A Fully Formalized Theory for Describing Visual Notations
(Extended Abstract)

Volker Haarslev
University of Hamburg, Computer Science Department,
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
haarslev@informatik.uni-hamburg.de

Abstract

This paper addresses issues in visual language theory with the help of logic formalisms that were developed for reasoning tasks by the artificial intelligence and spatial databases community, especially for spatial and diagrammatical reasoning. We describe an approach based on three formal components. *Topology* is used to define basic geometric objects. Theory about *spatial relations* from the domain of spatial databases is employed to define possible relationships between visual language elements. *Description logic theory* from the AI community is used to combine topology and spatial relations. The resulting theory has been successfully applied to formally specifying semantics of visual languages. The theory's application is illustrated with a specification of entity-relationship diagrams.

1 Introduction

This paper reports on an approach to formalizing visual notations. In contrast to many grammatical approaches dealing primarily with syntactic issues of visual languages (VLs) we propose a *spatial logic* for describing semantics of visual notations. This logic combines three components (topology, spatial/topological relations, description logic) that are themselves also formally specified with precise semantics. These components were derived from research communities that are related to VL research: reasoning on diagrammatic representations and spatial databases. The goal of this paper is the attempt to intensify the dialogue between these research communities and to "advertise" the benefits of this particular view of VL theory. The successful application of our theory to a completely visual language for concurrent logic programming, Pictorial Janus (PJ) [1, 2], has been reported elsewhere [3, 4]. This experience resulted in the development of an editor for visual notations [5] whose generic semantics are based and controlled by the theory described in this paper.

Our approach is generic in the sense that particular instances can be chosen for the above mentioned components. This process depends on the nature of specific visual notations to be formally specified. For instance, the definition of PJ is mostly based on topological relations between lines, arrows, and convex regions. Therefore, we selected corresponding definitions for primitive geometric objects, an appropriate theory on spatial (topological) relations [6] that can deal with true 1D objects and regions, and a matching description logic. However, we like to emphasize that other visual languages or notations might require different definitions for objects and their possible relationships. In the following section we shortly review alternative instances for spatial relations theory.

2 Theoretical Foundation

We believe that the semantics of representational devices used for VL theory should be well understood. That is, the meanings of represented language concepts should be unambiguously determined by explicit notational devices whose meanings (semantics) are understood, so that algorithms can operate on the representation in accordance with the semantics of the notation, without needing ad hoc provisions for specific VL domains. In the following we outline a fully formalized theory for describing visual notations that consists of three components. Each component is defined by precise semantics. The definition of objects and relations is based on point-sets and topology. Description logic theory can be based on model-theoretic semantics appealing to first-order logic or using a compositional axiomatization with set theory.

2.1 Objects and Topology

The definition of basic geometric objects (the elementary vocabulary of a visual notation) usually relies on topology which is itself a basis for defining relationships between objects. In the following we assume the usual concepts of point-set topology with open and closed sets [7]. The *interior* of a set λ_n (denoted by λ_n°) is the union of all open sets in λ_n. The *closure* of λ_n (denoted by $\overline{\lambda_n}$) is the intersection of all closed sets in λ_n. The *complement* of λ_n (denoted by λ_n^{-1}) with respect to the embedding space $\Re^n$ is the set of all points of $\Re^n$ not contained in λ_n. The *boundary* of λ_n (denoted by $\partial\lambda_n$) is the intersection of the closure of λ_n and the closure of the complement of λ_n. It follows from these definitions that $\partial\lambda_n$, λ_n°, and λ_n^{-1} are mutually exclusive and $\partial\lambda_n \cup \lambda_n^{\circ} \cup \lambda_n^{-1}$ is $\Re^n$.

These definitions form the basis for Egenhofer's approach [8] to define topological relations between objects using the so-called *9-intersection*. This method defines relations between two objects by nine set intersections (every pairwise combination of interior, boundary, and complement). The following restrictions apply to every pair of sets. (1) λ_i, λ_j be n-dimensional and $\lambda_i, \lambda_j \subset \Re^n$, (2) $\lambda_i, \lambda_j \neq \emptyset$, (3) all boundaries, interiors, and complements are connected, and (4) $\lambda_i = \overline{\lambda_i^{\circ}}$ and $\lambda_j = \overline{\lambda_j^{\circ}}$. A major drawback of this approach is the failure to describe true one-dimensional objects.

