

CASL Specifications of Qualitative Calculi

Stefan Wöflf¹ and Till Mossakowski²

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

² Department of Computer Science, University of Bremen, P.O. Box 330440,
28334 Bremen, Germany
till@tzi.de

Abstract. In AI a large number of calculi for efficient reasoning about spatial and temporal entities have been developed. The most prominent temporal calculi are the point algebra of linear time and Allen's interval calculus. Examples of spatial calculi include mereotopological calculi, Frank's cardinal direction calculus, Freksa's double cross calculus, Egenhofer and Franzosa's intersection calculi, and Randell, Cui, and Cohn's region connection calculi.

These calculi are designed for modeling specific aspects of space or time, respectively, to the effect that the class of intended models may vary widely with the calculus at hand. But from a formal point of view these calculi are often closely related to each other. For example, the spatial region connection calculus RCC5 may be considered a coarsening of Allen's (temporal) interval calculus. And vice versa, intervals can be used to represent spatial objects that feature an internal direction.

The central question of this paper is how these calculi as well as their mutual dependencies can be axiomatized by algebraic specifications. This question will be investigated within the framework of the *Common Algebraic Specification Language* (CASL), a specification language developed by the *Common Framework Initiative for algebraic specification and development* (COFI). We explain scope and expressiveness of CASL by discussing the specifications of some of the calculi mentioned before.

1 Introduction: Calculemus!

In the past 25 years qualitative spatial and temporal reasoning has evolved to a discipline in its own right within AI. Qualitative reasoning aims at describing the common-sense background knowledge on which our human perspective on the physical reality is based. The calculi, that is, formal languages and reasoning techniques, developed in this research area are of special interest for all application fields that rely on human-machine interaction in static or dynamically changing spatial environments. For example, some of these calculi may be implemented for handling spatial GIS queries efficiently and some may be used for navigating, and communicating with, a mobile robot.

One will hardly find an exact definition of the notion of qualitative reasoning, but the whole research area has been very much inspired by Hayes' naïve manifesto [15]. In fact, in different areas of mathematics and physics very expressive formalisms for reasoning about space and time have been developed. But from a computer scientist's

point of view most of these formalisms are too expressive. Since there is an inevitable trade-off between the expressiveness of a language and the computational costs for reasoning with its formulae, expressiveness can get a crucial point when these calculi are to be integrated in applications. Thus the fundamental idea of qualitative reasoning is to restrict the vocabulary of rich mathematical theories in such a way that diversified aspects of these theories are treated within distinguished decidable fragments with simple qualitative (i. e., non-metrical) languages.

From this starting point a large number of calculi for efficient reasoning about spatial and temporal entities have been proposed in the literature. The most prominent temporal calculi are the so-called point algebra, which deals with instants of a given linear flow of time, and Allen's interval algebra [1], which describes possible relations between intervals in linear flows of time (cf. Fig. 1). Analogous calculi have been proposed for more general classes of models such as branching flows of time [e. g., 5] or even structures, where flows of time are just required to satisfy the conditions of a partial order [e. g., 6]. Despite these one-sorted calculi, also many-sorted calculi have been proposed. For example, Vilain's point-interval calculus [24] deals with instants and intervals in linear flows of time and may be considered a combination of the point algebra and the interval algebra.

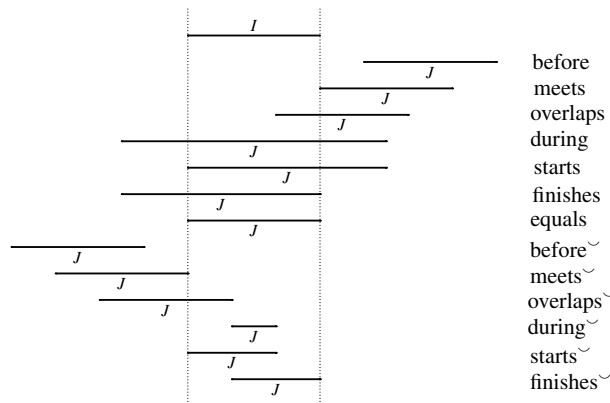


Fig. 1. Allen's interval relations

Examples of spatial calculi include mereotopological calculi (e. g., [2]), Frank's cardinal direction calculus [11], Freksa's double cross calculus [13], Egenhofer and Franzosa's 4- and 9-intersection calculi [9, 10], Ligozat's flip-flop calculus [16], and various region connection calculi proposed by Randell et al. [21], Cohn et al. [7], Düntsch et al. [8], and Gerevini and Renz [14]. It is interesting to see that even these few calculi employ concepts from a wide range of mathematical theories. Some of them are based on geometrical notions such as lines, half-planes, and angels, some describe relations between physical objects in terms of point set topology, and some include qualitative size information.

Interestingly, some spatial calculi are closely related to the temporal calculi mentioned previously. For example, the *2-point calculus* describes points of the plane and their relationships in terms of the point-to-point relations between their coordinates. This means, that one considers for each dimension one of the three possible point-to-point relations $<$, $=$, and $>$ between the point coordinates. The relation $(=, >)$, for instance, expresses the relation “north-to”, $(>, >)$ corresponds to “north-east-to”, etc. For this reason the 2-point calculus is often referred to as *cardinal direction calculus* in the literature. Analogously, the *rectangle calculus* describes possible relations between rectangles in the plane by comparing their coordinate projections in terms of the interval algebra.

