Qualitative Constraint Calculi: Heterogeneous Verification of Composition Tables

Stefan Wölfl[1], Till Mossakowski[2], and Lutz Schröder[2]

[1] Department of Computer Science, University of Freiburg,
Georges-Köhler-Allee, 79110 Freiburg, Germany
woelfl@informatik.uni-freiburg.de

[2] DFKI Lab Bremen and Department of Computer Science, University of Bremen,
P.O. Box 330440, 28334 Bremen, Germany
till, lschrode @informatik.uni-bremen.de

Abstract. In the domain of qualitative constraint reasoning, a subfield of AI which has evolved in the past 25 years, a large number of calculi for efficient reasoning about spatial and temporal entities has been developed. Reasoning techniques developed for these constraint calculi typically rely on so-called composition tables of the calculus at hand, which allow for replacing semantic reasoning by symbolic operations. Often these composition tables are developed in a quite informal, pictorial manner and hence composition tables are prone to errors. In view of possible safety critical applications of qualitative calculi, however, it is desirable to formally verify these composition tables. In general, the verification of composition tables is a tedious task, in particular in cases where the semantics of the calculus depends on higher-order constructs such as sets. In this paper we address this problem by presenting a heterogeneous proof method that allows for combining a higher-order proof assistance system (such as Isabelle) with an automatic (first order) reasoner (such as SPASS or VAMPIRE). The benefit of this method is that the number of proof obligations that is to be proven interactively with a semi-automatic reasoner can be minimized to an acceptable level.

Keywords. Knowledge representation and reasoning, geometric and spatial reasoning, qualitative reasoning, automated versus interaction proving, heterogeneous specification

1 Introduction

Qualitative reasoning aims at describing the common-sense background knowledge on which our human perspective on the physical reality is based. Methodologically, qualitative constraint calculi restrict the vocabulary of rich mathematical theories dealing with temporal or spatial entities such that specific aspects of these theories can be treated within decidable fragments with simple qualitative (i. e., non-metrical) languages. Contrary to mathematical or physical theories about space and time, qualitative constraint calculi allow for rather inexpensive reasoning about entities located in space and time. For this reason, the limited expressiveness of qualitative representation formalisms is a benefit if applications require online processing of spatial or temporal information. To mention just two possible application fields, some qualitative calculi may be implemented for handling spatial GIS queries efficiently and some may be used for enabling human-machine interaction, for example, with a mobile robot.

In the past 25 years the number of qualitative calculi dealing with spatial and temporal entities has grown quite steadily. The calculi discussed in the literature employ concepts from a wide range of mathematical theories. Some of them are based on geometrical notions such as lines, half-planes, and angles, some describe relations between physical objects in terms of point set topology, and some include qualitative size information. Here, we are specifically interested in calculi that are interpreted over higher-order entities such as *sets* of points or on entities that can be characterized only via *second-order* properties. The most prominent calculus of this kind are the various region connection calculi [1,2,3,4] as well as the 4- and the 9-intersection calculus [5,6]. Further examples include the cardinal direction calculus for spatially extended objects in the Euclidean plane [7], or calculi that crucially rely on the second-order aspects of the real numbers (conceived of as, e. g., a complete linear order).

Reasoning problems in qualitative calculi are usually formulated as so-called constraint satisfaction problems. Starting from a set of base relations (i. e., a family of relations that partitions the set of all tuples of domain elements), a constraint is a formula of the form xRy with variables x and y (taking values in given domains D_x and D_y) and a set of base relations R defined between the domains of x and y. Constraints may also contain sets of base relations between two variables — sets of base relations (referred to as *relations*) are read disjunctively and hence express imprecise knowledge about the concrete scenario described by the constraint formula. The constraint satisfaction problem with respect to a fixed qualitative calculus is to determine for a given constraint network (i. e., a finite set of constraints) whether there exists an assignment to its variables such that all constraints of the network become true. Further typical reasoning tasks are to check that some constraint is entailed by a constraint network, and to compute an equivalent minimal constraint network (all these reasoning tasks are equivalent under polynomial Turing reductions).

As an example, let us consider the region connection calculus RCC-8. In this calculus it is possible to express relations between *regions*, which often are represented as non-void, connected, and regular closed (or regular open) subsets of some topological space. The set of RCC-8 base relations consists of the relations DC ("DisConnected"), EC ("Externally Connected"), PO ("Partially Overlap"), TPP ("Tangential Proper Part"), NTPP ("Non-Tangential Proper Part"), the converses of the latter two relations (TPP*i* and NTPP*i*, resp.) and EQ ("EQuals") (cf. Fig. 1 for a pictorial representation). To put it more formally, if we interpret these relations on the non-empty regular closed subsets of a topological space S, the relation NTPP, for example, is the set of all pairs of such closed subsets X and Y such that there exists an open set U with $X \subseteq U \subseteq Y$.