This was the motivation for the proposal by Clementini et al. [6]. They extended Egenhofer's approach by introducing points and lines as additional object types and the dimension of intersections as new feature for discriminating more cases. Three types of geometric objects are modeled. *Regions* have to be convex, connected and without holes. *Lines* and *arrows* must not be self-intersecting, are either circular or directed, and have exactly two end points. *Points* are elements of lines and describe their start or end points. The boundary of a point is an empty point-set, the boundary of a line is either an empty point-set (for a circular line) or a point-set consisting of its two end points (for a non-circular line). The boundary of a region is a circular line. The interior of an object is the object without its boundary. In case of points and circular lines their interior is identical to the object itself. Both approaches cannot deal with concave objects.

A third but different approach is based on the work of Clarke about 'individuals and points' [9, 10]. Clarke's calculus interprets individual variables as ranging over spatio-temporal regions and the two-place primitive predicate, '*x is connected with y*', as a rendering of '*x and y share a common point*'. Randell and Cohn [11] developed their RCC theory based on this single property of connectedness. The RCC theory is a superset of Egenhofer's theory. It can even describe relationships with concave objects by using a convex hull operator.

Of course, there exists a strong interaction between the way of defining basic geometric objects and a set of corresponding spatial relations that can hold between these objects. Each of the above mentioned approaches defines a set of spatial (topological) relationships which are outlined in the next section.

2.2 Spatial Relations

Egenhofer's approach distinguishes eight mutually exclusive relations (out of $9^2 = 18$ different cases). The other cases can be eliminated since the above mentioned restrictions on sets have to hold. The remaining relations cover all possible cases. The 9-intersection is defined as matrix.

$$I_n(\lambda_i, \lambda_j) = \begin{pmatrix} \partial\lambda_i \cap \partial\lambda_j & \partial\lambda_i \cap \lambda_j^{\circ} & \partial\lambda_i \cap \overline{\lambda_j} \\ \lambda_i^{\circ} \cap \partial\lambda_j & \lambda_i^{\circ} \cap \lambda_j^{\circ} & \lambda_i^{\circ} \cap \overline{\lambda_j} \\ \overline{\lambda_i} \cap \partial\lambda_j & \overline{\lambda_i} \cap \lambda_j^{\circ} & \overline{\lambda_i} \cap \overline{\lambda_j} \end{pmatrix}$$

With this definition the eight cases (disjoint, meet, equal covers/coveredBy, contains/inside, overlap) can be easily characterized by the distinction between empty and non-empty intersections. For instance, the *contains* relation is specified by the 9-intersection as follows.

$$I_5(\lambda_i, \lambda_j) = \begin{pmatrix} \emptyset & \emptyset & \neg\,\emptyset \\ \neg\,\emptyset & \neg\,\emptyset & \neg\,\emptyset \\ \emptyset & \emptyset & \neg\,\emptyset \end{pmatrix}$$

Clementini et al. have to deal with $4^4 = 256$ different cases caused by taking into account the dimension of intersections. The number can be reduced to a total of 52 real cases considering the restrictions on objects. They furthermore reduced this still large number of possible relationships to five with the help of an object calculus. These five binary topological relations (touch, overlap, cross, in, disjoint) are mutually exclusive and cover all possible cases (see [6] for a proof). For instance, the *in* relation is defined as follows: object λ_2 is *in* object λ_1 if the intersection between λ_1's and λ_2's region is equal to λ_2 and the interiors of their regions intersect. It is transitive and applies to every situation.

$$<\lambda_2, in, \lambda_1> \Leftrightarrow (\lambda_1 \cap \lambda_2 = \lambda_2) \wedge (\lambda_1^{\circ} \cap \lambda_2^{\circ} \neq \emptyset)$$