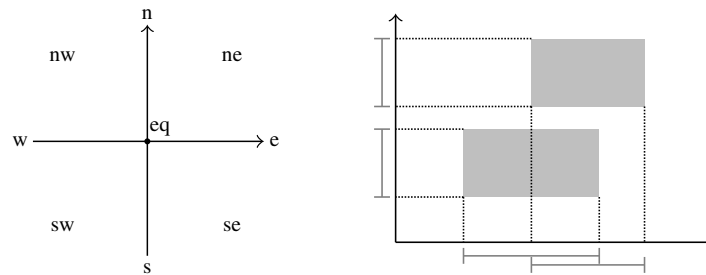


Fig. 2. Spatial calculi derivable from temporal algebras: The cardinal direction calculus and the rectangle calculus

To sum up this little discussion, researchers in the domain of qualitative reasoning face a vast, still increasing amount of calculi (much more than the rather incomplete list of calculi previously mentioned can indicate). On the other hand, many of these calculi are closely related to each other: some are simple extensions of others, some show similarities on the syntactic level, some have related classes of intended models, i. e., they are based on more or less the same background theory. Thus the guiding question of this paper is how to present qualitative calculi within a common framework in such a way that the mutual dependencies between them and their respective background theories becomes more transparent. In fact, transparency is an important issue for both avoiding redundancies and ensuring reusability of these calculi.

In our opinion, a naïve ontological classification system of qualitative or other calculi will not be able to fulfill these requirements in an adequate manner. A calculus classified as, say, “temporal” will always be a calculus about *temporal entities* such as instants or intervals, and can hardly be subsumed under the term “spatial calculus”. Moreover, connections between temporal and spatial calculi like the ones previously mentioned seem inexpressible in any ontology of such calculi. For this reason we propose to present qualitative calculi by means of algebraic specification, as was already suggested by Frank [12]. More exactly, we explain how to develop such specifications within the *Common Algebraic Specification Language (CASL)*.

The paper is organized as follows: In section 2 we briefly explain fundamental notions related to qualitative reasoning in more detail. Section 3 provides a short intro-

duction into CASL and its extension HASCASL. Then in section 4 we discuss algebraic specifications of the concepts introduced in section 2 in an informal manner.

2 Qualitative Calculi

To exemplify the most fundamental ideas of qualitative reasoning, let us discuss the point algebra for linear time in more detail. From a model-theoretical perspective this point algebra aims at describing the class of *linear flows of time*, i. e., first order structures $\mathcal{T} = \langle T, < \rangle$ that are models of the following axioms:

Irreflexivity: $\forall x x \not< x$

Transitivity: $\forall xyz (x < y \wedge y < z \rightarrow x < z)$

Linearity: $\forall xy (x < y \vee x = y \vee y < x)$

From an ontological point of view, the point algebra takes instants of time as *primary objects*, and states that these entities are linearly ordered. If we shift consideration from these primary objects towards the relations between them, we can state the following observations: From $<$ being irreflexive and transitive, it follows that the relations $<$, $=$, and $>$ (where $>$ is just defined as the converse of relation $<$) are pairwise disjoint. Linearity guarantees that these relations are jointly exhaustive, i. e., for each pair of instants t and t' , one of the relations $t < t'$, $t = t'$, or $t > t'$ holds. Speaking algebraically, the set $\{<, =, >\}$ forms a jointly exhaustive and pairwise disjoint (JEPD) system of relations.

Often temporal or spatial information is imprecise, for example, when we only have the information that instant t is not before instant t' , or that instants t and t' are distinct. In this situation it becomes interesting to consider not only the *base relations* $<$, $=$, and $>$, but also arbitrary unions of them. Obviously, the system of unions of base relations defines an atomic Boolean algebra with the base relations as atoms. By the way, since the system of base relations is JEPD, unions of base relations may also be represented as *sets* of base relations.

Reasoning problems, then, are usually formulated as constraint satisfaction problems. A constraint network is a finite set of constraints where each 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 relation R (a set of base relation) defined between the domains of x and y . Typical reasoning tasks are then to determine whether a constraint set is satisfiable, to check that some constraint is entailed by a constraint network, and to compute an equivalent minimal constraint set — it is not hard to see that all these reasoning tasks are equivalent under Turing reductions.

A crucial aspect for developing efficient algorithms for qualitative spatial and temporal reasoning is the fact that the underlying model classes usually contain infinite models. Hence, in order to test satisfiability of constraint networks, it is not feasible to enumerate all models until one finds a satisfying model. For this reason other techniques (such as path-consistency algorithms) must be applied for testing satisfiability. Many of these techniques, in turn, rely on semantically verified composition tables that list which relations are consistent when two base relations are composed. For

Table 1. The composition table of the point algebra for linear time

	<	>	=
<	<	<, =, >	<
>	<, =, >	>	>
=	<	>	=

example, Table 1 presents the composition table of the point algebra for linear time.¹ But a set of base relations together with a composition table satisfying some minimal conditions defines a *relation algebra* on the set of all unions of base relations. For this reason studying relation algebras has become a central aspect in the field of qualitative reasoning.

3 CASL and Friends

The Common Algebraic Specification Language (CASL) is a specification language, 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.

In the sequel we will explain these concepts in more detail (for a full discussion see Bidoit and Mosses [4] and Mosses [19]).

3.1 Constructing Specifications

To start with, let us briefly explain the formal underpinnings of CASL specifications. As said before, CASL allows for specifying first order theories with partial function symbols. More precisely, CASL accepts languages with (*many-sorted*) *signatures* $\Sigma = \langle S, TF, PF, R \rangle$ such that:

¹ Note that there are (at least) two ways of reading such composition tables, the *extensional* and the *consistency-based* reading [cf. 3]. Following, we will only use the extensional reading, which means that the algebraic function of composing relations (as used, for example, in relation algebras) coincides with its set-theoretical characterization. In the case of the point algebra, for example, the consistency-based reading is correct for the class of linear flows of time, while the extensional reading is only correct for the class of dense linear flows of time without endpoints.