A crucial aspect for developing efficient algorithms for qualitative spatial and temporal calculi is the fact that the underlying model classes usually contain infinite models. Hence, in order to test satisfiability of constraint networks in an infinite model, it is not feasible to enumerate all possible assignments to variables in that model until one finds one that satisfies the constraint network. For this reason other techniques must be applied for testing satisfiability. Most prominently, the path consistency algorithm manipulates a given constraint network C by successively refining the relations $R_{x,y}$ that can hold between any two variables x and y occurring in the network via the following

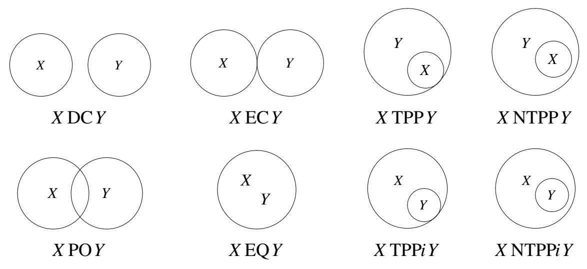


Fig. 1: The RCC-8 relations

operation:

$$R_{x,y} \longleftarrow R_{x,y} \cap (R_{x,z} \circ R_{z,y})$$

where z is any third variable occurring in C and $\circ$ is the composition function defined by a composition table (see Table 1 for the composition table of RCC-8). This composition-based method is at the heart of many theoretical investigations regarding qualitative constraint calculi, since the method often allows for replacing semantic reasoning by syntactic symbol manipulations. On the other hand, this method crucially depends on *semantically correct* composition tables. Quite often, however, composition tables are developed just in an *ut-figura-docet* manner, that is, composition tables are "proved" by referring to pictorial representations of possible configurations. For this reason these tables are prone to errors. In this context it is worth recalling the following fact [8]: in order to generate the composition table of a binary constraint calculus, one needs to check

$$\frac{1}{6}(n^3 + 3n^2 + 2n) - na$$

(possible or impossible) configurations of relations between three objects, where n is the number of base relations and a is the number of non-symmetrical relations.[3]

In view of possible safety critical applications it is at least desirable to formally verify these hand-crafted composition tables, which in general is a tedious task, because the number of composition table entries grows quadratically in the number of base relations of the calculus at hand. If the semantics of the calculus can be axiomatized in a first-order theory, the verification of the composition table can be done by using an automatic first-order reasoner (e. g., SPASS [9] or VAMPIRE [10]). For calculi that rely on a higher-order semantic concepts such as sets (as in the case of RCC-8), the verification of composition tables via higher-order proof assistance systems such as Isabelle [11] seems unreasonable, because frequent user interaction is needed which becomes crucial in particular if the the set of base relations is large.

One strategy to automatically prove the composition table entries of a qualitative calculus is to find a satisfiability equivalent encoding of constraint formulae in a suitable modal logic and then to use a modal logic reasoner or a description logic reasoner (via the standard translation between multi-modal and description logics). The drawback of

[3] In practice, the number of such checks will often be less due to inherent (e. g., geometrical) symmetries of the relations under consideration.

Table 1: The composition table of RCC-8

$\circ$	DC	EC	PO	TPP	NTPP	TPP*i*	NTPP*i*
DC	1	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC	DC
EC	DC, EC, PO, TPP*i*, NTPP*i*	DC, EC, PO, TPP, TPP*i*, EQ	DC, EC, PO, TPP, NTPP	EC, PO, TPP, NTPP	PO, TPP, NTPP	DC, EC	DC
PO	DC, EC, PO, TPP*i*, NTPP*i*	DC, EC, PO, TPP*i*, NTPP*i*	1	PO, TPP, NTPP	PO, TPP, NTPP	DC, EC, PO, TPP*i*, NTPP*i*	DC, EC, PO, TPP*i*, NTPP*i*
TPP	DC	DC, EC	DC, EC, PO, TPP, NTPP	TPP, NTPP	NTPP	DC, EC, PO, TPP, TPP*i*, EQ	DC, EC, PO, TPP*i*, NTPP*i*
NTPP	DC	DC	DC, EC, PO, TPP, NTPP	NTPP	NTPP	DC, EC, PO, TPP, NTPP	1
TPP*i*	DC, EC, PO, TPP*i*, NTPP*i*	EC, PO, TPP*i*, NTPP*i*	PO, TPP*i*, NTPP*i*	PO, TPP, TPP*i*, EQ	PO, TPP, NTPP	TPP*i*, NTPP*i*	NTPP*i*
NTPP*i*	DC, EC, PO, TPP*i*, NTPP*i*	PO, TPP*i*, NTPP*i*	PO, TPP*i*, NTPP*i*	PO, TPP*i*, NTPP*i*	PO, TPP, TPP*i*, NTPP, NTPP*i*, EQ	NTPP*i*	NTPP*i*

