Confirming the QSR Promise

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

Abstract

Within the qualitative spatial reasoning community it has been a widely accepted commonplace that reasoning in qualitative constraint calculi outperforms reasoning in other more general and expressive formalisms. To check the correctness of this assumption we conducted some empirical case studies in which we compared the performance of a qualitative constraint solver with different automated reasoning systems, namely first-order and description logic reasoners. We also report on some first results from comparing the performance of qualitative and finite constraint solvers. Our empirical tests are based on randomly generated instances of qualitative constraint satisfaction problems, which have been encoded as reasoning problems for first-order reasoners, description logic reasoners, and finite CSP solvers, respectively. Given our currently used encodings, these studies show that first-order and description logic reasoners are far from being feasible for problem sizes that can easily be solved by a qualitative reasoner. In contrast, finite CSP solvers are competitive, but still outperformed by a qualitative reasoner on the problem instances considered here.

Introduction

One of the main computational arguments for the development of constraint-based formalisms for representing spatial and temporal knowledge is that these formalisms allow for solving reasoning tasks in an efficient manner. In this paper we will underpin this research promise of Qualitative Spatial Reasoning (QSR) by concrete data resulting from performance tests of different automated reasoning systems. More precisely, we will compare the Generic Qualitative Reasoner (GQR) (Gantner, Westphal, and Wölfl 2008) with the first-order reasoners SPASS (Weidenbach et al. 2007) and E (Schulz 2004), then with the description logic reasoner FaCT++ (Tsarkov and Horrocks 2006), and finally with the CSP solver Mistral (Hebrard 2008).

Benchmarking of automated reasoners for first-order (FO) logic has an established tradition. The annual CADE ATP System Competition (CASC) on automated theorem proving builds on the problem library TPTP (Sutcliffe and Suttner 1998), which groups reasoning problems into different mathematical application domains (such as algebra, topology, etc.). Problem instances in the TPTP library may be considered *natural* in the sense that the formulas to be validated express typical reasoning problems that arise from the given theory.[1] Benchmarking of description logic (DL) reasoners are also mostly conducted against natural knowledge bases, namely DL ontologies expressed in different DL languages and with varying parameters (number of concepts, size of the TBox, etc.) (Gardiner, Horrocks, and Tsarkov 2006). Moreover, Tsarkov et al. (2004) report on a performance comparison between FO and DL reasoners. Finally, in the finite CSP domain solver competitions have been conducted since 2005. Comparisons are performed in different problem categories ranging from real-world instances to abstract academic instances on the one hand and to completely randomly generated instances on the other.

In the QSR domain, benchmarking of reasoning systems has not played a similar role so far. Moreover, it has been a widely accepted assumption that reasoning methods tailored to the considered constraint languages outperform reasoning methods that are applicable to suitable encodings of qualitative CSP instances in other formal languages. This assumption, however, has been tackled by several authors. For instance, Brand (2004) pointed out several advantages of encoding qualitative CSP instances as finite CSP instances, and Pham et al. (2006) have shown in an empirical study that SAT solvers applied to SAT encodings of qualitative temporal reasoning instances can perform better than a qualitative reasoner.

In this paper we report on the results of empirical performance tests, which we set up to compare a qualitative constraint solver with different automated reasoning systems. For our performance analyses we consider three test settings. In the first setting we compare the FO reasoners E and SPASS with the qualitative CSP solver GQR. The problem instances generated for this setting are entailment proofs in the theory of strict linear orders, which we thought to be easily solvable by first-order reasoners. In more detail, the problem instances for this test setting are constructed from randomly generated CSP instances in Allen's interval calculus (Allen 1983), which are then translated into FO formulas expressing the constraints between interval start and

Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] In this context it is also worth mentioning that ontology fragments from SUMO and ResearchCyc have been included in the last CADE competitions (Sutcliffe 2007).

endpoints.