- S is a (finite) set of sorts.
- For each $(w, s) \in S^* \times S$, $TF_{w,s}$ and $PF_{w,s}$ are disjoint sets of total and partial function symbols, respectively (tuples $w \in S^*$ are referred to as *sort profiles*).
- 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 partial first-order structures: Given a signature Σ , a Σ -*model* is a structure consisting of non-empty carrier sets s^M (for each sort $s \in S$), partial and total functions $f^M : w^M \rightarrow s^M$ (for each function symbol $f \in PF_{w,s}$ or $f \in TF_{w,s}$, respectively), and relations $r^M \subseteq w^M$ (for each relation symbol $r \in R_w$).

The most fundamental notion related to extensions, unions, and translations of specifications is that of a signature morphism. To explain this notion, let $\Sigma = \langle S, TF, PF, R \rangle$ and $\Sigma' = \langle S', TF', PF', R' \rangle$ be signatures. Then a *signature morphism* $\Sigma \rightarrow \Sigma'$ is a 4-tuple $\sigma = \langle \sigma^s, \sigma^t, \sigma^p, \sigma^r \rangle$ consisting of maps (families of maps, resp.):

- $\sigma^s : S \rightarrow S'$,
- $\sigma_{w,s}^t : TF_{w,s} \rightarrow TF'_{\sigma^s(w), \sigma^s(s)}$,
- $\sigma_{w,s}^p : PF_{w,s} \rightarrow TF'_{\sigma^s(w), \sigma^s(s)} \cup PF'_{\sigma^s(w), \sigma^s(s)}$, and
- $\sigma_w^r : R_w \rightarrow R'_{\sigma^s(w)}$.

That is, partial function symbols may be mapped to total function symbols, but not vice versa.

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

$$s^{M'|_\sigma} := \sigma^s(s)^{M'}, \quad f^{M'|_\sigma} := \sigma^{t/p}(f)^{M'}, \quad \text{and} \quad r^{M'|_\sigma} := \sigma^r(r)^{M'}.$$

We are now ready to explain some fundamental notions in more detail.

Basic Specification. The most simple kind of CASL specifications, called *basic specifications*, are asserted by the keyword **spec** and have the form

$$\mathbf{spec} \text{ SpecName} = \text{Spec}$$

where *SpecName* is the name of the specification and *Spec* is a list of signature declarations and first order axioms.

Extension. *Extensions* have the form

$$\text{Spec1} \mathbf{then} \text{Spec2}$$

where *Spec1* is a specification or a specification name, and *Spec2* is a specification that extends the signature of *Spec1* and/or adds additional axioms. CASL allows to indicate the type of the extension: *Definitional extensions* are introduced by annotating the keyword **then** with **%def**. From a model-theoretical point of view, definitional extensions are justified when each model of *Spec1* can be uniquely extended to a model of the specification *Spec1-then-Spec2*. *Implied extensions* state some theorems of the original specification, i. e., *Spec1* and *Spec1-then-Spec2* have the same model class. This kind of extension is introduced by the annotated keyword **then %implies** and is considered well-formed only if *Spec1* and *Spec1-then-Spec2* have the same signature.

Union. It is possible to join two specifications, i. e., to build their *union*. The signature of a union of specifications *Spec1* and *Spec2* is just the union of the signatures of *Spec1* and *Spec2*. The models of the union are exactly those models of the union signature whose reducts are models of *Spec1* and *Spec2*, respectively. Unions of specifications obey the “same name, same thing” principle, which means that each symbol contained in both signatures has a single interpretation in each model of their union. Unions are declared by

$$\textit{Spec1 and Spec2.}$$

Translations. *Translations* are renamings of sort symbols and/or signature symbols and thus provide a signature morphism from the source specification into the specification resulting from the translation. The models of the translation are exactly those models of the result specification whose reducts along the morphism are models of the source specification. Translations are declared by

$$\textit{Spec with SymbolMappings.}$$

Reductions. CASL also provides constructs to restrict the signature of a given specification. It is possible to hide symbols or, alternatively, declare the symbols that are revealed. Reductions can be declared by

$$\textit{Spec hide Symbols and Spec reveal SymbolMappings.}$$

Parameterization and Instantiations. CASL also allows parameterized specifications, which are written as:

$$\textit{spec Spec1[Spec2] \dots [SpecN] = Spec.}$$

Sometimes it is necessary to instantiate a previously declared specification via a symbol mapping before it is used in a parameterized specification. This can be obtained by

$$\textit{Spec1[Spec2 fit SymbolMappings].}$$

Views. A *view* provides a signature morphism (defined by a symbol mapping) between two specifications. Actually, it is required to be a *theory morphism* (or *interpretation of theories*), which means that each model of the target specification induces (when reduced via the signature morphism) a model of the source specification. Views are declared by

$$\textit{view View : Spec1 to Spec2 = SymbolMapping.}$$

The possibility of specifying views is one of the distinguished features of CASL.

3.2 HASCASL

HASCASL (see [22]) is a higher-order extension of CASL based on the partial λ -calculus. The user declared sorts are used to generate *higher types* by closing the set of types under total and partial function spaces (written by $t_1 \rightarrow t_2$ and $t_1 \rightarrow ?t_2$,

resp.). Predicates (written $Pred\ t$) are coded as partial functions into a singleton type (i. e., only the domain provides relevant information). In fact, we often will need HASCASL for specifying the model classes of qualitative calculi (e. g., for the real numbers, for metric and topological spaces). CASL's structuring constructs (union, translation, hiding, etc.) are independent of the underlying logical system and hence can be used for HASCASL as well.