this method is that such encodings may be hard to find[4] and that it is not clear how these encodings behave to readings of the composition table which are stronger than the mere consistency-based reading [12].

In this paper we present a heterogeneous proof method for proving the correctness of composition tables, which essentially consists of two steps. In a first step we axiomatize the domain of the higher-order entities occurring as relata of the calculus relations in a first-order theory and use a higher-order proof assistance system to verify that this first-order theory is in fact entailed by the the higher-order theory. In a second step we verify that all the entries of the composition table are correct with respect to the first-order theory. From this we can conclude that the composition table is correct with respect to the higher-order theory as well. Our running example will be the calculus RCC-8, but the general method, of course, is not restricted to that calculus at all. The benefit of this method is that the number of proof obligations to be proven interactively with a semi-automatic reasoner (such as Isabelle) can be minimized to an acceptable level, while the possibly large number of composition table entries can be verified by using an automatic first-order reasoner.

The paper is organized as follows: In section 2 we present the formal underpinnings of our proof method in more detail. Then in section 3 we briefly describe the tools that we used to apply this method. In section 4 the verification of composition tables is explained with an example in more detail. Section 5 provides a summary and a short outlook.

[4] In the case of RCC-8, for example, such an encoding is quite natural. But this stems from the fact that the the modal logic S4 and topological spaces are closely related, since the necessity operator can be read as an interior function.

2 Qualitative Constraint Calculi, Composition Tables, and Theory Morphisms

Let us start by briefly sketching qualitative constraint calculi in a more formal manner. We will use a purely syntactic definition of a qualitative calculus (cp. this to the definition by [13]).

Definition 1. A *qualitative (binary) calculus* is a triple $\mathscr{C} = \langle B, ^{\smile}, \circ, \mathrm{id} \rangle$ consisting of a non-empty finite set B (elements of B are referred to as *base relations*), a unary function $^{\smile}\colon B \to B$ (*converse*), a binary function $\circ\colon B \times B \to 2^B$ (*composition*) and a distinguished element $\mathrm{id} \in B$ (the *identity relation*) such that for all $a, b, c \in B$,

(a) $(a^{\smile})^{\smile} = a$
(b) $\mathrm{id} \circ a = a \circ \mathrm{id} = a$
(c) $(a \circ b)^{\smile} = b^{\smile} \circ a^{\smile}$
(d) $a^{\smile} \in b \circ c \iff c^{\smile} \in a \circ b$

Given a qualitative calculus in this sense, the set 2^B is a Boolean algebra (its elements are referred to as *relations*). Moreover, a non-associative relation algebra is defined on 2^B if the functions $^{\smile}$ and $\circ$ are extended to functions $^{\smile}\colon 2^B \to 2^B$ and $\circ\colon 2^B \times 2^B \to 2^B$, respectively, as follows:

$$r^{\smile} := \{b^{\smile} : b \in r\} \quad \text{and} \quad r \circ r' := \bigcup_{b \in r, b' \in r'} b \circ b'.$$

To explain model classes of qualitative calculi we will first introduce the concepts of *signature* and *model*.[5]

Definition 2. A *(many-sorted) signature* is a tuple $\Sigma = \langle S, F, R \rangle$ such that:

(a) S is a (finite) set of sorts.
(b) For each $(w, s) \in S^* \times S$, $T_{w,s}$ are disjoint sets of (total) function symbols (tuples $w \in S^*$ are referred to as *sort profiles*).
(c) For each $w \in S^*$, R_w is a set of relation symbols.

