Some Practical Issues in Building a Hybrid Deductive Geographic Information System with a DL-Component

Michael Wessel
University of Hamburg, Computer Science Department,
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
Email: mwessel@informatik.uni-hamburg.de

Abstract

We report about some preliminary issues from the DFG project "Description Logics and Spatial Reasoning" ("DLS", DFG Grant NE 279/8-1), one of whose goals is to develop a prototypical deductive hybrid Geographic Information System (GIS) with a DL-component. In this paper we discuss the multi-dimensionality of the space of design decisions from a software engineering perspective. In order to support appropriate representation of spatial and thematic aspects and, considering the different aspects of the geographic data, querying the GIS in a *uniform way*, we are developing a *hybrid* representation and reasoning framework, offering support for different description languages (not necessarily being description *logics*). In order to be applicable to a wide range of representation and reasoning tasks, the exploited description languages are not fixed, but exchangeable. The paper sketches our vision of a deductive GIS and we evaluate how standard description logic systems can be of value in this setting. We also introduce a class of *spatio-thematic conjunctive queries* which is useful in our setting here and argue that query satisfiability and containment are decidable.

1 Introduction

As part of the "DLS" project ("Description Logics and Spatial Reasoning", DFG Grant *NE 279/8-1*) we are developing an experimental prototypical deductive *Geographic Information System (GIS).* One of the principal goals of the the project is to exploit "DL technology", even though other techniques might be suitable as well, especially from the "Logic Programming" or database (especially OODB) realm. The plan of this paper is, one the one hand, to introduce the project to the public, to sketch the vision of a deductive GIS and report about the preliminary prototype and its pragmatic solutions (in terms of a progress report), and on the other hand, to explore the *multi-dimensionality of the space of design decisions* of this endeavor and to evaluate how standard description logic systems can be of value in this setting. In order to exploit the DL system RACER ([4]) for the system's reasoning tasks, we are developing a software framework which will also be suitable for similar hybrid representation and reasoning tasks; e.g. the representation of large amounts of semi-structured data as will be found in forthcoming semantic web applications ([2]).

We will show that our system will have to handle large amounts of heterogeneous data, in principal *spatial* and *thematic* data. In order to support appropriate representation of (and reasoning with) the various different aspects of the data in a uniform way, it will turn out that we need an additional layer of software on top of, or in combination with, a DL ABox (please refer to [1] in case any of these terms is not familiar). We will call this layer a *substrate layer*; in combination with a RACER ABox we call it a *RACER substrate*.

In our GIS application the data base consists of a digital vector map of the city of Hamburg. This map can be seen as semi-structured collection of instances of spatial data types, e.g. polygons, lines, symbols, points, and aggregates of these. We will call these objects *map objects* or *geographic objects*. Map objects are annotated with *thematic information*. For example, in case of a map object of type *lake*, which is a polygon, we have

Thematic aspects: its name as given in the map, its water quality, the amount of water contained within it, a reference to its owner (if it is owned at all), etc., as well as

Spatial aspects: its concrete geometric description, e.g. the polygon of the lake, uniquely representing its geometric attributes, as well as probably *additional explicitly stored qualitative spatial relationships*. Since these can always be computed from the geometry of the objects, these explicitly stored edges might be seen as an index (a special "materialized view"), supporting fast navigation through the data. However, in combination with this qualitative index, we also use a true spatial index, similarly to the ones found in conventional GIS (bucket and R-tree structures).

The objects have multi-modal / hybrid flavor. We use the following terminology: a *thematic concept* refers only to thematic aspects, similarly a *spatial concept* solely to spatial aspects, whereas a *spatio-thematic concept* refers to both. We also talk of *thematic*, *spatial* and *spatio-thematic queries*. A strict separation might be difficult sometimes.

In our software framework reasoning works on so-called *substrates*. A substrate offers uniform protocols and reasoning services in a flavor similar to a DL system. As it turns out, a substrate is a quite general notion:

Definition 1 A *substrate* is an edge- and vertex-labeled directed graph (V, E, L_V, L_E), with V being the set of *substrate nodes*, and E being a set of *substrate edges*. The vertex labeling function $L_V : V \to \mathcal{L}_\mathcal{V}$ maps vertices to descriptions in an appropriate vertex description language $\mathcal{L}_\mathcal{V}$, and likewise for $L_E : E \to \mathcal{L}_\mathcal{E}$, where $\mathcal{L}_\mathcal{E}$ is an edge description language. ■

The languages $\mathcal{L}_\mathcal{V}$ and $\mathcal{L}_\mathcal{E}$ are not fixed and can be seen as subsets of predicate logic (in appropriate syntax). The framework provides software infrastructure for various representation and reasoning tasks, including query processing in a uniform way (see below), in forms of generic protocols and classes. We make heavy use of inheritance to derive specializations of substrates and/or description languages.