3.3 Tools

HETS. The *Heterogeneous Tool Set* (HETS) [18], which is 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 logic(s). It is possible to translate a CASL specification into XML-, \LaTeX -, and other formats. HETS also provides an interface to translate CASL specifications into Isabelle theory files. Of course, HETS also supports HASCASL specifications.

Isabelle. Isabelle is a generic proof assistant, which is developed by L. C. Paulson (University of Cambridge, UK) and T. Nipkow (Technical University of Munich, Germany). Isabelle provides a rich language for expressing mathematical formulae and contains tools for proving these formulae in a logical calculus. Its main application fields are the formalization of mathematical proofs and the formal verification of computer languages, protocols, computer hardware, and software specifications.

A main feature of Isabelle is that it is not restricted to a single formal calculus since it supports higher-order logic, axiomatic set theory, etc. We use Isabelle/HOL, the coding of higher-order logic in Isabelle. For a more comprehensive introduction we refer to Nipkow et al. [20].

4 Specifications of Qualitative Calculi

4.1 Relation Algebras

To start with, let us first discuss some specifications related to relation algebras. A *relation algebra* is a Boolean algebra with complement (its elements are referred to as *relations*) and with a distinguished element id (the *identity relation*), a unary total function \smile assigning to each relation its converse relation (algebraically, its *involution*), and a binary total function \circ assigning to each pair of relations their composition. Note that here the term “relation” is used in an abstract manner, i. e., independently of its usual set-theoretical interpretation. Rather, these functions are characterized implicitly by the axioms listed in the following specification.

```

spec RELATIONALALGEBRA =
  BOOLEANALGEBRAWITHCOMPL with sort  $Elem \mapsto Rel$ 
then
  ops  $id : Rel$ ;
   $\smile : Rel \rightarrow Rel$ ;
   $\circ : Rel \times Rel \rightarrow Rel$ , assoc, unit  $id$ ;

```

```

     $\forall x, y, z : Rel$ 
    •  $(x^\sim)^\sim = x$  %(inv_idempot)%
    •  $(x \sqcup y)^\sim = x^\sim \sqcup y^\sim$  %(inv_cup)%
    •  $(-x)^\sim = -x^\sim$  %(inv_compl)%
    •  $(x \circ y)^\sim = y^\sim \circ x^\sim$  %(inv_cmps)%
    •  $(x \circ y) \sqcap z^\sim = 0 \Rightarrow (y \circ z) \sqcap x^\sim = 0$  %(triangle)%
then %implies
     $\forall x, y, z : Rel$ 
    •  $(x \sqcup y) \circ z = (x \circ z) \sqcup (y \circ z)$  %(cmps_cup_rdistib)%
    •  $z \circ (x \sqcup y) = (z \circ x) \sqcup (z \circ y)$  %(cmps_cup_ldistrib)%
    •  $(x^\sim \circ -(x \circ y)) \sqcap y = 0$  %(RelAlg)%
end

```

We may define a partial order on a relation algebra in exactly the same way as we could introduce it for arbitrary Boolean algebras. This gives us some nice corollaries, which could easily be proven by Isabelle. For example, the composition of relations behaves monotonic with respect to the canonical partial order.

```

spec EXTRELATIONALALGEBRABYPARTIALORDER[RELATIONALALGEBRA] = %def
    EXTBOOLEANALGEBRABYPARTIALORDER[BOOLEANALGEBRA]
    with sort Elem  $\mapsto$  Rel
then %implies
     $\forall x, y, x', y' : Rel$ 
    •  $x \leq x' \wedge y \leq y' \Rightarrow x \circ y \leq x' \circ y'$  %(cmps_monotonic)%
    •  $x \leq id \Rightarrow x^\sim = x$  %(inv_below_id)%
end

```

In the following, we will be mainly interested in *atomic relation algebras*. An *atom* of a Boolean algebra is a non-zero element x such that the zero element is the only element y with $y < x$. A relation algebra (or Boolean algebra) is said to be *atomic* if for each non-zero element x , there exists an atom a with $a \leq x$. Hence the unary predicate “Atom” defines a genuine subsort.

```

spec ATOMICRELATIONALALGEBRA =
    RELATIONALALGEBRA
and ATOMICBOOLEANALGEBRA with sort Elem  $\mapsto$  Rel, AtomElem  $\mapsto$  AtomRel
end

```

As explained in section 2, often an (abstract) atomic relation algebra can be constructed from a set of base relations and a composition table. In CASL this procedure can be reconstructed as follows: First we define relations (i. e., sort *Rel*) as arbitrary sets of base relations such that base relations correspond to singleton sets of base relations.

```

spec SETREPRESENTATIONOFRELATIONS [sort BaseRel] = %def
local { SET [sort BaseRel fit Elem  $\mapsto$  BaseRel]
    with  $\sqcup \_ \mapsto \sqcup \_$ ,  $\sqcap \_ \mapsto \sqcap \_$ ,  $\subseteq \_ \mapsto \subseteq \_$  }
within
    free type Rel ::= sort Set[BaseRel]
    sort BaseRel < Rel
    ops 0, 1 : Rel;

```

```

    - : Rel → Rel;
    -□-, -□_ : Rel × Rel → Rel
preds -∈_ : BaseRel × Rel;
    -□_ : Rel × Rel
    ∀x : BaseRel; r : Rel
    • x = {x}
    • x ∈ 1 ∧ ¬x ∈ 0
    • x ∈ -r ⇔ ¬x ∈ r
then %implies
    ...
end

view SETREPRESENTATION_AS_ATOMICBOOLEANALGEBRA [sort BaseRel] :
    ATOMICBOOLEANALGEBRA
to SETREPRESENTATIONOFRELATIONS [sort BaseRel]
    = Elem ↦ Rel, AtomElem ↦ BaseRel
end