As usual, individual symbols can be introduced as 0-ary total function symbols. Accordingly, models of such signatures are many-sorted first-order structures: Given a signature Σ, a *Σ-model* is a structure consisting of non-empty carrier sets s^M (for each sort $s \in S$), total functions $f^M\colon w^M \to s^M$ (for each function symbol $f \in F_{w,s}$), and relations $r^M \subseteq w^M$ (for each relation symbol $r \in R_w$).

Definition 3. Let $\Sigma = \langle S, F, R \rangle$ and $\Sigma' = \langle S', F', R' \rangle$ be signatures. A *signature morphism* $\Sigma' \to \Sigma$ is a triple $\sigma = \langle \sigma^{\mathrm{s}}, \sigma^{\mathrm{f}}, \sigma^{\mathrm{r}} \rangle$ consisting of maps (families of maps, resp.):

(a) $\sigma^{\mathrm{s}}\colon S' \to S$,
(b) $\sigma^{\mathrm{f}}_{w,s}\colon F'_{w,s} \to F_{\sigma^{\mathrm{s}}(w), \sigma^{\mathrm{s}}(s)}$, and

[5] We will here and in the following use simplified concepts that underly the algebraic specification language CASL (which will be explained in more detail in the next section) as we used this language and its extensions for specifying constraint calculi and their semantics. For example, S^* denotes the set of all finite (possibly empty) sequences of elements in S.

(c) $\sigma^{\mathrm{r}}_w \colon R'_w \to R_{\sigma^{\mathrm{s}}(w)}$.

On the semantic level, signature morphisms inherit models from the target to the source signature. To see this, let $\sigma\colon \Sigma' \to \Sigma$ be a signature morphism, and let M be a Σ-model. Then σ defines a Σ'-model $M|_\sigma$ (referred to as the σ-*reduct* of M) by

$$s^{M|_\sigma} := \sigma^{\mathrm{s}}(s)^M, \quad f^{M|_\sigma} := \sigma^{\mathrm{f}}(f)^M, \quad \text{and} \quad r^{M|_\sigma} := \sigma^{\mathrm{r}}(r)^M.$$

and it holds:

$$M|_\sigma \models \phi \iff M \models \sigma(\phi), \tag{1}$$

where ϕ is a closed Σ'-formula, and $\sigma(\phi)$ is the translation of ϕ into Σ along σ.

In what follows, let $\mathscr{C} = \langle B, {}^{\smile}, \circ, \mathrm{id}\rangle$ be a binary constraint calculus. Let Σ be a (possibly many-sorted) signature containing a distinguished sort s_B and a binary relation symbol with sort-profile (s_B, s_B) for each $b \in B$ (for the sake of simplicity we will use b to denote this symbol as well). Finally, let T be a first-order (or higher-order) Σ-theory such that

(a) $T \models \forall x,y : s_B \left(x\, b\, y \leftrightarrow y\, b^{\smile}\, x\right)$;
(b) $T \models \forall x,y : s_B \left(x \,\mathrm{id}\, y \leftrightarrow x = y\right)$;
(c) $T \models \forall x,y : s_B \left(x\, b\, y \to \neg x\, b'\, y\right)$, if $b \neq b'$;
(d) $T \models \forall x,y : s_B \bigvee_{b\in B} x\, b\, y$.

The first two axioms express that the identity symbol and converse function are interpreted in the natural way. The third and the fourth axiom express that each model of T defines a system of pairwise disjoint and jointly exhaustive relations on the domain of the model.

Definition 4. $\mathscr{C}$ is *weakly correct* for T if

$$T \models \forall x,y,z : s_B \Big(x\, b\, y \wedge y\, b'\, z \to \bigvee_{b\text{''}\in b\circ b'} x\, b''\, z\Big).$$

$\mathscr{C}$ is *strongly correct* for T if

$$T \models \forall x,z : s_B \Big(\exists y : s_B \left(x\, b\, y \wedge y\, b'\, z\right) \leftrightarrow \bigvee_{b\text{''}\in b\circ b'} x\, b''\, z\Big).$$

It can easily be checked that these correctness concepts are closely related to algebraic representation concepts (see, e. g., [14]).[6]

In what follows, let $\Sigma = \langle S, F, R, s_B\rangle$ and $\Sigma' = \langle S', F', R', s_B{}'\rangle$ be signatures for a constraint calculus $\mathscr{C}$, and let T and T' be Σ- and Σ'-theories for $\mathscr{C}$ as specified above, respectively.

Definition 5. A signature morphism $\sigma\colon \Sigma' \to \Sigma$ is said to be a $\mathscr{C}$*-theory morphism* from T' to T if $\sigma(s_B{}') = s_B$, if σ preserves the binary relation symbols for elements of B, and if for each model M of T, the σ-reduct of M is a model of T'.

