

Spatio-temporal schema integration with validation: a practical approach. ^{*}

A. Sotnykova¹, N. Cullot², and C. Vangenot¹

¹ École Polytechnique Fédérale de Lausanne, Database Laboratory,
CH-1015 Lausanne, Switzerland,

{Anastasiya.Sotnykova, Christelle.Vangenot}@epfl.ch

² Université de Bourgogne, Laboratoire LE2I, F-21078 Dijon Cedex,
Nadine.Cullot@u-bourgogne.fr

Abstract. We propose to enhance a schema integration process with a validation phase employing logic-based data models. In our methodology, we validate the source schemas against the data model; the inter-schema mappings are validated against the semantics of the data model and the syntax of the correspondence language. In this paper, we focus on how to employ a reasoning engine to validate spatio-temporal schemas and describe where the reasoning engine is plugged into our integration methodology. The validation phase distinguishes our integration methodology from other approaches. We shift the emphasis on automation from the a priori discovery to the a posteriori checking of the inter-schema mappings. By doing so, we take advantage of the expressive power of the common data model in the source schema description and inter-schema mapping definition.

1 Integration Approach

Database integration has been and continues to be the focus of many research efforts. The integration task involving the spatial and temporal domains becomes even more complex bearing a weak number of formal approaches reported in the literature. In this paper, we describe a hybrid approach exploiting advantages of two formalisms: a spatio-temporal conceptual model and an expressive description logic. The relationships between database conceptual models, in particular the Entity-Relationship (ER) model, and Description Logics (DLs) are addressed in [1, 2], where it is shown that DLs are powerful enough to capture the domain semantics represented by ER based models. In this paper, we present a translation of MADS conceptual data model into a DL formalism. MADS [3] is an object+relationship conceptual data model featuring structural, spatial, temporal, and multiple perceptions dimensions. In our previous paper [4], we presented our integration methodology where MADS is the common data model

^{*} This work is supported, in the framework of the EPFL Center for Global Computing, by the Swiss National Funding Agency OFES as part of the European projects KnowledgeWeb (FP6-507482) and DIP (FP6-507483).

for the source and resulting integrated spatio-temporal database schemas. The methodology consists of four main phases where 1) source schemas are translated into MADS data model; 2) the inter-schema mappings are expressed in the MADS correspondence language; 3) based on the set of inter-schema mappings and integrity constraints, the structural solutions for related parts of the schemas are proposed to the designer; and 4) finally, the integrated schema is composed. Theoretical concepts underlying these phases include syntax and semantics for the inter-schema mapping languages and a predefined set of structural patterns for different set relationships between source schema populations. The integration process is not automatic: the source schemas are translated into the MADS model by the schema designer, the inter-schema mappings and the integrity constraints are asserted manually. Thus, our methodology requires a verification mechanism for the totality of the expressions stated manually.

We now propose an enhanced integration approach where every manual phase is supported by a validation operation ensuring data model and language compliance. For the phase 1), we aim to verify the validity of the expressions of the source schemas in the MADS model. The MADS data model itself and the source schemas with integrity constraints are translated into a DL based language and the satisfiability of the resulting translated DL model is validated. In phase 2), inter-schema mappings are added to the DL models in terms of elements of the source schemas and mapping operators [5] thus ensuring that the mappings are data model compliant and there are no contradictory mappings. Several possible ways to construct the integrated schema are then presented to the schema designer. During the phase 3) these putative structural solutions are checked for compatibility with the integrity constraints imposed on the schema elements that are to be integrated. A successful satisfiability check signifies that the populations of the related schema elements can be merged; in case of failure, only the structural solutions that model the populations separately are applicable. The final decision on the choice of the structural solutions is up to the designer and/or a domain expert. By composing our integration method of the phases described above, we assure that the schema designer has sufficient knowledge about the compatibility of the source schemas and possible structural solutions for the final integrated schema.

The examples that we will use throughout the paper are two database schemas (Fig. 1.1 and 1.2) developed by two tourist offices describing the same geographical area, the city of Paris¹. Schema T_1 models a cadastral division of the city, where the city is decomposed in several city boroughs and specifies where some of the urban services available for tourists, e.g., public transport stops available in each city borough. The object type `TouristPlace` is a spatio-temporal object type. Its spatial extension is of type `simple area` and its temporal one of type `interval`. The subtypes of `TouristPlace` inherit the spatial and temporal properties of their ancestor. The schema T_2 has a different focus. Tourist attractions that compose the population of the `TouristSite` object type are those accessible by public transport. The `TouristSite` object type has a spatial extension of type

¹ complete figures can be found in [4].