```

In a second step we generate a relation algebra-like structure by extending the functions “composition” and “involution” (as defined for base relations) to total functions on all relations. In fact, not each composition table defines a relation algebra since many such constructed structures violate the associativity axiom of relation algebras [see, e. g., 17]. The view contained in the following small library states that we obtain a genuine relation algebra if the composition table satisfies certain conditions. For the sake of simplicity, our specification of composition tables includes both the the composition function and the involution function for base relations.

```

spec COMPOSITIONTABLE =
    sorts BaseRel < Rel
    ops id : BaseRel;
    0, 1 : Rel;
    -~ : BaseRel → BaseRel;
    -○_ : BaseRel × BaseRel → Rel;
    -_ : Rel → Rel;
    -□_ : Rel × Rel → Rel, assoc, idem, comm, unit 1
    ∀x : BaseRel
    • x ○ id = x ∧ id ○ x = x
    • id~ = id
    • (x~)~ = x
end

spec CONSTRUCTRELATIONALGEBRA [sort BaseRel] [COMPOSITIONTABLE] = %def
    SETREPRESENTATIONOFRELATIONS [sort BaseRel]
then %def
    ops id : Rel;
    -~ : Rel → Rel;
    -○_ : Rel × Rel → Rel;
    ∀x, y : BaseRel; r, s : Rel
    • x ∈ r~ ⇔ x~ ∈ r

```

```

    •  $x \in (r \circ s) \Leftrightarrow \exists y, z : BaseRel \bullet y \in r \wedge z \in s \wedge x \in (y \circ z)$ 
then %implies
    op  $\_ \circ \_ : Rel \times Rel \rightarrow Rel$ , unit id;
end

view CONSTRUCTEDRELATIONALALGEBRA_AS_ATOMICALRELATIONALALGEBRA
    [sort BaseRel] [GOODCOMPOSITIONTABLE]:
    ATOMICALRELATIONALALGEBRA
to CONSTRUCTRELATIONALALGEBRA [sort BaseRel] [GOODCOMPOSITIONTABLE]
    =  $Rel \mapsto Rel, AtomRel \mapsto BaseRel$ 
end

```

Let us now turn to the semantic level. First we define the concept *algebra of binary relations* (BRA). Given a set X , an algebra of binary relations is a Boolean subalgebra of the set algebra of all binary relations on X that contains the identity relation and is closed with respect to involution and composition (in their usual set-theoretical meaning). Of course, each algebra of binary relations is a relation algebra. But contrary to Boolean algebras (cf. Stone's representation theorem), it is in general not the case that each relation algebra can be represented as an algebra of binary relations.

```

logic HASCASL

spec BINARYRELATIONS [sort Elem] = %mono
    SET
then %mono
    type Relation ::= abs(rep : Set(Elem × Elem))
end

spec SETALGEBRAOFBINARYRELATIONS =
    BINARYRELATIONS [sort Elem]
then
    type Rel < Relation
    ops  $0, 1 : Rel$ ;
     $- : Rel \rightarrow Rel$ ;
     $\_ \sqcup \_, \_ \sqcap \_ : Rel \times Rel \rightarrow Rel$ 
     $\forall r, s : Rel$ 
    •  $rep(0) = \emptyset$ 
    •  $rep(1) = allSet$ 
    •  $rep(r \sqcup s) = rep(r) \cup rep(s)$ 
    •  $rep(r \sqcap s) = rep(r) \cap rep(s)$ 
    •  $rep(-r) = rep(1) \setminus rep(r)$ 
end

spec ALGEBRAOFBINARYRELATIONS =
    SETALGEBRAOFBINARYRELATIONS
then
    ops  $id : Rel$ ;
     $\_ \smile : Rel \rightarrow Rel$ ;
     $\_ \circ \_ : Rel \times Rel \rightarrow Rel$ ;
     $\forall r, s : Rel; x, y : Elem$ 

```

```

    •  $(x, y) \in \text{rep}(r \circ s) \Leftrightarrow \exists z: \text{Elem} \bullet (x, z) \in \text{rep}(r) \wedge (z, y) \in \text{rep}(s)$ 
    •  $(x, y) \in \text{rep}(r^\sim) \Leftrightarrow (y, x) \in \text{rep}(r)$ 
    •  $(x, y) \in \text{rep}(id) \Leftrightarrow x = y$ 
then %implies
    ops  $\_ \circ \_ : Rel \times Rel \rightarrow Rel$ , assoc, unit id
end

spec FULLALGEBRAOFBINARYRELATIONS [sort Elem] = %def
    { ALGEBRAOFBINARYRELATIONS with type Rel  $\mapsto$  Relation }
end

view ALGEBRAOFBINARYRELATIONS_AS_RELATIONALALGEBRA:
    RELATIONALALGEBRA to ALGEBRAOFBINARYRELATIONS
end