In the second setting the performance of the DL reasoner FaCT++ is compared with that of GQR. Here we use satisfiability tests on problem instances in the RCC8 calculus (Randell, Cui, and Cohn 1992), since these can be translated in a natural manner into the description logic *SROIQ* (Horrocks, Kutz, and Sattler 2006): first RCC8 instances are translated into modal logics by using the encoding presented by Nutt (1999). DL reasoners can then be applied to the TBoxes that one obtains from the standard translation of the relevant multi-modal logic into DL (Schild 1991; Baader, Horrocks, and Sattler 2007).

In the third setting the finite CSP solver Mistral is compared with GQR. Again, problem instances are randomly generated instances in Allen's interval calculus, but with a different problem generator than in the first setting. Interval networks are encoded as finite CSP instances by translating them into a variant of the so-called dual constraint problem (Renz and Nebel 2001; Brand 2004; Condotta et al. 2006).

GQR and Constraint-based Qualitative Spatio-Temporal Reasoning

GQR is a solver for binary qualitative constraint networks. Such networks are 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 are satisfied. Since the domains considered in qualitative reasoning are usually infinite, constraint solving techniques need to be applied to finite, symbolic representations of constraint networks. These are directed finite constraint graphs, where each edge is labeled by a set of relation symbols: each symbol (called *base relation*) represents a concrete binary relation on the domain. Sets of base relations are read disjunctively, that is, they can be used to express imprecise knowledge about the actual configuration.

We assume that, given a finite set B of base relations, the set 2^B has the algebraic structure of a non-associative relation algebra. Depending on the considered calculus, this algebra is constructed from functions ${}^\smile : B \to B$ (assigning to each base relation its converse) and $\circ : B \times B \to 2^B$ (assigning to each pair of base relations their composition) if these are extended to functions on 2^B as follows: $r^\smile := \{b^\smile : b \in r\}$ 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 refinement of the constraint graph, in which each edge is a base relation and which is closed under composition (path-consistent). This is sufficient to prove the satisfiability problem for the calculi used in our tests. GQR represents reasoning problems as constraint graphs, and implements the symbolic path consistency algorithm (also known as algebraic closure) and heuristic backtracking search for decompositions of relations, based on tractable fragments (Renz and Nebel 2001) of qualitative constraint calculi.

Test 1: Comparison with FO Reasoners

In our first test setting we compared the performance of GQR with that of the first-order reasoners E and SPASS:

The *Equational Theorem Prover (E)* is a purely equational theorem prover for first-order logic. The inference system used by E is based on the superposition calculus for equational clausal logic (Schulz 2002).[2] For our benchmarks we used the latest version of E, E 0.999, which took part at CASC-21 and performed reasonably well in many test categories.[3]

SPASS, developed at MPI Saarbrücken, is a saturation-based automated theorem prover for first-order logic with equality. SPASS is based on resolution and integrates various other proof procedures (e.g., paramodulation). In the benchmarks the most recent available version of SPASS, SPASS-3.0, was used.[4]

Each reasoner was used with its standard settings, i.e., without problem-tailored optimization options.

For our first setting we decided to compare the performance of GQR on problem instances that can be encoded as a reasoning problem in a simple FO theory. Hence we considered problem instances in Allen's interval calculus: Problem instances can here be reformulated using the theory of strict linear orders, T, as background theory, i.e., the signature of T has a single binary relation symbol and the theory consists of three axioms (irreflexivity, transitivity, and linearity).[5]

Furthermore, instead of satisfiability checks we considered entailment problems. In constraint-based reasoning the *(minimal) entailment problem* can be defined as follows: Given a conjunctive formula

$$\varphi := \bigwedge_{i<j\leq n} x_i \, r_{ij} \, x_j, \quad i,j,n \in \mathbb{N}, r_{ij} \in 2^B$$

find the (minimal) relation r such that $\varphi \models x_1 \, r \, x_2$.