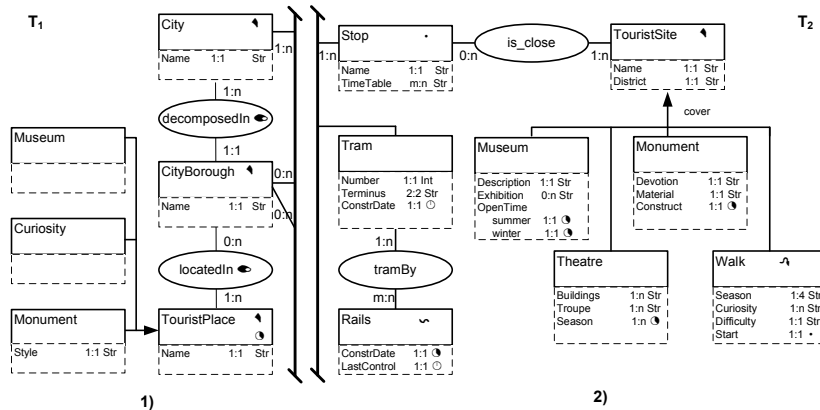


Fig. 1. Fragments of schema T_1 and schema T_2 both modeling a city for tourists.

simple area and Stop of type point. Given those properties and the cardinality of the `is_close` relationship, the population of `TouristSite` is limited to the tourist attractions located not further than at a certain distance from a transport stop. The subtypes of `TouristSite` inherit its spatiality except for the `Walk` subtype whose spatial extension is redefined to the oriented line type.

2 Validation: Practice

Theoretically, we were looking for the description logic(s) best suited for modeling spatial and temporal data. As discussed in [4], to provide validation to the theoretical foundations of our integration approach, we exploited the $\mathcal{ALCRP}(\mathcal{S}_2 \oplus \mathcal{T})$ expressive power to describe the source spatio-temporal schemas and interschema mappings. The $\mathcal{ALCRP}(\mathcal{D})$ DL proposed in [6], extends $\mathcal{ALC}(\mathcal{D})$ to build complex roles based on a role-forming predicate operator [7]. In particular, an appropriate concrete domain \mathcal{S}_2 is defined for polygons using \mathcal{RCC}_8 relations as basic predicates of concrete domain. For temporal aspects, the concrete domain \mathcal{T} is a set of time intervals and Allen relationships are used as basic predicates describing the relationships between intervals.

In practice, as we want to use an existing reasoner, we are limited by the $\mathcal{SHOIN}(\mathcal{D})$ [2] description logic which is the base for the OWL ontology language. For this logic we can use Protégé [8] graphical tool, and RACER [9] reasoning system. The $\mathcal{SHOIN}(\mathcal{D})$ logic does not treat either spatial, or temporal concrete domains. Next, we will show how we implement the MADS model in this logic aiming at emulating spatial and temporal semantics for describing spatio-temporal information.

2.1 Data Model Definition in OWL

In this section we present the translation of the structural, spatial, and temporal dimensions of the MADS data model in the OWL (OWL-DL) language. Due

to the space limitation we will not detail MADS constrained relationships and multiple perceptions. Below, we refer to the MADS model expressed in OWL as MADS-OWL.

Structural Dimension. An OWL model is composed of classes, properties, and instances. Restrictions on properties and classes are the integral part of an OWL model. In MADS, a schema is composed of object and relationship types, and attributes.

Object types definition. We define a MADS object type in OWL by `<owl:Class>` constructor. A MADS object type can have several subtypes and supertypes, allowing for the multi-inheritance which is an important feature of the MADS model. In OWL, classes may also have several subtypes and supertypes. OWL classes by default, share their instances; in MADS, object types have explicit extensions. To restrict an OWL class, for example, the `TouristPlace` class to have only predefined set of instances we use the `<owl:unionOf>` axiom over its subclasses:

```
<owl:Class rdf:ID="TouristPlace">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Museum"/>
        <owl:Class rdf:ID="Curiosity"/>
        <owl:Class rdf:ID="Monument"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

With the above described OWL features, we are able define object types, the generalization/specialization links with disjoint and cover axioms, and multi-inheritance as in the MADS model.

Relationship types definition. A MADS relationship is a link between two object types. An OWL property is a binary relationship that connects two OWL concepts, called domain and range of the property. Linked concepts can be either classes, or a class and a datatype thus defining two types of properties: object properties linking individuals to individuals; and datatype properties linking individuals to data values. Unless otherwise constrained, OWL properties are multivalued. To represent MADS relationships we use OWL object properties, for example, MADS relationship `tramBy` from Fig. 1.2 is defined as follows:

```
<owl:ObjectProperty rdf:ID="tramBy">
  <rdfs:domain rdf:ID="Tram"/>
  <rdfs:range rdf:ID="Rails"/>
</owl:ObjectProperty>
```

Like for the object types, in MADS we can define the generalization/specialization link between relationship types. OWL also support hierarchies of properties which can be specified with `<owl:subPropertyOf>` axiom.

Attributes definition. MADS attributes can either be complex or simple, mono-valued or multivalued, mandatory or optional. A MADS attribute is translated

to OWL by one of the two property types depending on the type of the attribute. To represent MADS simple attributes, like the `Description` attribute of the object type `Museum` from Fig. 1.2, we create an OWL property `Description`, with the domain restricted to the class `Museum`, and the range of the datatype `string`:

```
<owl:DatatypeProperty rdf:ID="description">
  <rdfs:domain rdf:ID="Museum"/>
  <rdfs:range rdf:ID="http://www.w3.org/2001/XMLSchema string"/>
</owl:DatatypeProperty>
```

Defining a complex attribute in OWL requires additional classes to be created. For instance, `Museum` has a complex attribute `OpenTime` with two component attributes `summer` and `winter`, Fig. 1.2. To translate MADS attribute `OpenTime` to OWL property `OpenTime`, we first create a class `MuseumOpenTime`, and then two object properties `summer` and `winter` with the domain restricted to this class. Then the object property `OpenTime` with domain `Museum` and range `MuseumOpenTime` can be created. Each value of the property `openTime` would be an instance of a class `MuseumOpenTime` with two values for `winter` and `summer` properties. We summarize all the structural elements that we use in two modeling approaches - MADS and OWL, in Table 1.

Table 1. Structural Dimension - MADS vs OWL

MADS concepts	OWL constructors / restrictions
Object type	<code>owl:Class</code>
IsA link	<code>owl:SubClassOf</code>
Covering axioms: cover, disjoint	<code>owl:UnionOf</code> , <code>owl:DisjointWith</code>
Relationship type	<code>owl:ObjectProperty</code> with defined <code>owl:range</code> <code>owl:domain</code> , and <code>owl:inversePropertyOf</code>
IsA link	<code>owl:subPropertyOf</code>
Role cardinalities	<code>owl:minCardinality</code> , <code>owl:maxCardinality</code> <code>owl:Cardinality</code>
Object simple attribute	<code>owl:datatypeProperty</code>
Object complex attribute	<code>owl:objectProperty</code>
Identifier attribute	<code>owl:functionalProperty</code>

Spatial Dimension. The MADS spatial dimension is defined by the hierarchy of spatial abstract data types (SADT) and the set of topological relationships. The SADTs are associated to object types, and attributes to add specific spatial features whereas topological features are added to relationship types to convey spatial constraints. Spatial dimension in MADS is orthogonal to the structural dimension. Similarly, modeling in OWL we aim at defining the spatial dimension in a way it could be freely added or removed as additional feature to structural elements. We define a hierarchy of OWL classes together with the restrictions

on the topological properties that may hold between spatial classes. The user spatial classes are defined as subclasses of the members of the spatial hierarchy according to the chosen spatial features. With the `<owl:subClassOf>` constructor, the restrictions of the superclass are inherited by its subclasses, preserving its spatial behavior and allowing controls on the topological relationships between spatial classes.