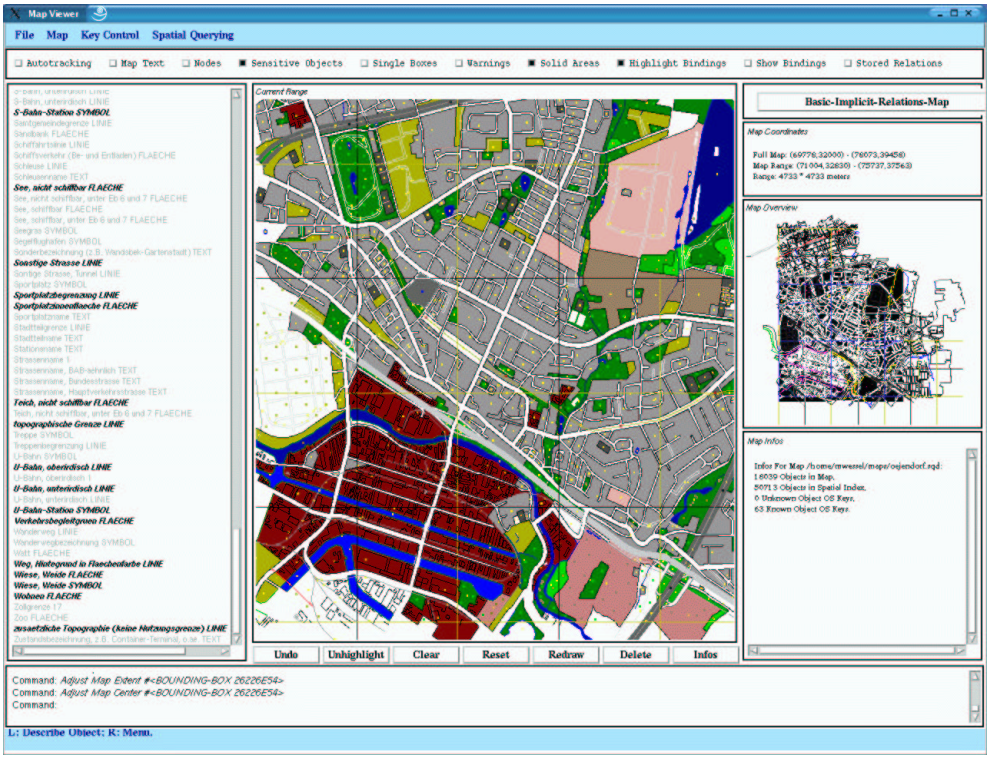


Figure 1: Screen Shot of the "Map Viewer" Component of the GIS Prototype

In order to represent the **spatial aspects** of a geographic map, we use inheritance to get a *geometric substrate.* In this specialized substrate, the nodes would be instances of spatial data types (polygons and the like). Additionally, a *spatial index* would be used to support spatial selection operations. **The languages $\mathcal{L}_{\mathcal{V}}$ and $\mathcal{L}_{\mathcal{E}}$ would be (variable free) ground terms; in fact, the geometric substrate is just a relational structure; neither disjunctions, variables nor complex descriptions are present.** It should be noted that a DL system (equipped with ABox support) with an appropriate *spatial concrete domain* could also be used as a representation device for the concrete geometry of the map, but none of the DL reasoners we know of offers yet the ability to define new concrete domains, especially not the ones we would find useful in this scenario. Despite this we believe that the performance in our application scenario would not be very good, since handling of a spatial concrete domain would need spatial index structures and the like, and it is unlikely that a generic DL system will offer these services sufficiently performant.

Alternatively, we could decide not to represent the geometry of the map at all, but just exhaustively represent certain selected *qualitative spatial aspects* of the map, using a predefined qualitative spatial description vocabulary. In this case a DL ABox would suffice. But still it would be a pity to throw the geometry of the map away. Nevertheless we performed such an experiment with RACER since we also wanted to evaluate how good a system like RACER performs on the resulting very large ABoxes, containing only very simple relational descriptions (see below).

Without doubt, a RACER ABox will be fine to represent the **thematic aspects** of the geographic map. An ABox might also be *seen* as a special substrate – $\mathcal{L}_\mathcal{E}$ would be just a set of role symbols resp. conjunctions build from them, and similarly for the nodes. In contrast to a geometric substrate, $\mathcal{L}_\mathcal{V}$ would be complex formulas, namely concept expressions from a DL. Ideally, in order to get a *map substrate* as the representation medium for the spatial DB, we would like to derive a *common subclass* (using multi-inheritance) of the classes *RACER ABox* and *geometric substrate* and use appropriate *products or combinations* of the thematic and spatial description languages $\mathcal{L}_{\mathcal{V}_{thematic}}$ and $\mathcal{L}_{\mathcal{V}_{geometric}}$, $\mathcal{L}_{\mathcal{E}_{thematic}}$ and $\mathcal{L}_{\mathcal{E}_{geometric}}$ and even allow for interaction between them, but unfortunately RACER is not extendible in this way, and the concrete domain interface is also not extendible. We can therefore either start implementing our own spatio-thematic DL-reasoner, or try to get a "*RACER substrate*" by writing a "wrapper class" and use *association* instead of inheritance:

Definition 2 A *RACER substrate* is a triple $((V, E, L_V, L_E), \mathfrak{A}, *)$, where (V, E, L_V, L_E) is a substrate, and $\mathfrak{A}$ is a RACER ABox. "$*$" is a partial function such that $* : V \rightarrow individuals(\mathfrak{A})$ injectively associates *some* substrate nodes with ABox individuals. If $a \in V$ we write $a^* =_{def} *(a)$ to refer to its associated ABox individual (if defined), and given $a^* \in \mathsf{individuals}(\mathfrak{A})$, we use $a \in V$ to refer to its associated substrate node $a = *^{-1}(a^*)$ (if defined). ■

Finally, a further specialization called *map substrate*, which is basically a RACER substrate $((V, E, L_V, L_E), \mathfrak{A}, *)$ with (V, E, L_V, L_E) being a geometric substrate will be used as the representation medium for the "spatial DB" of the GIS application. Note that there might be substrate nodes without associated ABox individuals, as well as ABox individuals without associated substrate nodes. A geographic object can be seen as a tuple of a spatial object (the substrate node) and a ABox individual.

For what follows, we make the following basic assumptions. Considering the geometric substrate which is just a relational structure, we exploit a *spatial closed domain assumption:* all spatial objects are *explicitly known* in this structure and coincide with V, as well as all their geometric properties from which we can derive any qualitative spatial description (probably reflected in E and L_E). Considering the *logical theory* of this structure, it is clear that *its theory is complete* - we therefore also assume a *spatial closed world assumption.* Please note that this is a consequence of the closed domain assumption and the assumptions that the languages $\mathcal{L}_\mathcal{V}$ and $\mathcal{L}_\mathcal{E}$ are just variable free ground terms; i.e. that the geometric substrate is just a relational structure. However, these assumptions are *not* hard-wired into the software framework - it all depends on the languages $\mathcal{L}_\mathcal{V}$ and $\mathcal{L}_\mathcal{E}$. In contrast to an ABox, we neither have disjunctive information nor can we assert the pure existence of spatial objects in terms of an *existential quantor* "$\exists$" within $\mathcal{L}_\mathcal{V}$.

Within this framework, we foresee the following components of the deductive GIS:

The *extensional component* $\mathcal{E}$, which has been described with the notion of a "spatial DB" or *map substrate* above (and the operations it provides). Recall that a closed domain assumption is exploited for the spatial part, but not for the ABox part.

The *intensional component* $\mathcal{I}$, offering intensional reasoning capabilities. This component should offer the ability to model *ontologies*, in the flavor of DL TBoxes. The language used to describe the relevant concepts from the application domain will be called a spatio-thematic concept language. The objects in the database can be classified resp. *realized* according to the definitions provided by the ontology. Subsumption relationships between concepts should be computed, resulting in a *subsumption taxonomy*.

The *query component* $\mathcal{Q}$, aimed at extracting relevant objects (tuples of data objects) from $\mathcal{E}$, satisfying certain criteria, posed in terms of a query language. The query language should allow to reference the concepts defined in the intensional component as additional vocabulary. Whereas these queries refer to the extensional component, other types of queries might address the intensional component (e.g. to retrieve the set of subconcepts of "public park", like in DL systems).

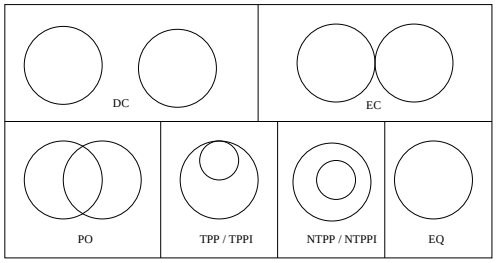
Each of these components might use its own language, and each language might refer to spatial and/or thematic aspects of the geographic objects. We are faced with a multi-dimensional space of design possibilities here. Which kinds of aspects should be addressable in each of the languages, and which spatio-thematic interactions should be permitted? This question has to be evaluated more thoroughly in the future.

It seems that the closed domain assumption makes only sense w.r.t. the components $\mathcal{E}$ and $\mathcal{Q}$. After all one wants to define spatio-thematic concepts in $\mathcal{I}$ *independently of any actual database* resp. domain. Consider the spatio-thematic concept "public park containing a lake". We have no doubt that this concept is satisfiable (i.e., we can easily imagine a world where there is such a park, witnessing the satisfiability of this concept), even if we do not have such an object in our current database, or have never seen such a park. But the same concept would be unsatisfiable over a DB with no such object if we assume a closed world assumption over a fixed DB. In contrast, DL systems use the open domain assumption as well as the open world assumption. All standard DL inference services are based on these assumptions. If we assume the closed world assumption we will run into problems computing satisfiability and subsumption relationships within $\mathcal{I}$. These problems have not been sufficiently solved yet. In the following we will therefore assume that the intensional component $\mathcal{I}$ models only purely thematic aspects of the data, see below.