We adapted and simplified this entailment task for FO reasoners by determining r beforehand. Then the FO reasoners were asked to prove

$$T \cup \{\Phi(\varphi)\} \models \Phi(x_1 \, r \, x_2),$$

where Φ is the standard translation, which assigns to each Allen constraint $x_i \, r \, x_j$ a FO formula expressing the relations between the start and endpoints of the intervals i and j. For example, the problem

$$x_1 \,\text{Before}\, x_3, \;\; x_2 \,\text{During}\, x_3 \models x_1 \,\text{Before}\, x_2$$

[2] The sources of E can be downloaded from `http://www.eprover.org`.

[3] Vampire (Riazanov and Voronkov 2002), which is considered one of the best FO reasoners, is not publicly available; the usage of the source code used in the CASC competition is forbidden by the Vampire developers.

[4] Precompiled binaries as well as the sources of SPASS can be downloaded from `http://www.spass-prover.org/`.

[5] Usually, one also assumes density and no first/last elements. But these existential axioms decreased the reasoning performance of the first-order reasoners drastically, so that they had to be removed.

is translated into the problem:

$$T \cup \{x_1^s < x_1^e,\ x_2^s < x_2^e,\ x_3^s < x_3^e,\ x_1^e < x_3^s,\ x_3^s < x_2^s,\ x_2^e < x_3^e\} \models x_1^e < x_2^s.$$

For the test instances we restricted consideration to tasks, where the premise, the formula φ (and hence $\Phi(\varphi)$), was satisfiable and further the entailment relation was valid. This was done since tasks with a valid premise are the interesting cases and the FO reasoners had problems disproving an entailment relation.

The generation of φ was performed by the following procedure dependent on parameters n, k, and l.[6]

1. Generate a random assignment of n interval variables, where each variable x_i ($i \in [1..n]$) is taking values in $\{(t,t') \in \mathbb{R}^2 : t < t'\}$ (assuming an equal distribution).
2. Translate this "scenario" into a conjunctive formula φ^* (all relations r_{ij} are base relations).
3. Construct a "noisy variant" φ out of φ^* as follows:
   - r_{ij} in φ is left unchanged (wrt. φ^*) with probability k, otherwise it is removed (*density*);
   - for each relation r_{ij} in the new φ, we replace r_{ij} by $r_{ij} \cup r'$ with probability l, where r' is randomly picked from an equal distribution of non-empty relations from 2^B (*noise*);
   - set $r = r_{1,2}$ as the universal relation.

In the empirical tests that followed GQR had to solve the entailment problem, i.e., it computed the entailed relation r. For the FO reasoners, we incorporated the, now known, entailed relation r directly into the problem, i.e., they had to verify: $T \cup \{\Phi(\varphi)\} \models \Phi(x_1\, r\, x_2)$.

Empirical Results

With formulas φ generated as described above, we ran GQR, E and SPASS on an Intel Xeon with 3 GHz, 3 GiB RAM and a time limit of 5 minutes for each problem instance. For E the translated problem was written in TPTP format and for SPASS the generated TPTP file was translated into DFG format by using the translator from the TPTP problem library (Sutcliffe and Suttner 1998).

The resulting formulas φ used for the benchmarks can be considered easy instances for constraint-based reasoning, since they are not close to the phase transition and do not use any restricted set of "hard relations" (e.g., 3-CNF). In particular, the actually used number of variables was too small to find hard instances for GQR (cf., e.g., (Renz and Nebel 2001)), i.e., GQR solved similarly sized instances within a split second.

The first test was done using the parameters $k = 0.9$ (density) and $l = 0.3$ (noise). We generated 100 problem instances for 15 and 20 nodes, respectively. As the plots in Figure 1 show, both SPASS and E did not scale well with the number of variables. It is interesting that E always solved around 20% of instances regardless of their size. All of the instances solved by E within 2 seconds entailed the relation Before or its converse. Furthermore, the solved instances which required more than 20 seconds had as entailed relation During or its converse. In particular, E did not manage to solve any instance that entailed a non-atomic relation. In contrast, SPASS solved a wide variety of instances, but failed to solve a reasonable amount of instances with 20 variables. Overall, clearly neither of the two FO reasoners came close to the performance of a constraint-based system.