[6] In more detail, let $\Sigma = \langle S, F, R, s_B\rangle$ be a signature for $\mathscr{C}$, and let T be a Σ-theory. Then T is weakly (strongly) correct for T if and only if for each Σ-model M of T, the assignment $b \mapsto b^M (\subseteq s_B^M \times s_B^M)$ defines a weak (strong) representation of $\mathscr{C}$.

Lemma 1. *Let $\sigma\colon \Sigma' \to \Sigma$ be a $\mathscr{C}$-*theory morphism *from T' to T. If $\mathscr{C}$ is weakly (resp. strongly) correct for T', then so is $\mathscr{C}$ for T.*

Proof. It is straightforward to see that $T' \models \phi$ implies $T \models \phi^\sigma$ by applying equation (1). $\square$

Lemma 1 is central for the justification of the heterogeneous proof method that we will use to verify composition tables. In more detail, at the place of theory T in this lemma we use a higher-order theory providing the intended semantics of a qualitative calculus (e. g., regular closed sets of a topological space as the relata of the RCC-8 relations). At the place of theory T' we axiomatize a first order theory providing an "intermediate semantics" (for RCC-8, e. g., one can use a fragment of the first order theory of RCC discussed by [8]). Then one has to define a signature morphism from T' to T and to prove that this is a $\mathscr{C}$-theory morphism. If the intermediate theory T' contains reasonably few axioms, the number of proof obligations that need to be checked by an interactive prover can be hold at a low level. Finally, one can prove the possibly huge number of composition table entries with respect to a FO theory by an automatic reasoner.

3 CASL, HETS, and Tools

In order to apply the proof method explained in the previous section, we used a proof management system that builds on the Common Algebraic Specification Language (CASL), which was developed by the *Common Framework Initiative for Algebraic Specification and Development* (CoFI). CASL allows for writing algebraic specifications that can be expressed in a many-sorted first order language (with partial function symbols). Basic CASL specifications consist of signature declarations and axioms characterizing the models to be described. These axioms, in turn, are first-order formulae or assertions regarding the definedness of partial function symbols. Going beyond first-order logic, CASL also provides constructs to state induction principles (called *sort generation constraints*) and datatype declarations. Furthermore, specifications may contain subsort declarations, whereby subsort inclusions are treated as embeddings. Finally, CASL also provides constructs for structured specifications, namely, translations, reductions, unions, and extensions of specifications (see [15] and [16], examples will be discussed in the following section).[7]

To specify the model classes of qualitative calculi that employ higher-order constructs (e. g., for the real numbers, for metric and topological spaces). we used a higher-order extension of CASL, HASCASL (see [18]), which is based on the partial λ-calculus. CASL's structuring constructs (union, translation, hiding, etc.) are independent of the underlying logical system and hence can be used for HASCASL as well. In the context of this paper, the distinguishing feature of CASL and its extensions is that it is possible to specify theory morphisms (as discussed in the previous section).

[7] In this context it is worth mentioning that there exists a variant of the CASL language, namely CASL-DL, which realizes a strongly typed variant of OWL-DL in CASL syntax [17]. This can be used to relate the semantics of constraint calculi with domain ontologies such as DOLCE (`http://www.loa-cnr.it/DOLCE.html`).