Considering query answering it turns out that, due to the concreteness of the geometric substrate, query answering w.r.t. the spatial part of the DB will be some kind of spatial model checking, in contrast to query answering in the thematic part. However, in order to compute *query entailment and satisfiability* (see below), we should also resort to the open world assumption, since these are really intensional reasoning tasks.

2 The Data

Basically, we got some digital vector maps from the local government of Hamburg ("Amt für Geoinformation und Vermessungswesen"), as a file in the proprietary "SQD"



a)

∘	DR	PO	EQ	PPI	PP
DR	*	DR PO PPI	DR	DR PO PPI	DR
PO	DR PO PP	*	PO	PO PPI	DR PO PP
EQ	DR	PO	EQ	PPI	PP
PP	DR PO PP	PO PP	PP	PO EQ PP PPI	PP
PPI	DR	DR PO PPI	PPI	PPI	*

b)

Figure 2: a) RCC8 Relations b) RCC5 Composition Table, $DR =_{def} \{EC, DC\}$, $PP =_{def} \{TPP, NTPP\}$ aka "contained in", $PPI =_{def} \{TPPI, NTPPI\}$ aka "contains"

format (from Siemens). The map comprises 18.039 geometric objects, with 5.418 primary objects (the non-primary objects are component objects of the primary ones). Part of this map is shown in Figure 1, showing the "Map Viewer" component of our prototype. Each map object is annotated with a four digit number: the so-called *object key*, describing its thematic aspects. These keys can be looked up in the so-called *object key dictionary* which simply maps these numbers to *concept names*. The dictionary contains concept names without definitions, such as "green area", "meadow", "public park", "lake", "public park containing a lake", etc. A few hundred concepts are present. This is how thematic information is currently handled in commercial GIS.

3 Adding a Simple Ontology

Considering the dictionary, it becomes obvious that no taxonomic structure is modeled, although it is implicitly present in the "intuitive semantics" of some of the concept names: a user querying the system for instances of *green_area* will not be supplied with instances of the sub concepts *meadow* and *public_park*. Thus, in a first step, we model the purely thematic aspects of these concepts in a TBox with inclusion axioms like $meadow \sqsubseteq green\_area$, $public\_park \sqsubseteq green\_area$, etc. This is not to say that some concepts might not also have complex thematic descriptions, but we do not want to go into detail here. Still the resulting TBox is quite simple and even *unfoldable.* Basically, $\mathcal{ALC}$ suffices for some basic modeling of thematic aspects, and RACER even offers $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$.

However, it should be mentioned that some of the concepts from the dictionary in fact have a *spatio-thematic character,* e.g. concepts like "public park containing a lake". In this case we would like to put definitions like

$$public\_park\_containing\_a\_lake =_{def} public\_park \sqcap \exists contains.lake,$$

into the intensional component $\mathcal{I}$ (the ontology) of our system, and we would need a logic that is aware of the specific properties of qualitative spatial relationships (in the following we will solely consider spatio-thematic descriptions referring to qualitative spatial relationships). For example, the contains relationship should be interpreted as a partial strict-order. Thus, independently of the content of the extensional "database component" $\mathcal{E}$, the system should infer that the concept

$$living\_area\_containing\_a\_park\_with\_a\_lake =_{def} \; living\_area \sqcap \exists contains.public\_park\_containing\_a\_lake$$

is actually subsumed by the concept

$$area\_containing\_an\_area =_{def} area \sqcap \exists contains.area,$$

due to the transitivity of the *contains* role, and assuming that $park \sqsubseteq area$, $lake \sqsubseteq area$, $living\_area \sqsubseteq area$, etc. Even though we could express the transitivity of the *contains* relationship within $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$ (so RACER would actually find this subsumption relationship!), most other properties of the qualitative spatial relationships would be lost. However, nothing prevents us from putting definitions like

$$public\_park\_containing\_a\_lake =_{def} public\_park \sqcap \exists contains.lake$$

into a RACER TBox. Surely incompleteness might arise, which means that *w.r.t. the intended semantics of the roles,* the TBox might become inconsistent without being noticed, and subsumption relationships might be missing as well.

Despite these problems with completeness in $\mathcal{I}$ we could still use its definitions (assuming it is unfoldable) as additional *vocabulary within queries;* e.g. to retrieve all instances of *public_park_containing_a_lake*. In this case we have to ensure that the constraints in the ABox are complete, resp. *closed*, in such a way that the explicitly present role membership assertions in the ABox mirror the exact semantics of the spatial relationships. In case of the qualitative RCC relationships (see Figure 2), the resulting ABox would have to take the form of a complete graph, satisfying the appropriate *RCC composition table* as well as some other conditions, like irreflexiveness, disjointness etc., forming a consistent and complete RCC network. Even though it is easy to compute such an ABox from the explicit geometry of a map, the resulting ABox will be very big (see below).

In order to overcome the incompleteness in the intensional component $\mathcal{I}$ if we simply use $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$, we have proposed to use a language called $\mathcal{ALCI}_{\mathcal{RCC}}$ (see [7]) as the DL for a deductive GIS using qualitative spatial RCC relationships. However, other spatial description logics / languages might be suitable as well.