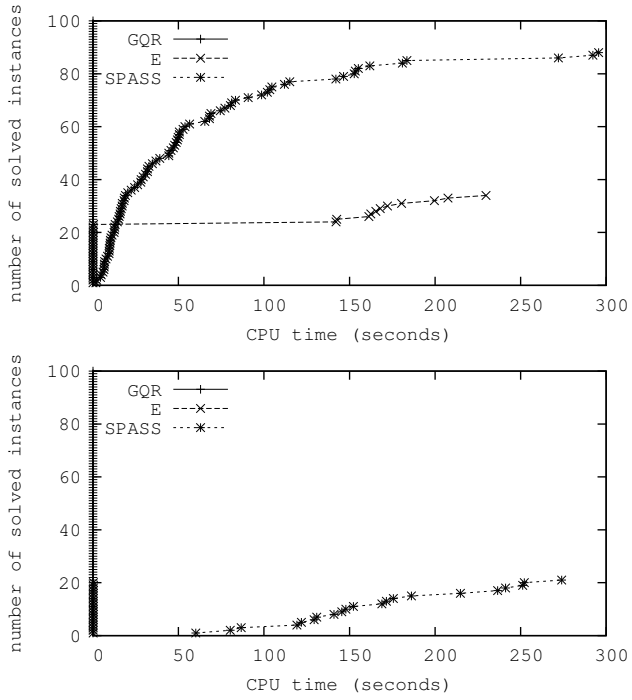

Figure 1: Performance on instances of the entailment problem in Allen's interval calculus. From top to bottom: $n = 15, 20$ (nodes), $k = 0.9$ (density), and $l = 0.3$ (noise).

As a second test we reduced the noise in the instances by lowering the noise parameter l to 0.1. We generated 100 problem instances each for 10, 20 and 30 nodes. The results for $n = 30$ can be seen in Figure 2. Again, E showed exactly the same behaviour. It solved more instances, but due to the reduced noise less non-atomic relations were entailed in general. The performance of SPASS improved with the reduced noise but SPASS still required too much time to conduct any reasonable comparison with GQR.

Test 2: Comparison with a DL Reasoner

In the second setting we compared the performance of the DL reasoner FaCT++ with GQR. FaCT++ is a tableaux-based reasoner and supports the description logic *SROIQ*

[6] By this method we only generate satisfiable formulas based on an equal distribution of *models*. This differs from the established method of choosing instances based on an equal distribution of relations. If one uses the latter method, one has to filter the satisfiable instances, which also results in a shift of the distribution of relations.

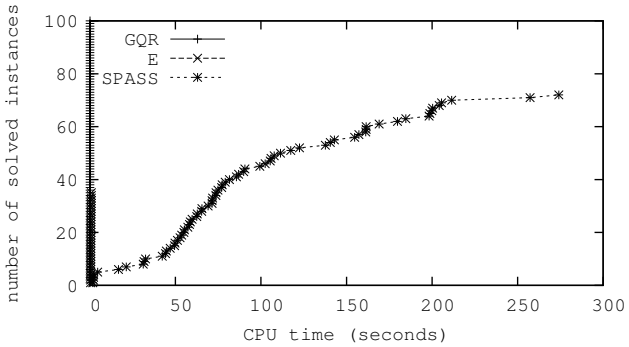

Figure 2: Performance on instances of the entailment problem in Allen's interval calculus with $n = 30$ (nodes), $k = 0.9$ (density), and $l = 0.1$ (noise).

which is the formal basis of the OWL 1.1 standard.[7]