```

In the following, we describe how a strong representation of an (abstract) atomic relation algebra can be constructed from a concrete interpretation of its atoms, i. e., its base relations. For this assume that we have a model for the base relations of the relation algebra, that is, a JEPD system of relations on a non-void set *Elem*. Of course, we want to extend this model in a canonical manner to a model of all relations. Relations are here exactly those binary relations on *Elem* that can be written as a (set-theoretical) union of base relations. In order to be a strong representation, the model we aim at must be an algebra of binary relations. But the fact that the system of relations is JEPD ensures only that the set of all unions of base relations forms a set algebra. This set algebra need not be an algebra of binary relation since, in general, it need not contain the identity relation, be closed with respect to involution, or closed with respect to composition. But if it does—a base relation model satisfying these conditions will be referred to as *closed for composition*—the canonical extension of the base relation model defines an atomic algebra of binary relations. In fact, the final view in the following list of specifications states that the concrete relation algebra defined in this way provides a strong representation of the abstract algebra.

```

spec JEPDBASERELMODEL =
    BINARYRELATIONS [sort Elem]
then
    type BaseRel < Relation
     $\forall x, y: \text{Elem}; r, s: \text{BaseRel}$ 
    •  $\exists r: \text{BaseRel} \bullet (x, y) \in \text{rep}(r)$  % (JointlyExhaustive)%
    •  $\neg r = s \Rightarrow \text{rep}(r) \cap \text{rep}(s) = \emptyset$  % (PairwiseDisjoint)%
end

spec RELATIONSFROMBASERELMODEL [JEPDBASERELMODEL] = %def
    type Rel =  $\{x: \text{Relation} \bullet \exists X: \text{Set}(\text{BaseRel}) \bullet$ 
     $(\forall y, z: \text{Elem} \bullet (y, z) \in \text{rep}(x) \Leftrightarrow (\exists r: \text{BaseRel} \bullet r \in X \wedge (y, z) \in \text{rep}(r)))\}$ 
end

spec CONSTRUCTMODEL [COMPCLOSEDBASERELMODEL] = %def
    RELATIONSFROMBASERELMODEL [JEPDBASERELMODEL]

```

```

and ALGEBRAOFBINARYRELATIONS
then %def
  preds  $\_ \in \_ : \text{BaseRel} * \text{Rel};$ 
          $\_ \sqsubset \_ : \text{Rel} * \text{Rel}$ 
   $\forall x : \text{BaseRel}; r, r' : \text{Rel}$ 
  •  $x \in r \Leftrightarrow \text{rep}(x) \subseteq \text{rep}(r)$ 
  •  $r \sqsubset r' \Leftrightarrow \text{rep}(r) \subseteq \text{rep}(r')$ 
end

view RELATIONALGEBRA_FROM_BASERELMODEL [COMPCLOSEDBASERELMODEL]:
  RELATIONALGEBRA
to CONSTRUCTMODEL [COMPCLOSEDBASERELMODEL]
end

```

4.2 RCC5, RCC8, and Allen’s Interval Algebra

We are now ready to explain how spatial and temporal calculi (more exactly, their respective relation algebras) can be incorporated into the framework provided by the specifications presented in the previous section. Let us start with the region connection calculi RCC5 and RCC8. Both calculi describe relations between *regions*, which may be thought of as non-void, regular open (or alternatively, regular closed) subsets of a topological space. In RCC5 the following relations count as base relations: DR (“discrete”), PO (“partially overlap”), PP (“proper part”), PPI (the converse of PP), and EQ (“equal”). The set of RCC8 base relations is more fine-grained: DR splits into the relations DC (“disconnected”) and EC (“externally connected”) and PP into the relations TPP (“tangential proper part”) and NTPP (“non-tangential proper part”). These relations can be defined in terms of the topological closure operation: the relation DC holds between regular open sets X and Y if their closures do not intersect, X NTPP Y holds if the closure of X is contained in Y , etc.

The following library shows how the (abstract) relation algebra of RCC5 can be defined via CASL specifications:

```

spec RCC5BASERELATIONS = %mono
  free type BaseRel ::= dr | po | pp | ppi | eq
end

spec RCC5COMPOSITIONTABLE =
  sort BaseRel
  ops dr, po, pp, ppi, eq: BaseRel
  and COMPOSITIONTABLE with op id  $\mapsto$  eq
  then
  •  $dr \smile = dr$  % (sym_dr)%
  •  $po \smile = po$  % (sym_po)%
  •  $pp \smile = ppi$  % (inv_pp)%
  •  $ppi \smile = pp$  % (inv_ppi)%
  •  $pp \circ pp = pp$  % (cmps_pppp)%

```

```

    •  $pp \circ ppi = 1$                                 %(cmps_ppppi)%
    •  $pp \circ po = pp \sqcup po \sqcup dr$                 %(cmps_pppo)%
    •  $pp \circ dr = dr$                                 %(cmps_ppdr)%
    •  $ppi \circ pp = -dr$                                %(cmps_ppipp)%
    •  $ppi \circ ppi = ppi$                              %(cmps_ppippi)%
    ...
end

spec RCC5 =
  CONSTRUCTRELATIONALALGEBRA [RCC5BASERELATIONS]
    [RCC5COMPOSITIONTABLE fit op id:BaseRel → eq]
end

view RCC5_AS_ATOMICALGEBRA :
  ATOMICRELATIONALALGEBRA to RCC5
= AtomRel ↦ BaseRel
end

```

Obviously, the relation algebras of other qualitative calculi (such as RCC8 or Allen's interval algebra) can be specified in the very same manner. Moreover, we can declare natural views between these calculi as follows:

```

view RCC5_TO_RCC8 :
  RCC5
to { RCC8 then %def
  ops dr,pp,ppi : Rel
  •  $dr = dc \sqcup ec$ 
  •  $pp = tpp \sqcup ntp$ 
  •  $ppi = tppi \sqcup ntppi$  }
= sort BaseRel ↦ Rel
end

view RCC5_TO_ALLENIA :
  RCC5
to { ALLENIA then %def
  ops dr,po,pp,ppi,eq : Rel
  •  $dr = b \sqcup bi \sqcup m \sqcup mi$                 %[ Allen relations are denoted by the first ]%
  •  $po = o \sqcup oi$                             %[ letter of their names (cf. Fig. 1), i.e., ]%
  •  $pp = d \sqcup s \sqcup f$                         %[ b: before, bi: before involuted, etc. ]%
  •  $ppi = di \sqcup si \sqcup fi$ 
  •  $eq = e$  }
= sort BaseRel ↦ Rel
end