4 Representing the Data - An Experiment With RACER

We have already sketched above that we can simply "mirror" computed qualitative spatial relationships into a RACER ABox and pose standard DL queries (instance retrieval queries). To do so, we use our map substrate, and arrange for automatic mirroring of *RCC8 relationship*. Since every object is in relation with every other object (including itself, note the *EQ* relation), we have $E = V \times V$, and $L_E : E \rightarrow \mathcal{L}_\mathcal{E}$ would be $\mathcal{L}_\mathcal{E} =_{def} \{EQ, DC, EC, PO, TPP, TPPI, NTPP, NTPPI\}$.

Optionally, we can then "close" the generated ABox $\mathfrak{A}$ - for each ABox individual $i \in \mathsf{individuals}(\mathfrak{A})$ and each RCC role R we simply count the number of R-successors in the generated ABox and add a role number restriction $i : (\leq \; R \; n) \sqcap (\geq \; R \; n)$ to $\mathfrak{A}$, effectively constraining the fillers of the role R to the be present ones, where $n =_{def} |\{ j \mid (i,j) : R \in \mathfrak{A} \}|$. This closing of roles is required if we want to have queries involving the "$\forall$"-quantor correctly answered (e.g., retrieve all instances of $living\_area \sqcap \forall EC.\neg industrial\_area$).

Using a smaller map resulting in only 130.321 role membership assertions, we tried to

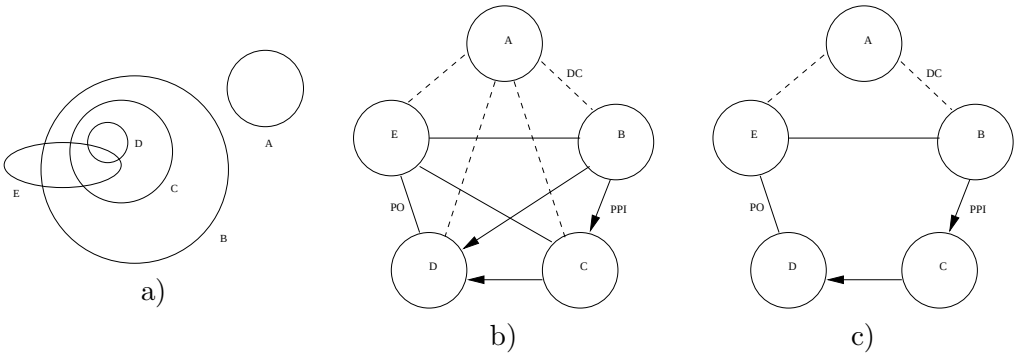


Figure 3: a) Scene, b) corresponding RCC5 net, c) edge-reduced RCC5 net

pose instance retrieval queries to RACER like

$$retrieve\_concept\_instances(public\_park \sqcap (\exists TPPI.lake \sqcup \exists NTPPI.lake)).$$

RACER 1.6 was unable to answer such queries, apparently due to the immense size of the ABoxes: even the simpler query $retrieve\_concept\_instances(public\_park)$ remained unanswered after more than 2 hours. However, RACER 1.7 processes these queries nearly instantaneously, apparently due to additional index structures (personal communication with Ralf Möller, one of the developers of RACER).

Nevertheless the size of the generated ABoxes may become a serious problem if bigger maps are considered. The map depicted in Figure 1 gives rise to over 29 million role membership assertions. By observing that most of the present relations are of type DC (disconnected) and that queries resorting to this spatial relationship do not appear to be very meaningful, we can remove the DC-assertions, resulting in a much smaller ABox.

Alternatively, instead of precomputing and mirroring the RCC relationships into an ABox, the geometry of the objects in the geometric substrate could be used to compute the required spatial relationships on the fly while query processing proceeds. On the one hand, this computation of RCC relationships during query evaluation is expensive. On the other hand, precomputation of the relations needs lots of memory (and initial preprocessing time). However, in the case of RCC5, a clever trade-off can be found. Consider the scene depicted in 3a. Its corresponding RCC5 network is depicted in 3b. By exploiting the properties of the RCC5 relationships (especially $PPI \circ PPI \Rightarrow PPI$, $DR \circ PPI \Rightarrow DR$), it is sufficient to precompute and store only the edges depicted in 3c, from which 3b can uniquely be reconstructed. The representation is sparse, on the one hand, and frees us from expensive runtime-computations at query execution time, on the other hand. But it seems that these "representation and indexing tricks" can only be implemented within the geometric substrate, since the source code of RACER ist not available. In order to avoid the precomputation of these really big ABoxes and to support more expressive queries than RACER currently offers (e.g., conjunctive queries), we therefore decide to use a truly hybrid representation and query language which we will describe next. We also sketch how a (restricted form of) reasoning could work in such a hybrid representation system.

