Manual of the Semantic Integrity Constraint Checker v1.0

Stephan Mäs

February 12, 2012

Contents

Chapter 1

Introduction

To express our understanding of categories of things we refer to the specific properties and relations of these things. For geographical features we often use common spatial relations or dependencies on objects of other categories like in the following statements:

- Alluvial forests are adjacent to a stream.
- Alluvial forests are spatially contained by a floodplain.
- Floodplains overlap with a stream.

Such relations are commonly defined in ontologies of geospatial entity classes or as semantic integrity constraints as part of data models. Class relations can be used to formally define such properties and dependencies of categories (e.g. (Donnelly and Bittner, 2005; Donnelly et al., 2006; Tarquini and Clementini, 2008; Bittner et al., 2009)). They are used whenever the semantics of entire classes are described, independently of concrete single entities that instantiate the classes. In my research I am focusing on class relations that define cardinality restrictions for a certain relation (like in the examples topological relations) between all entities of the involved classes.

The "Semantic Integrity Constraint Checker" is a little tool for the definition and reasoning on class relations. It enables to detect implicit constraints, conflicts and redundancies in sets of class relations by analyzing the satisfiability of the defined constraints (Mäs, 2007, 2008, 2010). The implemented reasoning algorithms base on the converseness (Mäs, 2007), composition (Mäs, 2008), complements (Egenhofer, 2011) and conceptual neighborhood / transitions (Mäs, 2008) of class relations. Originally it was developed as part of my PhD thesis (Mäs, 2010) to verify the feasibility of the developed reasoning methodologies. I am continuously improving and extending the tool, whenever I find the time...

I hope the tool is useful for somebody working with class relations or at least helpful to get a basic understanding of the topic.

Chapter 2

Installation

The Semantic Integrity Constraint Checker is implemented as a tab widget plug-in for the Protégé ontology editor. Protégé is a free, open source ontology editor and knowledge base framework. It implements a rich set of knowledge modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be extended by way of a plug-in architecture and a Java-based API for building knowledge-based tools and applications. (http://protege.stanford.edu/)

Download and Installation. Please download the latest version of the Semantic Integrity Constraint Checker from www.stephanmaes.de/classrelations.html . On request I also provide the actual version of the source. Unzip the file into Protégé's plugins directory. Correctly installed the plug-in appears in the list of loaded plug-ins when Protégé is started. The plug-in can be activated by selecting it in the tab widget list: Protégé menu bar → Project → Configure... → select "ConstraintCheckerTab".

Compatibility. The plug-in has been originally implemented for Protégé 3.3.1 and JDK 1.5; this combination should still work. Version 1.1 has been tested with Protégé-Frames 3.4.7 and JRE 1.7.0. Under Protégé-OWL 3.4.7 the search in the class panel is not working, everything else is doing well. Protégé 4.x does not support this type of plug-in any more.

Disclaimer. Please note: the tool is a research prototype that might contain errors. I do not guarantee the correct working and cannot offer comprehensive support. Since the applied reasoning methodology bases on qualitative knowledge there is no guarantee that all conflicts in a set of constraints are detected.

Chapter 3

Functionality

A Protégé tab widget plug-in allows for functionality and application extensions, which appear on a new tab in the Protégé user interface. The user interface of the Semantic Integrity Constraint Checker allows specifying spatial semantic integrity constraints (SICs) for the classes defined in the Protégé knowledge base (KB). Due to deficiencies in expressiveness and reasoning capabilities the storage and reasoning on the constraints is separated from the Protégé knowledge base.