As a basis for our performance tests we used randomly generated instances in the RCC8 calculus. While GQR had to check whether the given problem instance is consistent, the FaCT++ reasoner had to check whether the concept defined by translating the problem instance into DL (Nutt 1999; Schild 1991; Baader, Horrocks, and Sattler 2007) is satisfiable. Under this translation, for example the RCC8 network

$$x_1 \,\mathrm{PartiallyOverlaps}\, x_2,\ x_2 \,\mathrm{TangentialProperPart}\, x_3$$

is translated into the problem of checking whether the following concept is satisfiable with respect to a TBox with two roles u and int, where the latter is assumed to be reflexive and transitive:

$$\begin{aligned}
&\textstyle\bigsqcap_{i=1..3} \exists \mathrm{u}.\exists \mathrm{int}.C_i \sqcap \\
&\textstyle\bigsqcap_{i=1..3} \forall \mathrm{u}.\forall \mathrm{int}.((\neg C_i \sqcup \exists \mathrm{int}.\forall \mathrm{int}.C_i) \sqcap (C_i \sqcup \forall \mathrm{int}.\exists \mathrm{int}.\neg C_i)) \sqcap \\
&\big(\exists \mathrm{u}.\exists \mathrm{int}.(\forall \mathrm{int}.C_1 \sqcap \forall \mathrm{int}.C_2) \sqcap \\
&\quad \exists \mathrm{u}.\exists \mathrm{int}.(C_1 \sqcap \neg C_2) \sqcap \exists \mathrm{u}.\exists \mathrm{int}.(\neg C_1 \sqcap C_2)\big) \sqcap \\
&\big(\forall \mathrm{u}.\forall \mathrm{int}.(\neg C_2 \sqcup C_3) \sqcap \exists \mathrm{u}.\exists \mathrm{int}.(C_3 \sqcap \neg C_2) \sqcap \\
&\quad \exists \mathrm{u}.\exists \mathrm{int}.(C_2 \sqcap \neg \forall \mathrm{int}.C_3)\big)
\end{aligned}$$

Here the concept in the 2nd line expresses the regularity condition for RCC8 regions, i.e., that all regions are regular closed subsets of a topological space.[8] Since FaCT++ performed rather poorly on this encoding, we also considered an approximative encoding, which simply leaves out the "regularity condition". In the following plots this approximative variant is referred to as "FaCT++-approx".

Given parameters n, k, s (as explained below), we generated problem instances φ as follows:

- for each possible relation $r_{ij}, 1 \leq i < j \leq n$, set r_{ij} to a non-empty relation with probability k, otherwise omit the relation, i.e., we set it to the universal relation (*density*);
- this non-empty relation is given by a binomial distribution in which a base relation is included with probability s.

Again, the generated problem instances are considered to be easy for constraint-based reasoning. Especially the number of variables that was used in the following tests was too small to find hard instances for constraint-based reasoning.

Empirical Results

We used similarly sized problem instances as in the first setting for our comparison of GQR with FaCT++. Both GQR and FaCT++ were run on an Intel Xeon with 3 GHz, 3 GiB RAM and a time limit of 5 minutes for each problem instance.

In our test series we considered problem instances with 10, 15, and 20 variables, and a fixed density parameter $k = 0.2$. For the binomial distribution we considered $s = 0.1$ and $s = 0.3$. For $s = 0.1$, FaCT++ was unable to solve all such instances. However, the approximative encoding increased performance drastically such that FaCT++ always terminated within the time limit. For the 100 test instances, the "FaCT++-approx"-results were only wrong in 8 cases (in each case an inconsistent description could not be detected due to the lack of some necessary regularity axiom).

The results for $s = 0.3$ are shown in Figure 3. Even the approximative encoding was unable to handle 20 variables and did not detect inconsistencies in 13 out of the 300 test instances in total. These results show that, given the used problem encoding, FaCT++ scales poorly with regard to both the number of disjunctions and the number of variables. Furthermore, FaCT++ seems unfeasible for problem sizes easily solved by constraint-based reasoning.

Test 3: Comparison with a Finite CSP Solver

To compare GQR with a finite CSP solver, a CSP solver capable of handling ternary constraints (cf. below) is needed. The CSP solver competition 2008 provides an overview of current state-of-the-art systems. We chose *Mistral* (Hebrard 2008), the highest ranking available solver, which was not portfolio-based. Mistral is a library for modeling and solving constraint problems, which, bundled with a front-end for reading and modeling problems, ranked fifth in the 2008 CSP competition. The `mistral-prime` version of Mistral for the CSP solver competition 2008 was used for our tests.