5 Hybrid Conjunctive Queries

Suppose the company "Agricultural Products" (AP) is looking for new sub-contractors: farmers that can supply potatoes, corn, and wheat, such that their farm lands are located directly adjacent to the company's factory. We could come up with a query like this one:

$$
\begin{aligned}
&query(?farmer*, ?factory\_area, ?cropland, ?potatoland, ?wheatland) \leftarrow \\
&\quad farmer(?farmer^*), has\_factory(AP^*, ?factory^*), \\
&\quad owns\_farmland(?farmer^*, ?cropland^*), (\forall grows.crop \sqcap \forall harvest.high)(?cropland^*), \\
&\quad owns\_farmland(?farmer^*, ?potatoeland^*), (\forall grows.potatoes \ldots)(?potatoeland^*), \\
&\quad owns\_farmland(?farmer^*, ?wheatland^*), (\forall grows.wheat \ldots)(?wheatland^*), \\
&\quad (TPPI(?factory\_area, ?factory) \vee NTPPI(?factory\_area, ?factory)), \\
&\quad EC(?factory\_area, ?cropland), EC(?factory\_area, ?potatoeland), \\
&\quad EC(?factory\_area, ?wheatland)
\end{aligned}
$$

$AP*$ is a named ABox individual, representing the company. The items prefixed with "?" are variables; $?cropland^*$ is a variable ranging over ABox individuals, and $?cropland$ is automatically bound to its corresponding substrate node (and vice versa). The system will return a set of 5-tuples as answer to this query.

Actually, we defined the hybrid queries for *racer substrates*; since a *map substrate* is a specialization of a racer substrate, the queries work on map substrates as well, but use special query vocabulary (e.g., the RCC predicates are only "understood" by the geometric substrate). Note that the vocabulary used within the hybrid queries has to be "matched" against the appropriate vocabulary with is used within the substrate ($\mathcal{L}_\mathcal{V}$, $\mathcal{L}_\mathcal{E}$, as well as the ABox language). We define the hybrid conjunctive queries as follows:

Definition 3 (Hybrid Conjunctive Queries) Let $((V, E, L_V, L_E), \mathfrak{A}, *)$ be a racer substrate. Let $\mathcal{V}_\mathcal{S} =_{def} \{x_1, \ldots, x_n\}$ be a set of *substrate variable names*, and $\mathcal{V}_\mathcal{T} =_{def} \{x_1^*, \ldots, x_n^*\}$ be a set of *ABox variable names*, such that the sets $\mathcal{V}_\mathcal{S}$, $\mathcal{V}_\mathcal{T}$, V and $\mathsf{individuals}(\mathfrak{A})$ are pairwise disjoint. Whenever we use x_i we mean *the* x_i from $\mathcal{V}_\mathcal{S}$, and likewise if we use x_j^*. If we say $u \in \mathcal{V}_\mathcal{S}$, we mean any of the variables from $\mathcal{V}_\mathcal{S}$ (analogously for $\mathcal{V}_\mathcal{T}$).

A *substrate vector* (n-tuple) $(u_1, \ldots, u_n)$ with $u_i \in \mathcal{V}_\mathcal{S} \cup V$ is denoted by $\vec{U}$; likewise an *ABox vector* $(v_1^*, \ldots, v_m^*)$ with $v_j^* \in \mathcal{V}_\mathcal{T} \cup \mathsf{individuals}(\mathfrak{A})$ by $\vec{V^*}$. Note that vectors might refer to specific substrate nodes as well as to ABox individuals. Each vector is either a substrate vector or an ABox vector; there are no "mixed vectors". If $v \in V$, then $v^* \in \mathsf{individuals}(\mathfrak{A})$ denotes its corresponding ABox individual (if it exists); and likewise, if $v^* \in \mathsf{individuals}(\mathfrak{A})$, then v denotes its corresponding substrate node (if it exists). We refer to the set $\{u_1, \ldots, u_n\} \cap (\mathcal{V}_\mathcal{S} \cup \mathcal{V}_\mathcal{T})$ as $\mathsf{vars}(\vec{U})$. Intuitively, variables named x_i^* are bound to ABox individuals, whereas variables without asterix, e.g x_i, are bound to *substrate individuals / nodes*.

A *substrate query conjunct* $SQC(\vec{U})$ is a sentence of a *substrate query language* $\mathcal{SQL}$ to be specified by the user, where $\vec{U}$ is a substrate vector. An *ABox query*

conjunct $AQC(\vec{V^*})$ is a sentence of an *ABox query language* $\mathcal{AQL}$ to be specified by the user and/or exploited $\mathcal{DL}$ reasoner, where $\vec{V}*$ is an ABox vector.