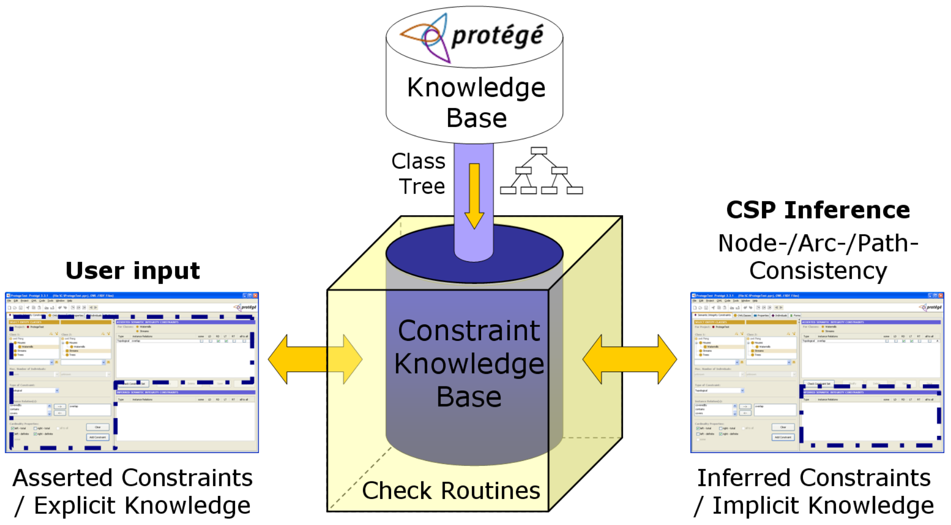
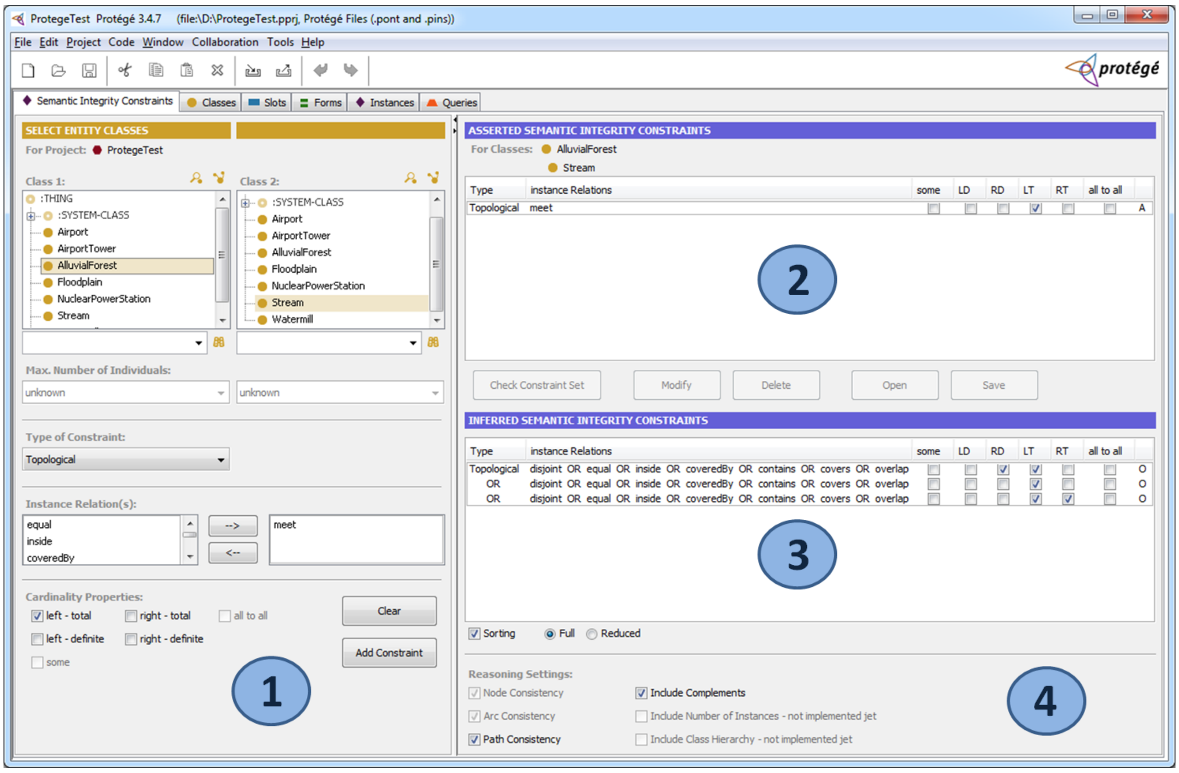

protégé
Knowledge Base
Class Tree
User input
CSP Inference
Node-/Arc-/Path-Consistency
Constraint Knowledge Base
Check Routines
Asserted Constraints / Explicit Knowledge
Inferred Constraints / Implicit Knowledge