Contrary to as how the spatial features are defined in MADS, in MADS-OWL we only introduce the `hasGeometry` property as a synthetic property without adding all the semantics of its MADS counterpart. We make it intrinsic or defining property of spatial classes. We define the root spatial class `Geo`, and restrict the domain of the property `hasGeometry` to it; the range of this property we restrict to the union of classes `GSimple` and `GComplex` which are two disjoint classes populated with spatial instances. Then, we define the complete OWL spatial hierarchy extending `Geo` class, Fig. 2; and precisising the values for the `hasGeometry` property for each spatial subclass.

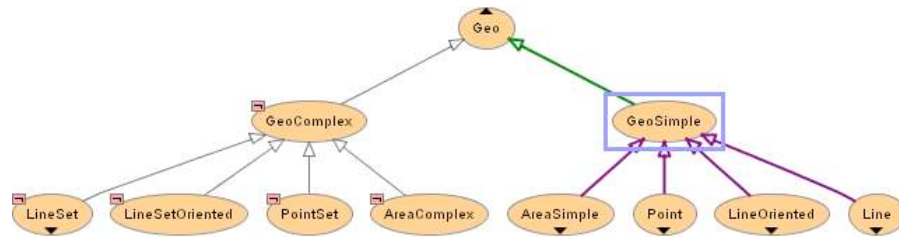


Fig. 2. OWL: the hierarchy of spatial classes.

Temporal Dimension. Temporal dimension is defined in a similar way to the spatial dimension. In compliance to the MADS temporal abstract type hierarchy we define the temporal MADS-OWL class hierarchy as shown in Fig. 3. The intrinsic property for temporal types is the `hasTime` property, meaning that the necessary and sufficient condition for a class to be classified as a temporal one, is the existence of the `hasTime` property. For the root temporal class `Time`, this property must exist, and the value of this property is restricted to one of the values from the `TSimple` or `TComplex` classes.

For the subclasses, we define additional axioms on the values of the property `hasTime`. For example, let us consider in details the temporal type `Interval`, its representation is the Protégé tool shown in Fig. 3. All the restrictions concern the `hasTime` property. From the root class `Time`, `Interval` inherits the most general restriction on the `hasTime` property. Then, from the `TimeSimple` class a more precise restriction is inherited. This restriction is compliant to the previous, more general one, and it restricts the range of the `hasTime` property only to the instances of the `TSimple` class. Finally, in the leaf class `Interval`, the restriction

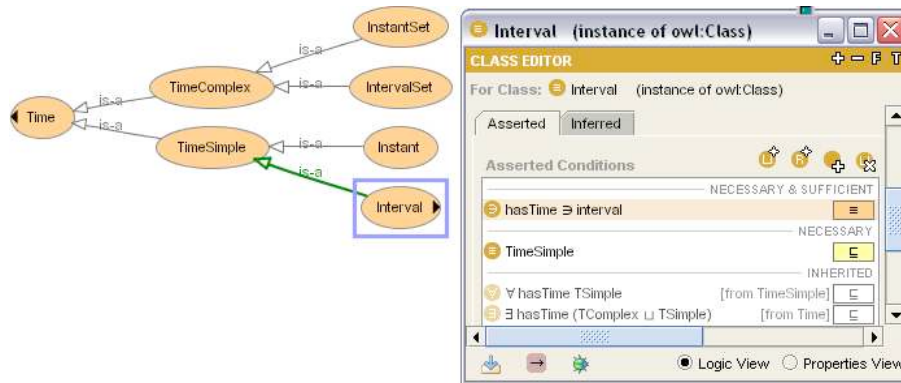


Fig. 3. OWL: Interval temporal class.

on the hasTime property is the most precise; the value of the property must be the interval instance of the TSimple class.

2.2 Schema Definition in OWL

With the MADS-OWL model, the designer can create his/her own models using the spatial and temporal dimensions of MADS-OWL. Fig. 4.1 depicts the classes that model schema T_1 in OWL. We do not show the relationships and attributes due to the complexity of the whole image. In MADS schema, the spatial object

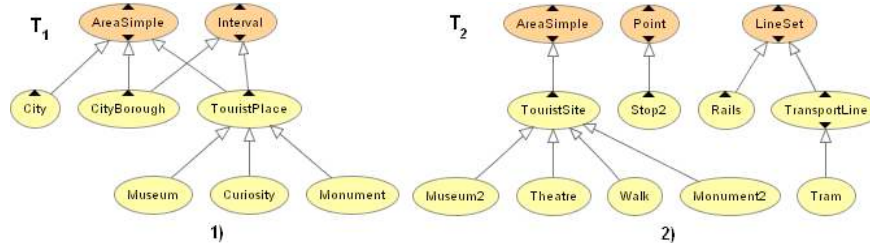


Fig. 4. Fragments of the schemas T_1 and T_2 in Protégé.

types *TouristPlace*, *City*, and *CityBorough* hold the simple area spatial semantics. Thus, in MADS-OWL we define them as subclasses of the *AreaSimple* spatial OWL class. The subclasses of *TouristPlace*, *Museum*, *Curiosity*, and *Monument*, inherit the spatial definitions of their superclass. As defined in MADS-OWL model, all the spatial classes are disjoint, i.e., a user-defined class cannot be a subclass of two spatial classes. Thus, all the subclasses of *AreaSimple* are disjoint with subclasses of *Point*. A MADS spatio-temporal object type is defined in MADS-OWL as a subclass of one of the temporal and one of the spatial classes.

We define the `TouristPlace` object type from Fig. 1.1 as a subclass of the `AreaSimple` and `Interval` MADS-OWL types. Fig. 4.2 depicts the classes that model schema T_2 in OWL. Due to OWL unique name assumption, we have added index 2 to some classes in T_2 , e.g., we name T_2 .Museum as `Museum2` in OWL.

To define a spatial attribute in MADS-OWL, we create an OWL property of type `<owl:objectProperty>` with the range restricted to a spatial class. For example, for the attribute `Start` of the object type `Walk` from Fig. 1.2, we define the OWL property `start` with the range restricted to the instances of the spatial class `Point`:

```
<owl:ObjectProperty rdf:ID="start">
  <rdfs:domain rdf:ID="Walk"/>
  <rdfs:range rdf:ID="Point"/>
</owl:ObjectProperty>
```

Properties in OWL are unidirectional, i.e., a MADS relationship is translated in OWL by two properties. Let us consider as an example the `locatedIn` relationship between object types `TouristPlace` and `CityBorough` from Fig. 1.1. This relationship holds topological inclusion semantics and therefore can link only spatial object types of specific spatial domains, e.g., a point can not include an area. In OWL, spatial property `locatedIn` is defined as a subproperty of the `include_area` topological property which belongs to the MADS-OWL model and restricts the domain and range of its subproperties by spatial classes:

```
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="include_area"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:ID="TouristPlace"/>
  <rdfs:range rdf:ID="CityBorough"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="locates"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

Note an inverse property `locates` that has `CityBorough` class as its domain and the `TouristPlace` class as its range. This property completes the definition of the MADS `locatedIn` relationship.

One of the assumptions that we make in our approach is the validity of the source schemas. Thus at this translation phase, ontology checks for each schema should not give any error. The subsumption check should not produce any new subclass/superclass relationships since all the classes within each schema are mutually disjoint. Consistency check will verify if there are contradictory restrictions, for example, cardinality constraints on relationships.

2.3 Inter-schema Mappings Definition in OWL

There are three types of the inter-schema mappings relating schemas that are to be integrated. Here we consider the first set of mappings, *Schema population Correspondences* that map related populations; we will use the same set of correspondences as we used in [4]:

- (1) $T_2.\text{TouristSite} \cap T_1.\text{TouristPlace}$;
- (2) $T_2.\text{Museum} \subseteq T_1.\text{Museum}$;
- (3) $T_2.\text{Monument} \subseteq T_1.\text{Monument}$;

In MADS, the *Schema population Correspondences* are stated between object types using the overlapping, inclusion, equality, and disjoint operators - $\{\cap, \subseteq, \equiv, \emptyset\}$. In OWL, classes are assumed to overlap, i.e., if it is not explicitly forbidden by the disjoint axiom, OWL classes can have common instances, which correspond to the overlapping relationship in MADS. Within each of our OWL schemas, we stated the `<owl:disjointWith>` axiom for all classes; but classes belonging to different schemas do not have this restriction. The inclusion in OWL is stated as the subclass restriction where subclass means necessary implication, i.e., if the $T_2.\text{Museum}$ is stated as a subclass of $T_1.\text{Museum}$, then all the instances of the $T_2.\text{Museum}$ are instances of $T_1.\text{Museum}$. This definition of subclasses in OWL corresponds exactly to the inclusion relationship definition in MADS. Thus, for the population correspondences (2) and (3) we have stated the corresponding `<owl:subClassOf>` axioms.