Qualitative constraint networks can be encoded as finite CSP instances (Renz and Nebel 2001; Brand 2004; Condotta et al. 2006): Given the constraint graph of a qualitative constraint problem with variables $X = \{x_1, \ldots, x_n\}$, we obtain a finite constraint satisfaction problem by considering a variant of the so-called *dual constraint problem*: Let V be a set of variables containing a variable v_{ij} for each pair of variables $x_i, x_j \in X$ with $i < j$. Then the dual problem has the form

$$\big\langle V, B, \{C_{ij} : 1 \leq i < j \leq n\} \cup \{C_{ijk} : 1 \leq i < j < k \leq n\} \big\rangle$$

[7] The project webpage is available at `http://owl.man.ac.uk/factplusplus/`.

[8] We used a slightly modified variant of Nutt's translation of RCC8 set constraints into modal logic: In (Nutt 1999) the encodings for the relations TPP and NTPP (and their converses) are too weak since these encodings do not enforce that the relata of those relations are distinct.

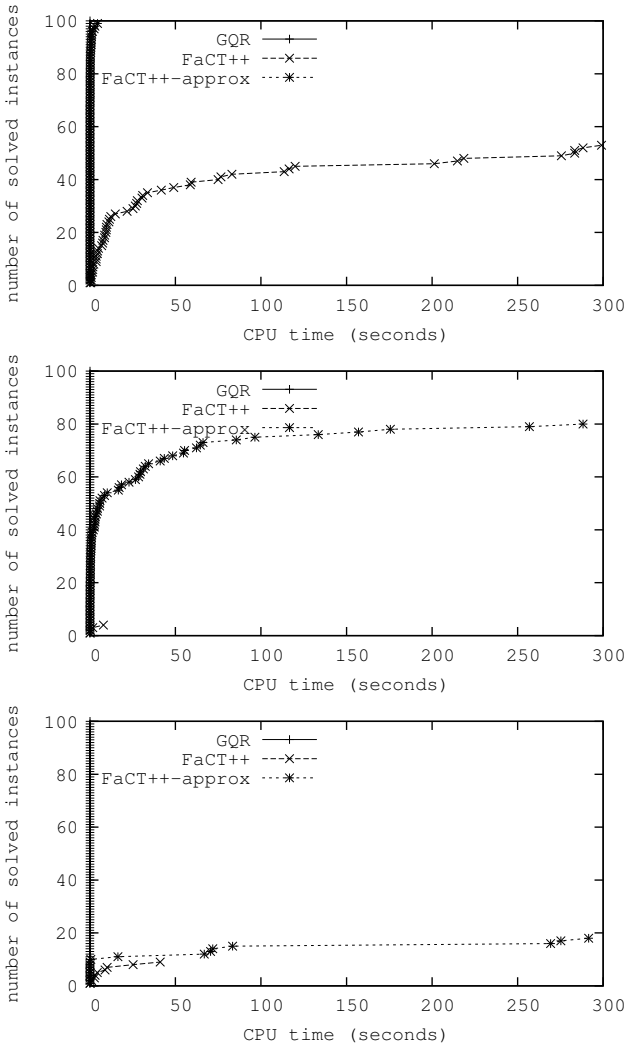

Figure 3: Reasoning in RCC8. From top to bottom: $n = 10, 15, 20$ (nodes), $k = 0.2$ (density) and $s = 0.3$ (label distribution).

where C_{ij} is a domain constraint restricting the values of v_{ij} to those base relations that are contained in the label on the arc from x_i to x_j. C_{ijk} is a binary constraint that enforces variables v_{ij}, v_{jk} ("qualitative edges") to assignments, in which x_j has the same value. Since this scheme does not work on infinite domains, these binary constraints are replaced by a set of ternary constraints $TC = \{((v_{ij}, v_{ik}, v_{kj}), R_\circ) : 1 \leq i < j < k \leq n\}$, where $R_\circ := \{(c, a, b) \in B^3 : c \in a \circ b\}$. The resulting finite network features $\frac{n \cdot (n-1)}{2}$ variables and $\binom{n}{3} = \frac{(n-1)^3 - (n-1)}{6}$ TC constraints.