Figure 3.1: Information exchange between Protégé ontology editor and SIC Checker

Figure 3.1 schematically illustrates the information exchange between the Protégé ontology editor and the SIC Checker. Protégé provides a KB, which contains the specified concepts of the ontology. This KB is accessible for plug-ins via a Java API. The SIC Checker Plug-in utilizes the tree of the class concepts from the Protégé KB. A further information exchange is currently not implemented. One of the reasons for choosing Protégé as a basis platform was the possibility to combine the developed reasoning methodology with existing inference capabilities of Protégé and to transfer the defined SICs back to the KB of the ontology editor for integration and possibly storing in standard ontology / rule languages like for example SWRL. This remains open for future work.

The specified spatial SICs are stored in the Constraint KB, which is managed by the plug-in and separated from the Protégé KB. The Constraint KB also contains the lists of relations for each set of instance relations and the abstract class relations, references to the corresponding identity relations and the symmetry and composition rules, etc. Such architecture with separated KBs and reasoners is commonly used to overcome deficiencies in expressiveness and reasoning capabilities in one of the system components.

3.1 User Interface


ProtegeTest Protégé 3.4.7 (file:\D:\ProtegeTest.pprj, Protégé Files (.pont and .pins))
File Edit Project Code Window Collaboration Tools Help
protégé
Semantic Integrity Constraints
Classes
Slots
Forms
Instances
Queries
Check Constraint Set
Modify
Delete
Open
Save
-->
<--
Clear
Add Constraint
1
2
3
4

SELECT ENTITY CLASSES

For Project: ProtegeTest

Class 1: :THING, :SYSTEM-CLASS, Airport, AirportTower, AlluvialForest, Floodplain, NuclearPowerStation, Stream

Class 2: :SYSTEM-CLASS, Airport, AirportTower, AlluvialForest, Floodplain, NuclearPowerStation, Stream, Watermill

Max. Number of Individuals: unknown / unknown

Type of Constraint: Topological

Instance Relation(s): equal, inside, coveredBy → meet

Cardinality Properties: [✓ left - total] right - total, all to all, left - definite, right - definite, some

ASSERTED SEMANTIC INTEGRITY CONSTRAINTS

For Classes: AlluvialForest, Stream

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	meet	☐	☐	☐	✓	☐	☐	A

INFERRED SEMANTIC INTEGRITY CONSTRAINTS

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	disjoint OR equal OR inside OR coveredBy OR contains OR covers OR overlap	☐	☐	✓	✓	☐	☐	O
OR	disjoint OR equal OR inside OR coveredBy OR contains OR covers OR overlap	☐	☐	☐	✓	☐	☐	O
OR	disjoint OR equal OR inside OR coveredBy OR contains OR covers OR overlap	☐	☐	☐	✓	✓	☐	O

[✓ Sorting] [✓ Full] Reduced

Reasoning Settings:

[✓ Node Consistency] [✓ Include Complements]

[✓ Arc Consistency] Include Number of Instances - not implemented jet

[✓ Path Consistency] Include Class Hierarchy - not implemented jet

Figure 3.2: Screenshot of the SIC Checker Plug-in

Figure 3.2 shows the user interface of the SIC Checker. It contains four components:

1. SIC definition panel
2. List of asserted SISs
3. List of inferred SICS
4. Listing and reasoning settings

SIC definition panel. The definition of a SIC (i.e. class relation) consists of four main steps. Firstly, the two entity classes that shall be restricted by the SIC are selected. The class hierarchy must be defined using the standard Protégé tabs. After selection, explicitly and implicitly defined SICs between these two classes are displayed in the lists on the right side of the tab. If the maximal number of existent instances of a class is known, it can be entered after the entity class selection. Currently these input boxes are not enabled in the user interface, since the reasoning algorithms, which check their consistency (Mäs, 2007), have not been included in the implementation so far. Secondly, the type of SIC must be selected, for instance a topological or directional spatial SIC or a non-spatial constraint such as a temporal SIC. At the moment the tool only supports topological constraints based on the eight topological relations of areal features (Egenhofer and Herring, 1990), but a corresponding extension is planned. In the third step, one or more instance relations can be selected. The fourth step is the selection of the cardinality properties to specify one of 17 class relations, which have been introduced in (Mäs, 2007, 2008).

