded from `https://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/Resources/GQR`.

Acknowledgments

This work was partially supported by Deutsche Forschungsgemeinschaft (DFG) as part of the Transregional Collaborative Research Center *SFB/TR 8 Spatial Cognition*.

References

Boussemart, F.; Hemery, F.; Lecoutre, C.; and Sais, L. 2004. Boosting systematic search by weighting constraints. In *Proc. of the 16th Eureopean Conference on Artificial Intelligence (ECAI'2004)*, 146–150. IOS Press.

Condotta, J.-F.; Ligozat, G.; and Saade, M. 2007. Eligible and frozen constraints for solving temporal qualitative constraint networks. In *Proc. of Principles and Practice of Constraint Programming (CP 2007)*, LNCS 4741, 806–814. Springer.

Condotta, J.-F.; Saade, M.; and Ligozat, G. 2006. A generic toolkit for n-ary qualitative temporal and spatial calculi. In *Thirteenth International Symposium on Temporal Representation and Reasoning - TIME'06*, 78–86. IEEE Computer Society.

Dylla, F.; Frommberger, L.; Wallgrün, J. O.; Wolter, D.; Nebel, B.; and Wölfl, S. 2007. SailAway: Formalizing navigation rules. In *Proc. of the Artificial and Ambient Intelligence Symposium on Spatial Reasoning and Communication (AISB'07)*.

Gantner, Z.; Westphal, M.; and Wölfl, S. 2008. GQR – A fast reasoner for binary qualitative constraint calculi. In *AAAI'08 Workshop on Spatial and Temporal Reasoning*. AAAI Press.

Ladkin, P. B., and Reinefeld, A. 1997. Fast algebraic methods for interval constraint problems. *Annals of Mathematics and Artificial Intelligence* 19(3-4):383–411.

Nebel, B. 1997. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *Constraints* 1(3):175–190.

Renz, J., and Schmid, F. 2007. Customizing qualitative spatial and temporal calculi. In *Australian Conference on Artificial Intelligence*, LNCS 4830, 293–304. Springer.

van Beek, P., and Manchak, D. W. 1996. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research* 4:1–18.

Wallgrün, J. O.; Frommberger, L.; Wolter, D.; Dylla, F.; and Freksa, C. 2006. Qualitative spatial representation and reasoning in the SparQ-toolbox. In *Spatial Cognition V*, LNCS 4387, 39–58. Springer.