```

It is worth mentioning that a corresponding view from RCC8 to ALLENIA is not valid: The function $dc \mapsto b \sqcup bi$, $ec \mapsto m \sqcup mi$, $po \mapsto o \sqcup oi$, $tpp \mapsto s \sqcup f$, $ntpp \mapsto d$, etc., only defines a view from RCC8 to Allen's interval algebra as Boolean algebras, but not as relation algebras. To put it another way, RCC8 does not form a subalgebra of Allen's interval algebra.

Let us now illustrate how the semantic level of such relation algebras can be presented via CASL specifications. The following library describes a specific class of RCC5 models, which is definable for the Euclidean plane (interpreted as a metric space). Its final view says that one obtains a model of RCC5 if we interpret the RCC5 base relations over open discs in the Euclidean plane. More precisely, it states that the canonical interpretation of RCC5 over open discs in the Euclidean plane provides a strong representation of RCC5.²

```

logic HASCASL

view EUCLIDEANPLANE_AS_METRICSPACE :
  METRICSPACE to EUCLIDEANPLANE
  ...
end

spec RCC5OPENDISCBASERELMODEL[METRICSPACE] =
  SET
then op openDisc(r : Real; x : Elem) : Set Elem =  $\lambda y : Elem \bullet \text{dist}(x, y) < r$ 
  type OpenDisc = {X : Set Elem  $\bullet \exists r : Real; x : Elem \bullet X = \text{openDisc}(r, x)$ }
then BINARYRELATIONS [sort OpenDisc]
then %def
  ops drRel, poRel, ppRel, ppiRel, eqRel : Relation
  type BaseRel ::= ppRel | ppiRel | poRel | drRel | eqRel
   $\forall x, y : \text{OpenDisc}$ 
  •  $(x, y) \in \text{rep}(\text{drRel}) \Leftrightarrow x \text{ disjoint } y$ 
  •  $(x, y) \in \text{rep}(\text{poRel}) \Leftrightarrow \neg x \subseteq y \wedge \neg y \subseteq x \wedge \neg x \text{ disjoint } y$ 
  •  $(x, y) \in \text{rep}(\text{ppRel}) \Leftrightarrow x \subseteq y \wedge \neg x = y$ 
  •  $(x, y) \in \text{rep}(\text{ppiRel}) \Leftrightarrow y \subseteq x \wedge \neg x = y$ 
  •  $(x, y) \in \text{rep}(\text{eqRel}) \Leftrightarrow x = y$ 
end

spec RCC5OPENDISCMODEL[EUCLIDEANPLANE] = %def
  CONSTRUCTMODEL [RCC5OPENDISCBASERELMODEL [
    view EUCLIDEANPLANE_AS_METRICSPACE]
    fit sort Elem  $\mapsto$  OpenDisc]
end

view METRICSPACE_INDUCES_RCC5OPENDISCMODEL :
  RCC5
to RCC5OPENDISCMODEL [EUCLIDEANPLANE]
= ops pp  $\mapsto$  ppRel, ppi  $\mapsto$  ppiRel, po  $\mapsto$  poRel, dr  $\mapsto$  drRel, eq  $\mapsto$  eqRel
end

```

The development graph of these libraries, as output by HETS (a subgraph of it is depicted in Fig. 3), exhibits the mutual dependencies between the presented specifications. Dark arrows denote inclusions of theories; light arrows denote proof obligations (theory morphisms) generated by views.

² Weak representations of abstract relation algebras could be presented as CASL specifications as well, but not in terms of CASL views.

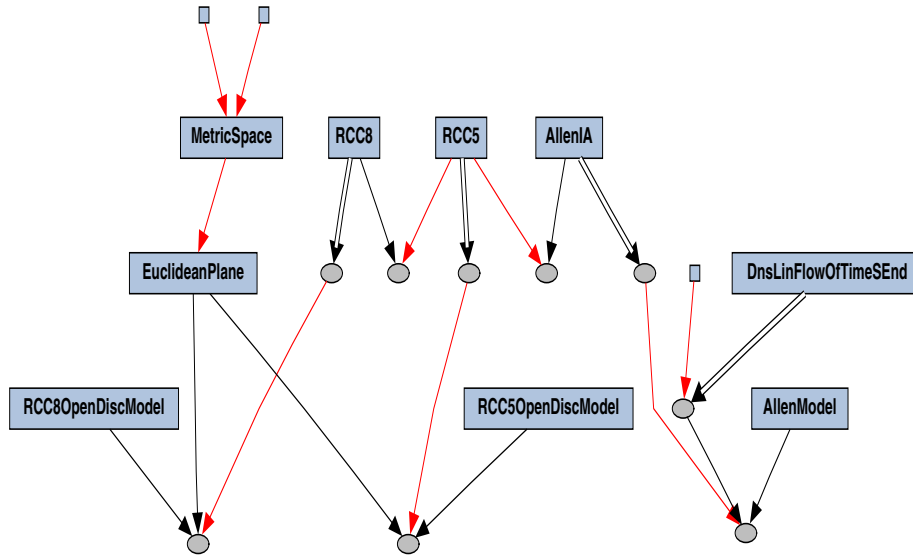


Fig. 3. Development graph of RCC5, RCC8, and Allen's interval algebra

5 Summary and Outlook

In this paper we discussed how qualitative calculi can be described via CASL specifications. We saw that CASL allows for an elegant representation of such calculi. Moreover, since the specifications presented here are built up in a modular way, we provide an easy interface for embedding other calculi into the CASL framework. Finally, CASL specifications ensure a high visibility of the mutual dependencies between qualitative calculi on both the syntactic and the semantic level, and hence may be considered superior to ontologies of such systems.

