Using Description Logic for Reasoning about Diagrammatical Notations
Volker Haarslev
University of Hamburg, Computer Science Department
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
http://kogs-www.informatik.uni-hamburg.de/~haarslev/

1 Introduction

This paper summarizes research [1; 2; 3] about a fully implemented logical framework to develop axiomatizations defining meaningful “constellations” of abstract diagrammatic objects. The proposed framework is based on a spatial logic for describing qualitative spatial relationships between objects and on description logic (DL) as specification formalism. The framework was successfully applied to three representative diagrammatic notations: simple entity-relationship (ER) diagrams, place-transition petri nets, and a visual language for concurrent logic programming.

This logical framework forms the basis for the generic object-oriented editor GenEd that supports the formal design and analysis of visual notations. Prominent features of GenEd are (1) it is generic, i.e. domain-specific syntax and semantics of drawings are specified as TBox expressions; (2) built-in parser for actual drawings that creates ABox assertions in accordance to our spatial logic; (3) reasoning capabilities about diagrams and their specification by utilizing the classifier and realizer of DL systems.

The next section sketches our theoretical foundation for this approach. It is followed by an example session applying our framework to entity-relationship diagrams and giving a short overview of GenEd’s user interface. Afterwards we report on our experience with two major DL systems. We conclude this paper with a short discussion of related work and future research.

2 Theoretical Foundation

Our approach is based on a fully formalized theory [2] for describing visual notations that consists of three components. Each component is defined by precise semantics. Objects and relations are defined by point-sets and topology. DL can be based on model-theoretic semantics using a compositional axiomatization with set theory. GenEd implements these three components in accordance to our theory.

2.1 Geometrical Objects

The implementation of geometrical objects and recognition of spatial relations uses well-known computer graphics techniques for reasons of efficiency. The semantics of these algorithms are still specified within our theory (see [4] for a complete treatment). GenEd offers a set of predefined geometrical objects (similar to other object-oriented graphic editors) that can be used to design examples of particular notations. Supported primitive objects are points, (directed) line segments, line segment chains, and (spline) polygons. These objects can be used to compose other objects (e.g. circles, ovals, etc).

2.2 Spatial Relations

GenEd recognizes seven primitive spatial relations (disjoint, touches, intersects, contains/contained by, covers/covered by) that may hold between objects (see Figure 1). It also computes the dimension of the intersection, if applicable. The semantics are defined in analogy to a proposal by Clementini et al. [5] and are based on point-sets and topology. The relations have a parameterized ‘fuzziness’ compensating for inexact positioning of objects (caused by users or scaling factors) and floating-point arithmetic. In contrast to several other approaches for spatial relations (e.g. see [2]) GenEd can also deal with concave objects. Additionally, an arbitrary collection of objects may be grouped together and treated as a composition object. Analogous semantics for composition objects were defined.

Higher-level relations were implemented with the help of the above mentioned seven relations (see Figure 2). GenEd currently recognizes specialized containment (directly-contains/inside), connectivity (linked-with), direction of line segments (starting-from, pointing-to), and partonomies (has-part/part-of). These relations are also applicable to composition objects.

2.3 Description logic

Description logic is used to combine geometrical objects and spatial relations. GenEd supports a predefined upper model resembling built-in geometrical objects and
computed relations. Objects are represented as a hierarchy of (primitive) concept definitions and relations as a set of (primitive) role definitions also partly organized in a hierarchy. We consider visual notations as a subclass of formal languages. Language elements are specified as defined concepts that represent meaningful constellations of geometrical objects. We decided to bypass the aggregation problem and always select one geometrical object from a constellation that represents this aggregation with the has-parts role.

In the following we describe the semantics of our DL in the usual manner. Let $C$ be the set of concepts and $R$ the set of roles in a DL theory. A model is a set $D$ and an assignment function $\xi$ such that $\xi : C \rightarrow 2^D$, $\xi : R \rightarrow 2^{D^2}$ where $2^D$ is the powerset of the domain $D$, and where $\xi$ must satisfy the following conditions (concept names are denoted by c and role names by r):