left-total:	holds if every instance of class 1 has the selected instance relation(s) to some instance of class 2.
right-total:	holds if for each instance of class 2 there is some instance of class 1 which stands in the selected instance relation(s) to it.
all to all:	holds if all instances of class 1 have the selected instance relation(s) to all instances of class 2.
left-definite:	holds if for no instance of class 2 there is more than one instance of class 1 which stands in the selected instance relation(s) to it
right-definite:	holds if no instance of class 1 stands in the selected instance relation(s) to more than one instance of class 2
some:	holds if none of the other cardinality properties is valid, but nevertheless some instances of class 1 stand in the selected instance relation(s) to some instances of class 2

The class relation properties can be separately activated and, if possible, combined to define one of the class relations. To ensure that the inputs conform with the 17 class relations only check boxes that lead to valid class relations are enabled.

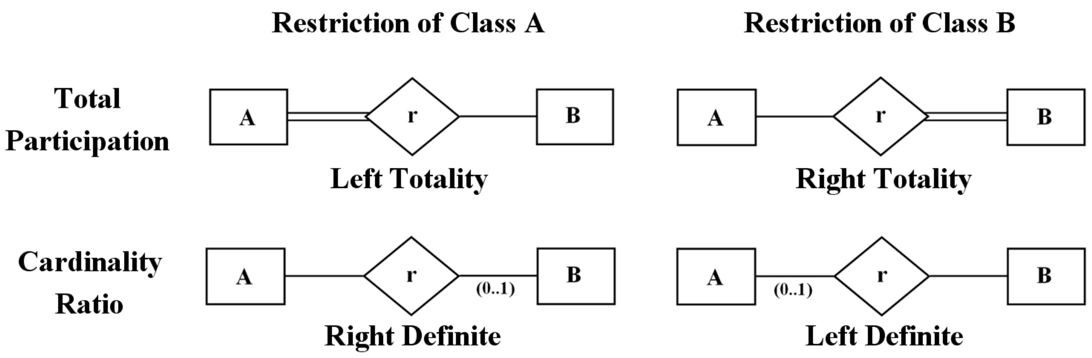


Figure 3.3: Entity Relationship Diagrams of the four cardinality properties

The final class relation results from the combination of the selected instance relation(s) and the cardinality properties. The new SIC is added to the Constraint KB when the “Add Constraint” button is pressed. The interface shown in Figure 3.2 contains the settings of the example SIC, which defines an $MEET_{LT}$ relation between the classes *alluvial forest* and *stream*. The same SIC is shown in the table of asserted constraints on the right top of the figure.

Lists of asserted and inferred SICs. The lists at the right side of the tab show the constraints defined for the classes selected in the SIC definition panel. The right columns of the tables provide information about the origin of the constraint, that means if it is asserted or the result of one of the inferences:

A	asserted constraints	**I**	identity constraints
S	symmetric/converse constraints	**C**	composition constraints
O	complement constraints		

After selection asserted SICs can be modified or deleted from the table. An import and export of asserted SICs is not jet implemented.

The inferences do not always result in a unique class relation; in many cases they lead to **disjunctions** of class relations. The table of inferred SICs in Figure 3.2 contains such a disjunction of three class relations, which is represented by the “OR” in the left column. This disjunction is the complement of the asserted constraint in the list above. Based on the given knowledge it is not possible to exclude one of the disjunction parts and the combination of all three constraints is valid.

Listing and reasoning settings. If the **"sorting" check box** in the settings area is enabled the inferred constraints are sorted according to their source (inferred from converseness, composition etc.) and the length of the disjunctions.

If inferred SICs are constraining each other the tool is automatically calculating the resulting SICs. Such reduction of inferred constraints is not always unambiguously possible. In such cases the reduction is not conducted for the constraints in the KB but can be enforced for the inferred SICs listing by means of the **"Full" / "Reduced" choice boxes**.

The reasoning settings allow for a selection of the applied inference methods explained in the next chapter.

3.2 Inferences

With the implemented reasoning algorithm and consistency check it is possible to detect conflicts and redundancies in sets of SICs (i.e. class relations).