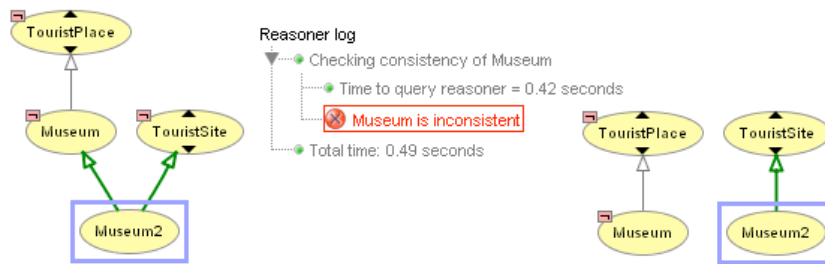


Fig. 5. Validation for the `<owl:subClassOf>` condition.

Let us now run the reasoner with an invalid condition which we have asserted intentionally: `TouristPlace` and `TouristSite` are disjoint. This condition invalidates the previous ones, as `TouristPlace` is the superclass of the `Museum`, and `TouristSite` is the superclass of `Museum2`, Fig. 4.1 and 4.2. The model asserts that with the disjoint superclasses some of their subclasses have common instances, as shown graphically in Fig. 5. If we check consistency for the `Museum` class, the reasoner finds it inconsistent. The inferred (consistent) model does not have the inclusion condition anymore.

For spatial and temporal mappings we use the MADS-OWL topological and synchronization properties respectively. For example, to state that for identical instances of the classes `Museum2` and `Museum` their spatial properties are equal, we add the `s.equal` topological property to the `Museum2` class. `s.equal` is the MADS-OWL property that expresses spatial equality; it is a symmetrical property and thus, it must have its range and domain of the same spatial type. If the designer states for example, that a museum is spatially equal to a bus

stop, the reasoner will find the `Museum` class inconsistent. To ensure that the set of mappings is not contradictory, we added several constraints in Protégé axiom language to our model. For the spatial mappings for example, there is a constraint in MADS-OWL restricting two spatial classes from having any other spatial mappings if there is a `s.disjoint` (spatially disjoint) mapping between them. A similar constraint is added for MADS-OWL temporal dimension.

3 Conclusion

To represent MADS model definitions in OWL, we defined the MADS-OWL model that is imported as a reference ontology in the user model. Then, the user defines spatial or temporal classes or properties by using the MADS-OWL ontology classes and properties as superior for his/her model elements. If the designer wants to rewrite the inherited values, his choice is checked for compatibility with parent values. If the designer defines a domain (or range) that is not a specialization of the domain (or range) of the parent property, this choice is rejected and an error message is displayed.

Our modeling approach insures that the model designer will correctly define spatial and temporal elements as well as the topological and synchronization properties. A model constructed with the help of the reference MADS-OWL model, can be checked for satisfiability considering MADS spatial and temporal semantics.

References

1. Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. In Chomicki, J., Saake, G., eds.: *Logics for Databases and Information Systems*. Kluwer Academic Publisher (1998) 229–263
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press (2003)
3. Parent, C., Spaccapietra, S., Zimanvi, E., Donini, P., Plazanet, C., Vangenot, C.: Modeling spatial data in the MADS conceptual model. In: *Proceedings of the International Symposium on Spatial Data Holding, Vancouver, Canada (1998)*
4. Sotnykova, A., Vangenot, C., Cullot, N., Bennacer, N., Aufaure, M.A.: Semantic mappings in description logics for spatio-temporal database schema integration. *Journal on Data Semantics III, Special Issue (2005)* to appear.
5. Sotnykova, A., Monties, S., Spaccapietra, S.: Semantic integration in MADS conceptual model. In Bestougeff, H., Thuraisingham, B., eds.: *Heterogeneous Information Exchange and Organizational Hubs*. Kluwer (2002)
6. Haarslev, V., Lutz, C., Möller, R.: A description logic with concrete domains and a role-forming predicate operator. *Journal of Logic and Computation* **9** (1999)
7. Lutz, C.: Description logics with concrete domains—a survey. In: *Advances in Modal Logics Volume 4*, King’s College Publications (2003)
8. (<http://protege.stanford.edu/>)
9. Haarslev, V., Möller, R.: Racer system description. In: *Proc. of International Joint Conference on Automated Reasoning, IJCAR 2001*, Springer-Verlag (2001) 701–705