\[
\begin{align*}
\xi[\text{concept name}] & \subseteq D \\
\xi[\text{role name}] & \subseteq D \times D \\
\xi[(c_1 \land \ldots \land c_n)] & = \cap_{i=1}^n \xi[c_i] \\
\xi[(c_1 \lor \ldots \lor c_n)] & = \cup_{i=1}^n \xi[c_i] \\
\xi[(\exists_n r)] & = \{x \mid \|(x, y)\mid x, y \in \xi[r]\| \geq n\} \\
\xi[(\exists_n r \land c)] & = \{x \mid \|(x, y)\mid x, y \in \xi[r] \land y \in \xi[c]\| \leq n\} \\
\xi[(\exists_n r \land c)] & = \{x \mid \{(x, y)\mid x, y \in \xi[r] \land y \in \xi[c]\|} \leq n\} \\
\xi[(\exists_n r \land c)] & = \{x \mid \|(x, y)\mid x, y \in \xi[r]\|} = n\} \\
\xi[(\exists_n r \land c)] & = \{x \mid \{(x, y)\mid x, y \in \xi[r] \land y \in \xi[c]\} = n\} \\
\xi[(\forall r)] & = \{x \mid \forall y : (x, y) \in \xi[r] \Rightarrow y \in \xi[c]\} \\
\xi[(\exists r)] & = \{x \mid \exists y : (x, y) \in \xi[r]\} = 1\} \\
\xi[(r \circ c)] & = \xi[r] \cap \{(x, y) \mid x \in \xi[c]\} \\
\xi[r_1 \circ r_2] & = \{(x, y) \mid \exists z : (x, z) \in \xi[r_1] \land (z, y) \in \xi[r_2]\}
\end{align*}
\]

3 Example Session: ER Diagrams

We applied our framework to three exemplary diagrammatic notations: simple entity-relationship (ER) diagrams, state-transition petri nets, visual programming languages, and evaluated its feasibility in the domain of geographical information systems (see [1; 3] for more details).

3.1 Knowledge Base

In the following we outline the specification of ER diagrams. Our taxonomy of concepts is designed in a way that every element of an example drawing has to be classified as an instance of a concept that is a leaf in the taxonomy. Elements that violate this property are an indication for a bug occurring either in the example program or in the KB. Figure 3 shows a screenshot of GenEd’s user interface displaying a subpart of a larger example modeling relationships in an airline company. We assume a few primitive concepts and spatial relations from GenEd’s upper model that represent geometrical objects (rectangle, circle, diamond, line, text) used in our ER diagram language. In the following, primitive concepts are typeset in a slanted style.

Connectors

A relationship-entity connection is a line that touches exactly one text label (expressing cardinality) and two other regions (rectangle or diamond). A cardinality is a text string with values chosen from the set $\{1, m, n\}$. 

\[
\begin{align*}
\text{relationship_entity} & \equiv \\
(line \land \exists_3 \text{touching}) & \land \exists_1 \text{touching text}) \\
(line \land \exists_2 \text{touching} (\text{rectangle} \lor \text{diamond})) & \land \\
(line \land \exists_1 \text{touching} \text{rectangle}) & \land (\exists_1 \text{touching} \text{diamond})
\end{align*}
\]

\[
\begin{align*}
\text{cardinality} & \equiv \\
(text \land (\forall \text{touching relationship-entity}) & \land \\
(\exists_1 \text{touching}) & \land (\forall \text{touching text value }\{1, m, n\})
\end{align*}
\]

An attribute-entity connection is a line that touches only two regions (circle or rectangle) and no text string.

\[
\begin{align*}
\text{attribute_entity} & \equiv \\
(line \land \exists_2 \text{touching}) & \land \\
(\forall \text{touching} (\text{circle} \lor \text{rectangle})) & \land \\
(\exists_1 \text{touching} \text{rectangle}) & \land (\exists_1 \text{touching} \text{circle})
\end{align*}
\]

Entities

An entity is a rectangle that contains its name. It touches at least one relationship-entity and optionally some attribute-entity connectors. It is linked with at least one diamond.

\[
\begin{align*}
\text{named_region} & \equiv \\
(region \land \exists_1 \text{containing}) & \land (\forall \text{containing text})
\end{align*}
\]
entity ≡
(rectangle ∧ named_region ∧
(∃n1 touching relationship_entity) ∧
(∀n touching (attribute_entity ∨ relationship_entity)) ∧
(∃n1 linked_with diamond) ∧
(∀n linked_with (circle ∨ diamond)))

Relationships
A relationship is a diamond that contains its name. It touches one relationship-entity and optionally some attribute-entity connectors. It is linked with two entities.

relationship ≡
(diamond ∧ named_region ∧
(∃n2 linked_with) ∧ (∀n linked_with entity) ∧
(∃n2 touching) ∧ (∀n touching relationship_entity) ∧
(∃n2 touching (=(touching ◦ text_value) 1)) ∧
(∃n1 touching (=(touching ◦ text_value) m)) ∧
(∃n1 touching (=(touching ◦ text_value) n)))

Attributes
An attribute is a circle that contains its name. It touches one attribute-entity connector and is linked with an entity.

attribute ≡
(circle ∧ named_region ∧
(∃n1 linked_with) ∧ (∀n linked_with entity))

3.2 Implementation
GenEd is implemented in Common Lisp using the Common Lisp Object System (CLOS) and the Common Lisp Interface Manager (CLIM) as interface toolkit. The classification of concepts and individuals takes place by using the lisp implementation of Classic [6; 7] as DL system. GenEd consists of 28 modules with a total of about 300 KB source code (without CLIM, CLOS, and Classic).