A solution of this finite CSP instance corresponds to an atomic, consistent refinement of the qualitative constraint network, and vice versa (Condotta et al. 2006).

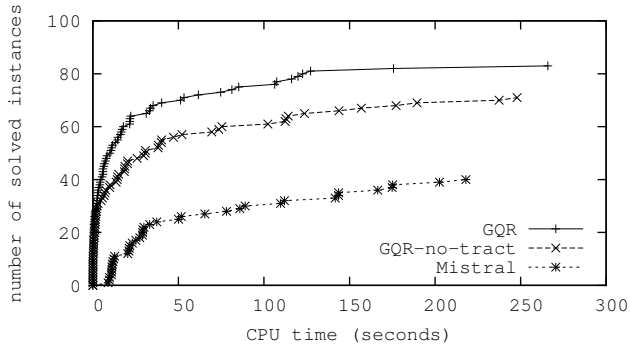

Figure 4: Reasoning about Allen intervals. Instances with 100 nodes, a density of $k = 0.106$ and labels with parameter $s = 6.5$.

Empirical Results

To generate test instances within the phase transition, we used the same setup as described for Test 2 with the following settings: number of nodes $n = 100$, density $k = 0.106$, and for the binomial distribution $s = 0.5$. Again all tests were performed on an Intel Xeon with 3 GHz, 3 GiB RAM and a time limit of 5 minutes for each problem instance.

Our results, depicted in Figure 4, show Mistral to be slower than GQR. In particular, it solved less instances within the considered five minute time frame. "GQR-no-tract" in the plot refers to a variant of GQR, which did not exploit tractability information. Even in this case, GQR outperformed Mistral. Moreover, as can be seen from the figure, Mistral performed always slower than GQR. In comparison to Test 1 and Test 2, however, Mistral's performance was much closer to the performance of GQR, in particular, with regard to the size of instances. Note that in the test GQR failed to solve some of the instances, but that all instances solved by Mistral were also solved by GQR.

Conclusion

In this paper we have presented the results of an empirical test series in which we compared the performance of a solver for binary constraint calculi to state-of-the-art reasoning systems for first-order logic, description logics, and finite CSPs. For this we considered standard encodings of qualitative constraint networks as known from the literature. On the one hand, these are *semantic encodings*, in which the qualitative information is restated in the context of an underlying theory, i.e., we represented Allen relations as relations between start and endpoints of intervals within the theory of strict linear orders and RCC8 relations by using the interior and closure function of a topological space. On the other hand, we used a *syntactic encoding*, namely the finite CSP encoding, which provides a syntactic reformulation of the qualitative problem instance (such that the actual search problem is essentially the same).

Our empirical analysis shows that, given the used translations into first-order and description logics respectively, the performance of both FO and DL reasoners is far behind the performance of tailored methods used in constraint-based

reasoning. Due to our results, it seems necessary to try alternative translations into first-order or description logics in order to allow for a more useful comparison. Enforcing specific reasoning procedures (via reasoner options) might also provide a speedup for both FO and DL reasoners. If it is possible to increase the performance of FO and DL reasoners, one could increase the number of variables and compare the performance on "hard" instances, e.g., instances with restricted relations (e.g., 3-CNF) taken from the phase transition. For such instances the comparison is certainly more interesting than for the instances considered here.