Because of lack of space, we could not show how these CASL specifications connect to theorem provers such as Isabelle. Furthermore, in this paper we could only explain a small fraction of algebraic theories related to spatial and temporal calculi. For example, the first-order theories of these calculi and Stell's Boolean connection algebras [23] should be integrated into CASL as well. Future work will also deal with the CSP languages of these qualitative calculi. In particular, we will investigate how these languages can be translated into MODALCASL, a sublanguage of CASL designed for specifying multi modal fragments of first order logic. Such translations would be particularly interesting for automated verifications of composition tables of the calculi mentioned in this paper.

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 Klaus Lüttich and Bernhard Nebel for helpful discussions. We also gratefully acknowledge the reviewers' critical comments, as well as their hints and suggestions.

References

- [1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- [2] B. Bennett. *Logical Representations for Automated Reasoning about Spatial Relationships*. PhD thesis, School of Computer Studies, The University of Leeds, 1997.
- [3] B. Bennett, A. Isli, and A. G. Cohn. 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.
- [4] M. Bidoit and P. D. Mosses. CASL User Manual. Lecture Notes in Computer Science. Springer, 2004.
- [5] M. Broxvall. The point algebra for branching time revisited. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI-2001)*, pages 106–121, Sep 2001.
- [6] M. Broxvall and P. Jonsson. Towards a complete classification of tractability in point algebras for nonlinear time. In *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming (CP-99)*, pages 129–143, Alexandria, VA, USA, 1999.
- [7] A. G. Cohn, B. Bennett, J. M. Gooday, and N. Gotts. RCC: A calculus for region based qualitative spatial reasoning. *GeoInformatica*, 1:275–316, 1997.
- [8] I. Düntsch, H. Wang, and S. McCloskey. Relation algebras in qualitative spatial reasoning. *Fundamenta Informaticae*, 39(3):229–249, 1999.
- [9] M. J. Egenhofer. Reasoning about binary topological relations. In O. Günther and H.-J. Schek, editors, *Proceedings of the Second Symposium on Large Spatial Databases, SSD'91 (Zürich, Switzerland)*, Lecture Notes in Computer Science 525, pages 143–160. Springer, 1991.
- [10] M. J. Egenhofer and R. D. Franzosa. Point set topological relations. *International Journal of Geographical Information Systems*, 5:161–174, 1991.
- [11] A. U. Frank. Qualitative spatial reasoning with cardinal directions. In H. Kaindl, editor, *Proceedings of the Seventh Austrian Conference on Artificial Intelligence*, Informatik-Fachberichte 287, pages 157–167. Springer, 1991.
- [12] A. U. Frank. One step up the abstraction ladder: Combining algebras - from functional pieces to a whole. In C. Freksa and D. M. Mark, editors, *Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT '99, Stade, Germany, August 25-29, 1999, Proceedings*, Lecture Notes in Computer Science 1661, pages 95–107. Springer, 1999.
- [13] C. Freksa. Using orientation information for qualitative spatial reasoning. In A. U. Frank, I. Campari, and U. Formentini, editors, *Spatio-Temporal Reasoning*, Lecture Notes in Computer Science 639, pages 162–178. Springer, 1992.
- [14] A. Gerevini and J. Renz. Combining topological and qualitative size constraints for spatial reasoning. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, pages 220–234. Springer, 1998.
- [15] P. J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert Systems in the Micro-Electronic Age*. Edinburgh University Press, 1978.

- [16] G. Ligozat. Qualitative triangulation for spatial reasoning. In A. U. Frank and I. Campari, editors, *Spatial Information Theory: A Theoretical Basis for GIS, International Conference COSIT '93, Marciana Marina, Elba Island, Italy, September 19-22, 1993, Proceedings*, Lecture Notes in Computer Science 716, pages 54–68. Springer, 1993.
- [17] G. Ligozat and J. Renz. What is a qualitative calculus? A general framework. In C. Zhang, H. W. Guesgen, and W.-K. Yeap, editors, *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, August 9-13, 2004, Proceedings*, Lecture Notes in Computer Science 3157, pages 53–64. Springer, 2004.
- [18] T. Mossakowski. Heterogeneous specification and the heterogeneous tool set. Habilitation thesis, University of Bremen, 2005.
- [19] P. D. Mosses, editor. *CASL Reference Manual*. Lecture Notes in Computer Science 2960. Springer, 2004.
- [20] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL, A Proof Assistant for Higher-Order Logic*. Lecture Notes in Computer Science 2283. Springer, 2002.
- [21] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference (KR-92)*, pages 165–176. Morgan Kaufmann, 1992.
- [22] L. Schröder and T. Mossakowski. HasCASL: Towards integrated specification and development of Haskell programs. In H. Kirchner and C. Reingeissen, editors, *Algebraic Methodology and Software Technology, 2002*, volume 2422 of *Lecture Notes in Computer Science*, pages 99–116. Springer-Verlag, 2002.
- [23] J. G. Stell. Boolean connection algebras: A new approach to the Region-Connection Calculus. *Artificial Intelligence*, 122(1-2):111–136, 2000.
- [24] M. B. Vilain. A system for reasoning about time. In D. L. Waltz, editor, *Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, August 18-20, 1982*, pages 197–201. AAAI Press, 1982.