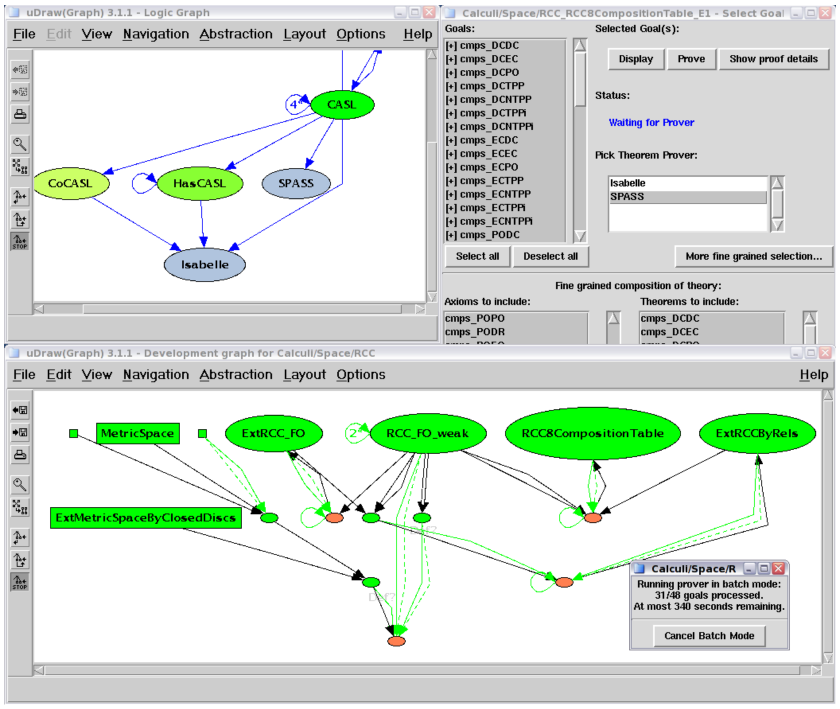
The proof management system HETS (*Heterogeneous Tool Set*, [19]) developed at the University of Bremen, Germany, is the main analysis tool for CASL and its extensions. HETS integrates a parser and a type-checker for heterogeneous specifications. A graphical interface allows for presenting the development graph (showing the specification structure) of CASL specifications as well as the logic graph presenting the underlying logics. HETS provides an interface to translate CASL specifications into Isabelle theory files. Of course, HETS also supports HASCASL specifications. In more detail, HETS provides a tool for heterogeneous multi-logic specification. It is based on a graph of logics and languages (formalized as so-called institutions), their tools, and their translations. This provides a clean semantics of heterogeneous specification, as well as a corresponding proof calculus. For proof management, the calculus of development graphs (known from other large-scale proof management systems) has been adapted to heterogeneous specification. Development graphs provide an overview of the (heterogeneous) specification module hierarchy and the current proof state, and thus may be used for monitoring the overall correctness of a heterogeneous development.

As a higher-order proof assistant system we use Isabelle [11]. Isabelle provides a rich language for expressing mathematical formulae and contains tools for proving these formulae in a logical calculus. As an automated theorem prover for first-order logic, we use SPASS [9] and VAMPIRE [10]. A useful feature of HETS is that the user can select between different reasoners or can use reasoning services provided by MathServ Broker [20].

4 Verification of Composition Tables

We now briefly sketch how the proof method presented above can be used to verify the correctness of composition tables. For the sake of simplicity, we show how the RCC-8 composition table can be shown to be weakly correct with respect to the closed discs semantics for metric spaces. We use variants of the CASL specifications presented in [21]: Based on a specification of metric spaces, one can easily build a higher-order (HASCASL) specification of closed discs, which axiomatizes the target theory of the theory morphism in Lemma 1. In order to specify the source theory of this morphism, we use a fragment of Bennett's first order theory of RCC [8]:

spec RCC_FO_WEAK =
  sort Elem
  pred __C__: Elem × Elem;
  $\forall x, y : \text{Elem}$
  • $x \, \mathrm{C} \, y \Rightarrow x \, \mathrm{C} \, x$ (C_non_null)
  • $x \, \mathrm{C} \, y \Rightarrow y \, \mathrm{C} \, x$ (C_sym)
  • $(\forall z : \text{Elem} \bullet z \, \mathrm{C} \, x \Leftrightarrow z \, \mathrm{C} \, y) \Rightarrow x = y$ (C_id)
  • $\exists x : \text{Elem} \bullet x \, \mathrm{C} \, x$ (C_non_triv)
then %**def**
  sort $\text{Reg} = \{x : \text{Elem} \bullet x \, \mathrm{C} \, x\}$
end


uDraw(Graph) 3.1.1 - Logic Graph
File Edit View Navigation Abstraction Layout Options Help
CASL
CoCASL
HasCASL
SPASS
Isabelle
uDraw(Graph) 3.1.1 - Development graph for Calculi/Space/RCC
MetricSpace
ExtRCC_FO
RCC_FO_weak
RCC8CompositionTable
ExtRCCByRels
ExtMetricSpaceByClosedDiscs

Calculi/Space/RCC_RCC8CompositionTable_E1 - Select Goal

Goals:

[+] cmps_DCDC
[+] cmps_DCEC
[+] cmps_DCPO
[+] cmps_DCTPP
[+] cmps_DCNTPP
[+] cmps_DCTPPi
[+] cmps_DCNTPPi
[+] cmps_ECDC
[+] cmps_ECEC
[+] cmps_ECPO
[+] cmps_ECTPP
[+] cmps_ECNTPP
[+] cmps_ECTPPi
[+] cmps_ECNTPPi
[+] cmps_PODC

Selected Goal(s):

Status: Waiting for Prover

Pick Theorem Prover: Isabelle, SPASS (selected)

Fine grained composition of theory:

Axioms to include: cmps_POPO, cmps_PODR

Theorems to include: cmps_DCDC, cmps_DCEC

Calculi/Space/R

Running prover in batch mode:
31/48 goals processed.
At most 340 seconds remaining.