In our third test setting, in which we considered a syntactic encoding of qualitative constraint networks as finite CSPs, the comparison of reasoners is more interesting. Our results show that a typical CSP solver is not as fast as a qualitative reasoner on instances from Allen's interval calculus. Its performance, however, is much more competitive than FO and DL reasoners, which is not surprising, since the considered encoding is much closer to qualitative constraint networks. Finally, it should be noted that we only used Allen's interval calculus in our comparison, which is a rather small calculus, compared to more recently discussed calculi such as RCC23 and calculi from the OPRA family. A thorough performance comparison with finite CSP solvers on large calculi would be particularly interesting in view of constraint propagation and heuristics (cf., e.g., (Bessière 1996)). In particular, this would allow for a detailed search space comparison in terms of visited nodes and pruning power.

Acknowledgements

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

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Baader, F.; Horrocks, I.; and Sattler, U. 2007. Description Logics. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier.

Bessière, C. 1996. A simple way to improve path consistency processing in interval algebra networks. In *AAAI/IAAI, Vol. 1*, 375–380.

Brand, S. 2004. Relation variables in qualitative spatial reasoning. In *Proc. of KI 2004: Advances in Artificial Intelligence*, LNCS 3238, 337–350. Springer.

Condotta, J.-F.; D'Almeida, D.; Lecoutre, C.; and Sais, L. 2006. From qualitative to discrete constraint networks. In *Proc. of the KI'2006 Workshop on Qualitative Constraint Calculi*, 54–64.

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

Gardiner, T.; Horrocks, I.; and Tsarkov, D. 2006. Automated benchmarking of description logic reasoners. In Parsia, B.; Sattler, U.; and Toman, D., eds., *Description Logics*, volume 189 of *CEUR Workshop Proceedings*.

Hebrard, E. 2008. Mistral, a constraint satisfaction library. In *Proc. of the Third International CSP Solver Competition*.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible SROIQ. In *Proc. of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, 57–67. AAAI Press.

Nutt, W. 1999. On the translation of qualitative spatial reasoning problems into modal logics. In *Proc. of KI-99: Advances in Artificial Intelligence, 23rd Annual German Conference on Artificial Intelligence*, LNCS 1701, 113–124. Springer.

Pham, D. N.; Thornton, J.; and Sattar, A. 2006. Towards an efficient SAT encoding for temporal reasoning. In *Proc. of Principles and Practice of Constraint Programming (CP 2006)*, LNCS 4204, 421–436. Springer.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. In *KR'92, Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference*, 165–176. Morgan Kaufmann.

Renz, J., and Nebel, B. 2001. Efficient methods for qualitative spatial reasoning. *J. Artif. Intell. Res. (JAIR)* 15:289–318.

Riazanov, A., and Voronkov, A. 2002. The design and implementation of Vampire. *AI Communications* 15(2-3):91–110.

Schild, K. 1991. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th International Joint Conference on Artificial Intelligence*, 466–471.

Schulz, S. 2002. E - A brainiac theorem prover. *AI Communications* 15(2-3):111–126.

Schulz, S. 2004. System Description: E 0.81. In *Proc. of the 2nd International Joint Conference on Automated Reasoning*, LNAI 3097, 223–228. Springer.

Sutcliffe, G., and Suttner, C. 1998. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning* 21(2):177–203.

Sutcliffe, G. 2007. TPTP, TSTP, CASC, etc. In *Proc. of the Computer Science Symposium in Russia*, LNCS 4649, 6–22. Springer.

Tsarkov, D., and Horrocks, I. 2006. FaCT++ description logic reasoner: System description. In *Proc. of the International Joint Conference on Automated Reasoning*, LNCS 4130, 292–297. Springer.

Tsarkov, D.; Riazanov, A.; Bechhofer, S.; and Horrocks, I. 2004. Using Vampire to reason with OWL. In *Proc. of the International Semantic Web Conference*, LNCS 3298, 471–485. Springer.

Weidenbach, C.; Schmidt, R. A.; Hillenbrand, T.; Rusev, R.; and Topic, D. 2007. System description: SPASS version 3.0. In *Proc. of Automated Deduction - CADE-21, 21st International Conference on Automated Deduction*, LNCS 4603, 514–520. Springer.