Then, a *hybrid conjunctive query* is an expression of the form

$$\begin{aligned} query(\vec{U}, \vec{V^*}) \leftarrow \quad & SQC_1(\vec{U_1}), SQC_2(\vec{U_2}) \ldots SQC_n(\vec{U_n}), \\ & AQC_1(\vec{V_1^*}), AQC_2(\vec{V_2^*}) \ldots AQC_m(\vec{V_m^*}), \end{aligned}$$

where the $SQC_i \in \mathcal{SQL}$ are substrate query conjuncts, and the $AQC_j \in \mathcal{AQL}$ are ABox query conjuncts. The antecedent of the implication is called the body. Additionally, $\mathsf{vars}(\vec{U}) \subseteq \bigcup_{i \in 1 \ldots n} \mathsf{vars}(\vec{U_i})$ and $\mathsf{vars}(\vec{V^*}) \subseteq \bigcup_{i \in 1 \ldots m} \mathsf{vars}(\vec{V_i^*})$ (i.e., the *query* predicate does not mention variables that don't appear in the body of the query). ■

Considering $\mathcal{SQL}$ and $\mathcal{AQL}$ in the example given above, $AQC(\vec{V^*})$ is either a *role query conjunct* of the form $R(x_i^*, x_j^*)$ or a *concept query conjunct* of the form $C(x_i^*)$. Obviously, one can also think of other expressions (e.g. $\neg R(x_i^*, x_j^*)$), but this is not the point here. W.r.t. the $\mathcal{SQL}$, we only used (disjunctions of) binary RCC8 predicates. In fact, a disjunction of binary RCC8 predicates can be seen as an RCC8 predicate as well; this makes no problems (see below). We can also process arbitrary boolean formulas build from RCC8 predicates. Due to the so-called *JEPD property* of the RCC relations, a negation of the form $\neg R(x_1, x_2)$ is again equivalent to a disjunction $\bigvee_{S \in RCC\_Relationships \setminus \{R\}} S$. Basically, the semantics of the queries is very simple:

Definition 4 (Semantics of Queries) Let $DB =_{def} ((V, E, L_V, L_E), \mathfrak{A}, *)$ be a map substrate, and $query(\vec{U}, \vec{V^*})$ be given, $\vec{U} = (u_1, \ldots, u_n)$, $\vec{V^*} = (v_1^*, \ldots, v_m^*)$. Then, the set of query results (answer tuples) is

$$\begin{aligned} \{ (\alpha(u_1), \ldots, \alpha(u_n), \alpha(v_1^*), \ldots, \alpha(v_m^*)) \mid & \\ & \text{there is some injective } \alpha \text{ such that for all } i \in 1 \ldots n, j \in 1 \ldots m : \\ & (V, E, L_V, L_E) \models SQ_i(\alpha(\vec{U_i})), \mathfrak{A} \models AQ_j(\alpha(\vec{V_j^*})) \}, \end{aligned}$$

where $\alpha(a)$ needs to satisfy: $\alpha(a) \in V$ iff $a \in \mathcal{V_S}$, $\alpha(a) \in \mathsf{individuals}(\mathfrak{A})$ iff $a \in \mathcal{V_T}$, and $\alpha(a) = a$ iff $a \in \mathsf{individuals}(\mathfrak{A}) \cup V$. That is, we are using the *active domain assumption;* variables range over explicitly mentioned objects in $V \cup \mathsf{individuals}(\mathfrak{A})$, and (ABox) individuals / nodes are mapped to themselves. Note that α is injective - therefore, the *unique name assumption for variables* is assumed (two different variables can never be bound against the same object).

The meaning of $\mathfrak{A} \models AQC(\alpha(\vec{V^*}))$ is the standard one; $\mathfrak{A} \models AQC(\alpha(\vec{V^*}))$ iff for all models $\mathcal{I}$, whenever $\mathcal{I} \models \mathfrak{A}$, also $\mathcal{I} \models AQC(\alpha(\vec{V^*}))$, that is, $\alpha(\vec{V_j^*})^{\mathcal{I}} \in AQC^{\mathcal{I}}$. Since AQC is a sentence from a unknown $\mathcal{AQL}$, we only require that the test is decidable; it will be delegated to the $\mathcal{DL}$ reasoner, e.g. RACER. The same applies to $(V, E, L_V, L_E) \models SQC(\alpha(\vec{U}))$. For example, simple model checking might be enough in some circumstances (e.g., if the substrate uses only ground terms). In the framework, it is completely left open for the application to define satisfaction in $\mathcal{I}$, i.e. when $\mathcal{I} \models SQC(\alpha(\vec{U^*}))$ holds. All what matters is that the test should be decidable. ■

Suppose $\mathcal{SQL}$ is given by all boolean formulas which can be build from the set of RCC8 base relations over a set of variables, like in the example query above. We can then state that

Proposition 1 Satisfiability of *spatio-thematic* queries is decidable.