Cancel Batch Mode

Fig. 2: Verifying a composition table with HETS.

We can use CASL's structuring constructs to extend this specification by definitions of further RCC relations (in particular, the RCC-8 relations). The composition table of RCC-8 is weakly correct for this extended theory. This can be expressed in CASL as follows:

spec RCC8COMPOSITIONTABLE[RCC_FO_WEAK] =
  EXTRCCBYRELS[RCC_FO_WEAK]
then %**implies**
  $\forall x, y, z : \mathrm{Reg}$
  • $x\,\mathrm{DC}\,y \wedge y\,\mathrm{DC}\,z \Rightarrow x\,1\,z$ (cmps_DCDC)
  • $x\,\mathrm{DC}\,y \wedge y\,\mathrm{EC}\,z \Rightarrow x\,\mathrm{DC}\,z \vee x\,\mathrm{EC}\,z \vee x\,\mathrm{PO}\,z \vee x\,\mathrm{TPP}\,z \vee x\,\mathrm{NTPP}\,z$ (cmps_DCEC)
  ...
  • $x\,\mathrm{DC}\,y \wedge y\,\mathrm{TPP}i\,z \Rightarrow x\,\mathrm{DC}\,z$ (cmps_DCTPPi)
  • $x\,\mathrm{DC}\,y \wedge y\,\mathrm{NTPP}i\,z \Rightarrow x\,\mathrm{DC}\,z$ (cmps_DCNTPPi)
  ... (see Table 1)
end

Fig. 2 shows a session of the Heterogeneous Tool Set. The upper left window depicts the graph of logics that can be used. The lower window contains the so-called development graph, showing the specification modules and the open proof obligations. The theorems listed beyond the annotated keyword **then %implies** in the previous specification are proof obligations that were proven by SPASS (upper right window in Fig. 2). It should not go unmentioned that the proof of weak correctness of the RCC-8 composition table takes less than 3 minutes on an industrial-standard PC. Finally, one has to prove that the signature morphism defined in the next specification is a theory morphism. This task was conducted by interactively using Isabelle.[8]

```
logic HasCASL
view RCC_FO_WEAK_To_CLOSEDDICS:
    RCC_FO_WEAK
to { EXTMETRICSPACEBYCLOSEDDISCS[METRICSPACE]
      then %def
      type NotEmptyClosedDiscs = {X : ClosedDiscs • ¬X = ∅}
      preds __C__ : ClosedDiscs × ClosedDiscs
      ∀x,y : ClosedDiscs
      • x C y ⇔ ¬x disjoint y
    }
=   Elem ↦ ClosedDiscs,  Reg ↦ NotEmptyClosedDiscs
end
```

It is clear that we could analogously prove a corresponding theory morphism from the theory spanned by RCC_FO_WEAK to a higher order theory of regular closed subsets in a topological space. With respect to stronger correctness concepts, we just need to modify RCC8COMPOSITIONTABLE (see Def. 4) and consider strengthenings of RCC_FO_WEAK. Moreover, the correctness of other RCC calculi such as RCC-5 can be proven by reusing already verified theory morphisms into higher-order theories.[9]

5 Summary and Outlook

In this paper we presented a heterogeneous proof method that allows for verifying the correctness of composition tables of qualitative constraint calculi. This method is of particular interest if the semantics of the calculus at hand essentially builds on higher-order constructs or entities (such as sets, real numbers, Euclidean spaces, etc.). By this method, it is possible to exploit the strengths of different theorem proving tools, such as higher-order proof assistance systems and automatic (first order) reasoners. In this context, a heterogeneous proof management tool, such as HETS, also proved valuable for a clean development of the specifications used in the verification process.

In future research we will analyze how our proof method can be modified in order to verify the correctness of ternary constraint calculi as well. Our mid-term goal is to provide a library of verified constraint calculi that can be used for the development of applications.