3.3 User Interface
The general procedure for working with GenEd is as follows. The user loads a domain-dependent knowledge base (KB) into GenEd. This KB has to comply to GenEd’s upper model. A new drawing may be created in the workspace (center window in Figure 3) or an existing one loaded. The built-in spatial parser analyzes a drawing in accordance to the upper model and creates ABox individuals and assertions resembling the elements of the drawing and their spatial relationships. Afterwards GenEd invokes the DL system. A protocol of the classification process can be displayed in GenEd’s rightmost vertical window. GenEd optionally shows the concept membership of drawing elements and several other useful information (see center window).

GenEd supports two reasoning modes. While in incremental mode GenEd records differences to previous states and reports these differences to the ABox. The reasoning process is invoked to automatically analyze drawings after every modification and to give the user an immediate feedback. If the batch mode is set drawings are always analyzed from scratch and the user has to start the reasoning process manually.

4 Experience with DL Systems
The first prototype for our framework used Loom [8] (version 2.1) as DL system. The logic implemented by Loom is quite powerful and was sufficient to express static semantics of a visual language for concurrent logic programming (see [1; 9] for examples). In case of transitive or recursive roles we escaped the standard DL by using rules or Loom’s ‘satisfies’ feature. It turned out that number restrictions for role fillers, role value maps, qualified roles, union of concepts (OR), and an implicit closed-world assumption (CWA) for selected concepts and relations are very important features of a DL suitable for the diagrammatic reasoning domain.

Our second and current prototype (GenEd) uses Classic (version 2.2) instead of Loom. A major advantage of Classic is its stable, bug-free implementation and the support of a fully implemented explanation facility for TBox and ABox reasoning. However, Classic implements only a subset of the logic supported by Loom. Classic allows only the definition of primitive roles and of role value maps that are restricted to attributes. It does not support qualified or domain/range restricted roles or the union of concepts. Closed-world reasoning is only available by explicitly closing roles for individuals. We also missed Loom’s powerful query facilities about the state of its ABox.

Since we decided to ‘live with Classic’ we had to escape Classic’s standard DL and to partially emulate or work around the missing features with the help of Classic’s rule facility. The union of concepts was accomplished by the definition of a new primitive concept representing an ‘OR-concept’ and by a set of rules each associated with one ‘OR-part’ as trigger concept. A corresponding rule fires whenever an individual is classified as member of an OR-part and it asserts the ‘OR-concept’ membership for this individual. This solution was sufficient for our domain since we needed these OR-concepts mostly at the bottom of the subsumption tree. for instance, a term (or circle diamond) is replaced by circle-or-diamond as name for a new primitive OR-concept.

The second major problem of emulating qualified roles
was much harder to solve. A range-restricted or qualified role is defined as a primitive subrole of a proper parent. The new subrole and its parent are required to have an inverse role since only the range restriction is known as trigger concept. Each new subrole is associated with a 'filler rule' that computes fillers for the inverse of this subrole. The filler rule is triggered whenever an individual is classified as member of the 'qualifying concept' and this individual is already known as filler of the parent's inverse of this new subrole. For instance, a term \((at-least\ 1\ touching\ circle)\) is replaced by \((at-least\ 1\ touching-circle)\) where \(touching-circle\) names the new primitive subrole.

The third problem is strongly related to the second one and caused by Classic's approach to closing of roles. Our domain depends heavily on inferences about 'all' and 'at-most terms' that are only possible with a CWA about the involved role and its fillers. Therefore, every role has to be closed for every known individual in order to get these inferences. Usually our solution for qualified roles results in a large set of subroles and worsens the time complexity of role closing. Moreover, the ordering of role closing is critical since filler rules may depend on the result of classifications triggered by other filler rules and role closing. In general, a top-down, breadth-first approach for closing the role hierarchy is safe if it complies to the implicit dependencies between subroles and their qualifying concepts.

5 Related Work

Our approach is mainly intended for the design and evaluation of diagrammatic visual notations/formalisms. We are not aware of any other approach using DL for diagrammatic reasoning. The understanding of diagrams
can be considered as a subproblem of image interpretation and is related to similar approaches in this area. The first treatment in this area was the MAPSEE approach [10]. However, their specifications rely on first-order predicate logic and cannot gain from the advantages of our DL approach. We also argue that DL notation — featuring concept and role definitions with inheritance and with a possible extension to concrete domains—is much more suitable for human and even mechanical inspection. This is an important issue since visual formalisms are still designed by humans. Another approach for the logical reconstruction of image interpretation [11; 12] uses DL as framework. A survey of related work for visual languages can be found in [2].

6 Conclusion and Further Research

We presented a framework for diagrammatic reasoning about visual notations that is based on DL. We successfully applied this approach to specify formal semantics of several representative visual notations. We are currently integrating an approach [13] to combine object-oriented programming and DL by offering a generic CLOS layer for programming with Classic. We are planning to incorporate concrete domains over the algebra of simple reals, i.e. extending our framework for reasoning about systems of (in)equalities over (non)linear polynomials.

References