The proof is more or less trivial, since there is (yet) no interaction between the two languages $\mathcal{AQL}$ and $\mathcal{SQL}$. First we collect all substrate query conjuncts SQC_i and check for their satisfiability. In case $\mathcal{SQL}$ comprises only descriptions of RCC networks, we simply construct the RCC network as specified by the SQC_i's and check for its consistency. In case the SQC_i's refers to individuals, we can of course only check satisfiability w.r.t. a given substrate; in this case we simple look-up the relationships. For $\mathcal{AQL}$, we just build an ABox, one individual for each variable; in case an AQC refers to an ABox individual, we just copy the relevant parts from the referred ABox (its connected component). In case we have AQC_i's using disjunctions, we perform case analysis over the disjuncts (resulting in eventually exponentially many ABoxes). In case we have an AQC_i like $\neg R(x^*, y^*)$ we add $\{y^* : \boxed{y^*}, x^* : \forall R.\neg \boxed{y^*}\}$ to the ABox to check, where $\boxed{y^*}$ is a new concept name. Finally, we check each of the constructed ABoxes for satisfiability. The only problems that might arise come from the possibility to refer to individuals from $\mathfrak{A}$, but they can be fixed (the only problematic constructs might be number restrictions). ■

Proposition 2 Query containment of *spatio-thematic* queries is decidable.

For example, the system will deduce that the query

$$\begin{aligned} &query(?germany, ?city, ?sea) \leftarrow \\ &\quad germany(?germany^*), federal\_division(?division^*), german\_city(?city^*), \\ &\quad (baltic\_see \sqcup north\_sea)(?sea^*), \\ &\quad PPI(?germany, ?division), PPI(?division, ?city), DC(?division, ?sea) \end{aligned}$$

entails the query

$$\begin{aligned} &query(?country, ?city, ?ocean) \leftarrow \\ &\quad country(?country^*), city(?city^*), ocean(?ocean^*), \\ &\quad DC(?ocean, ?city), PPI(?country, ?city) \end{aligned}$$

To see this, consider the textual substitution $?country \leftarrow ?germany, ?ocean \leftarrow ?sea$ applied to the latter query. Assume that $germany \models country$, $(baltic\_see \sqcup north\_sea) \models ocean$ due to a reasonable ontology, and $(PPI(?division, ?city) \wedge DC(?division, ?sea)) \models DC(?city, ?sea)$, $DC(?city, ?sea) \models DC(?sea, ?city)$, $(PPI(?germany, ?division) \wedge PPI(?division, ?city)) \models PPI(?germany, ?city)$ simply by the semantics of the RCC relationships.

Proof Sketch: Given two queries $\mathcal{A}$ and $\mathcal{B}$, we first check whether their *query* head predicates have the same arity; if so, we rename the variables referenced in $\mathcal{B}$'s *query*-predicate in such a way to match $\mathcal{A}$'s *query*-predicate. Individuals cannot be renamed; if there is a naming clash, the answer to the query containment problem is already "No". Otherwise, check for the un-satisfiability of $\mathcal{A} \wedge$ "$\neg\mathcal{B}''$. Collect all SQC_i's from $\mathcal{A}$, and all SQC_j's from $\mathcal{B}$. Then test, for each SQC_j from $\mathcal{B}$, if adding its negation $\neg SQC_j$ to $\mathcal{A}$'s SQC's yields an unsatisfiable RCC network. If not, return "No". Otherwise proceed by applying the analogue test (adding $\mathcal{B}$'s negated AQC_j's to the AQC's of $\mathcal{A}$) and check for ABox satisfiability using the translation sketched

above. If any of the ABoxes is satisfiable, return "No", otherwise "Yes". More detailed proofs can be found in a forthcoming report. ■

As a final example, suppose that $A =_{def} query(?x^*) \leftarrow (\exists R.C)(?x^*)$, and $B =_{def} query(?x^*) \leftarrow R(?x^*, ?y^*), C(?y^*)$. Please note that, due to the given semantics, $A \not\models B$, but $B \models A$.

6 Conclusion

We have reported about work carried out in the context of the "DLS" project and have argued that a lot of interesting things can be done in a setting as ours with a DL system, but it is not trivial to get a *working* hybrid system. DLs are no silver bullet for deductive information systems. In order to get a working hybrid information system, appropriate combinations of techniques will be needed, control over application-dependent index structures (see also [6]), appropriate classes of queries and query answering strategies, etc. The space of design decisions is very large. We claim that applications in the context of the future semantic web, will need similar hybrid representation and reasoning techniques in order to perform sufficiently good. Eventually, we plan to augment the system with non-recursive DATALOG clauses (see also [3], [5]), either in the extensional component, or solely in the query component.

References

[1] F. Baader. Description logic terminology. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 485–495. Cambridge University Press, 2003.

[2] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003. To appear.

[3] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[4] Volker Haarslev and Ralf Möller. Description of the RACER system and its applications. In *Description Logics*, 2001.

[5] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conference on Artificial Intelligence*, pages 323–327, 1996.

[6] Albrecht Schmiedel. Semantic indexing based on description logics. In *Knowledge Representation Meets Databases*, 1994.

[7] M. Wessel. On spatial reasoning with description logics - position paper. In Sergio Tessaris Ian Horrocks, editor, *Proceedings of the International Workshop on Description Logics 2002 (DL2002)*, number 53 in CEUR-WS, pages 156–163, Toulouse, France, April 19–21 2002. RWTH Aachen. Proceedings online available from `http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-53/`.