[8] Specifications and proof scripts are available under `http://www.cofi.info/Libraries` in the folder `CASL-lib/Calculi/Space`

[9] We used analogous techniques to verify strong correctness of Allen's interval algebra with respect to its standard semantics in the real numbers. The method described here was also used to find axioms that validate the strong reading of composition tables.

Acknowledgments

This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition. We would like to thank Bernhard Nebel for helpful discussions and Klaus Lüttich for implementing the SPASS interface of HETS.

References

1. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In Nebel, B., Swartout, W., Rich, C., eds.: Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference (KR-92), Morgan Kaufmann (1992) 165–176
2. Cohn, A.G., Bennett, B., Gooday, J.M., Gotts, N.: RCC: A calculus for region based qualitative spatial reasoning. GeoInformatica **1** (1997) 275–316
3. Düntsch, I., Wang, H., McCloskey, S.: Relation algebras in qualitative spatial reasoning. Fundamenta Informaticae **39**(3) (1999) 229–249
4. Gerevini, A., Renz, J.: Combining topological and qualitative size constraints for spatial reasoning. In: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, Springer (1998) 220–234
5. Egenhofer, M.J.: Reasoning about binary topological relations. In Günther, O., Schek, H.J., eds.: Proceedings of the Second Symposium on Large Spatial Databases, SSD'91 (Zürich, Switzerland). Lecture Notes in Computer Science 525, Springer (1991) 143–160
6. Egenhofer, M.J., Franzosa, R.D.: Point set topological relations. International Journal of Geographical Information Systems **5** (1991) 161–174
7. Skiadopoulos, S., Koubarakis, M.: Composing cardinal direction relations. Artifical Intelligence **152**(2) (2004) 143–171
8. Bennett, B.: Logical Representations for Automated Reasoning about Spatial Relationships. PhD thesis, School of Computer Studies, The University of Leeds (1997)
9. Weidenbach, C., Brahm, U., Hillenbrand, T., Keen, E., Theobald, C., Topic, D.: SPASS Version 2.0. [22] 275–279
10. Riazanov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Communications **15**(2-3) (2002) 91–110
11. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL, A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science 2283. Springer (2002)
12. Bennett, B., Isli, A., Cohn, A.G.: When does a composition table provide a complete and tractable proof procedure for a relational constraint language? In: Proceedings of the IJCAI97 Workshop on Spatial and Temporal Reasoning, Nagoya, Japan (1997)
13. Ligozat, G., Renz, J.: What is a qualitative calculus? A general framework. In Zhang, C., Guesgen, H.W., Yeap, W.K., eds.: PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Proceedings. Lecture Notes in Computer Science 3157, Springer (2004) 53–64
14. Mossakowski, T., Schröder, L., Wölfl, S.: A categorical perspective on qualitative constraint calculi. In Wölfl, S., Mossakowski, T., eds.: Qualitative Constraint Calculi: Application and Integration, Workshop Proceedings. (2006) 28–39
15. Bidoit, M., Mosses, P.D.: CASL User Manual. Lecture Notes in Computer Science. Springer (2004)
16. Mosses, P.D., ed.: CASL Reference Manual. Lecture Notes in Computer Science 2960. Springer (2004)

17. Lüttich, K., Mossakowski, T., Krieg-Brückner, B.: Ontologies for the semantic web in casl. In Fiadeiro, J.L., Mosses, P.D., Orejas, F., eds.: Recent Trends in Algebraic Development Techniques, 17th International Workshop, WADT 2004. Lecture Notes in Computer Science 3423, Springer (2004) 106–125
18. Schröder, L., Mossakowski, T.: HasCASL: Towards integrated specification and development of Haskell programs. In Kirchner, H., Ringeissen, C., eds.: Algebraic Methodology and Software Technology, 2002. Lecture Notes in Computer Science 2422. Springer (2002) 99–116
19. Mossakowski, T.: Heterogeneous specification and the heterogeneous tool set. Habilitation thesis, University of Bremen (2005)
20. Zimmer, J., Kohlhase, M.: System description: The MathWeb software bus for distributed mathematical reasoning. [22] 139–143
21. Wölfl, S., Mossakowski, T.: CASL specifications of qualitative calculi. In Cohn, A.G., Mark, D.M., eds.: Spatial Information Theory: Cognitive and Computational Foundations, Proceedings of COSIT'05. Lecture Notes in Computer Science 3693, Springer (2005)
22. Voronkov, A., ed.: Automated Deduction – CADE-18, 18th International Conference on Automated Deduction, Proceedings. Lecture Notes in Computer Science 2392, Springer (2002)