The SICs entered into the KB originate either from user input or from inference. Before a SIC of one of these sources is added to the KB it has to pass checking routines, which detect redundancies and conflicts with the SICs already stored in the KB. Therefore some inconsistencies are found directly when constraints are defined by the user, for example duplicate SICs, invalid SICs constraining a single class or an invalid selection of instance relations in combination with cardinality properties (see subsection 6.3 in (Mäs, 2010)).

The actual inference is started when the "Check Constraints Set" button is pressed. This enables to control the reasoning process, which is a main objective of the prototype. The inference is based on the solution of a constraint satisfaction problem of class relation networks (Mäs, 2007, 2010). Therefore identity (Mäs, 2007, 2010), converse (Mäs, 2007, 2010) and complement (Egenhofer, 2011) relations, as well as compositions and conceptual neighborhood / transitions (Mäs, 2008, 2010) of class relations are analyzed. The reasoning rules of the class relations are independent of a specific set of instance relations. Therewith the overall reasoning formalism can be applied with spatial, temporal or other sets of instance relations. The only conditions are that the applied instance relations belong to the same set of JEPD relations and that this set allows for reasoning on symmetry and compositions. Anyway, at the moment only the eight topological relations among areal entities (Egenhofer and Herring, 1990) are supported.

Possible restrictions of class relations (subsection 6.4 in (Mäs, 2010)) are only applied on the inferred SICs. SICs, which are asserted by the user, are considered as fully consistent and therefore not restricted. Knowledge about the maximal number of existent instances of the classes and about the class hierarchy is not jet included for the inferences.

3.3 Possible future extensions

These are the things I might come up with sooner or later:

Reasoning extensions

- Inclusion of other instance relations (e.g. temporal relations (Allen, 1983)),
- Reasoning on number of instances,
- Inclusion of knowledge about the class hierarchy,
- ...

Functionality extensions

- Dynamic import of instance relation sets and their inference rules when starting,
- Import and export of asserted and inferred semantic integrity constraints,
- ...

Bibliography

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

Bittner, T., Donnelly, M., and Smith, B. (2009). A spatio-temporal ontology for geographic information integration. *International Journal of Geographical Information Science*, 23(6):765–798. 4

Donnelly, M. and Bittner, T. (2005). Spatial relations between classes of individuals. In Cohn, A. G. and Mark, D. M., editors, *Spatial Information Theory, International Conference, COSIT 2005, Proceedings*, volume 3693 of *Lecture Notes in Computer Science*, pages 182–199. Springer. 4

Donnelly, M., Bittner, T., and Rosse, C. (2006). A formal theory for spatial representation and reasoning in biomedical ontologies. *Artificial Intelligence in Medicine*, 36:1–27. 4

Egenhofer, M. (2011). Reasoning with complements. In De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., and Van Mingroot, H., editors, *Advances in Conceptual Modeling. Recent Developments and New Directions, ER 2011, Fifth International Workshop on Semantic and Conceptual Issues in GIS (SeCoGIS 2011)*, volume 6999 of *Lecture Notes in Computer Science*, pages 261–270. Springer. 4, 10

Egenhofer, M. J. and Herring, J. R. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, Department of Surveying Engineering, University of Maine. 8, 10

Mäs, S. (2007). Reasoning on spatial semantic integrity constraints. In Winter, S., Duckham, M., Kulik, L., and Kuipers, B., editors, *Spatial Information Theory, 8th International Conference, COSIT 2007, Proceedings*, volume 4736 of *Lecture Notes in Computer Science*, pages 285–302. Springer. 4, 8, 10

Mäs, S. (2008). Reasoning on spatial relations between entity classes. In Cova, T. J., Miller, H. J., Beard, K., Frank, A. U., and Goodchild, M. F., editors, *Geographic Information Science, 5th International Conference, GIScience 2008, Proceedings*, volume 5266 of *Lecture Notes in Computer Science*, pages 234–248. Springer. 4, 8, 10

Mäs, S. (2010). *On the Consistency of Spatial Semantic Integrity Constraints.* Dissertation, University of the Bundeswehr Munich, Neubiberg, Germany. Akademische Verlagsgesellschaft AKA GmbH, Heidelberg. 4, 10

Tarquini, F. and Clementini, E. (2008). Spatial relations between classes as integrity constraints. *Transactions in GIS*, 12(s1):45–57. 4