Cohn and Randell define nine spatial relations (that are similar to Egenhofer's set) in terms of a single primitive relation 'C(x,y)' read as 'x is connected with y'. The authors also introduce an operator 'conv(x)' which computes the convex hull of a possibly concave object. Its definition enables reasoning with concave objects. This approach is motivated by the idea that spatial databases might easily compute whether the single relation C(x,y) holds between two objects in the database. Further deductions could be based on this primitive relation. For instance, the relation 'x is a part of y' (denoted as P(x,y)) is defined as follows.

$$P(x, y) \equiv \exists z : (C(z, x) \supset C(z, y))$$

2.3 Description Logic

This section gives a brief introduction to some aspects of description logic (DL) theory. We do not attempt to give a thorough overview and formal account of DL theory. However, we try to summarize the notions important for VL theory. We refer to [12, 13, 14] for more complete information about description logic theory.

DL theories are based on the ideas of structured inheritance networks [15]. A DL can be considered as a term rewriting language restricting the left side of equations to single unique term names. The specification of a DL consists of a set of *concepts* (or terms), a set of *roles* (binary relations that may hold between individuals of concepts), a set of disjointness assertions among concepts and among roles, a set of concept membership assertions for individuals, and a terminology, which maps names to specifications of concepts or roles. Concepts may be *primitive* or *defined*. A specification of a primitive concept represents conditions that are necessary but *not* sufficient. The specification of a defined concept represents conditions that are both necessary *and* sufficient. Primitive and defined roles are similarly specified. If a role holds between individuals, these individuals are referred to as *fillers* of this role.

Concept specifications may consist of a set of anonymous concept terms or other concept names. Unary (e.g. $\neg$) and binary operators (e.g. $\wedge$, $\vee$) are used as connectives. A concept term can also be given as a restriction of a role. *Number restrictions* specify the maximum or minimum number of allowed fillers (e.g. ($\exists_{\leq 5}$ touching), ($\exists_{\geq 1}$ inside)). *Value restrictions* constrain the range of roles and allow only fillers that are individuals of a specific concept (e.g. ($\forall$ touching arrow)). Value and number restrictions may also be combined (e.g. ($\exists_{\geq 1}$ touching arrow)). The above mentioned concept specifications are only a subset of all possible specifications. Figure 1 lists the model-theoretic semantics of commonly used specification elements (see also Section 3 for the specification of entity-relationship diagrams). The expressiveness and tractability of a particular DL depends on the type and allowed combinations of connectives and restrictions. The underlying algorithms can be NP-complete regarding their worst-case time complexity with respect to the size of the terms or even undecidable [14].

There exist several theorem provers for special types of DLs. These theorem provers (referred to as DL systems) offer powerful reasoning mechanisms that are based on the DL semantics. DL systems usually distinguish two separate reasoning components. The *terminological reasoner* or *classifier* (TBox) classifies concepts with respect to subsumption relationships between these concepts and organizes them into a taxonomy. The TBox language is designed to facilitate the construction of concept expressions describing classes (types) of individuals. The classifier automatically performs consistency checking (e.g. for incoherence, cycles) of concept definitions and offers retrieval facilities about the classification hierarchy. The forward-chaining *assertional reasoner* or *realizer* (ABox) recognizes and maintains the type (concept membership) of individuals. The purpose of the ABox language is to state constraints or facts (usually restricted to unary or binary predications) that apply to a particular domain or world. Assertional reasoners

Let $\mathcal{C}$ be the set of concepts and $\mathcal{R}$ the set of roles in a DL theory. A model is a set $\mathcal{D}$ and an assignment function ξ such that $\xi : \mathcal{C} \longrightarrow 2^{\mathcal{D}}$, $\xi : \mathcal{R} \longrightarrow 2^{\mathcal{D}^2}$ where $2^{\mathcal{D}}$ is the powerset of the domain $\mathcal{D}$, where $\mathcal{D}^2 = (\mathcal{D} \times \mathcal{D})$ and where ξ must satisfy the following conditions (concept names are denoted by c and role names by r):

$$\begin{aligned}
\xi[\textsf{concept name}] &\subseteq \mathcal{D}\\
\xi[\textsf{role name}] &\subseteq \mathcal{D} \times \mathcal{D}\\
\xi[(c_1 \wedge \ldots \wedge c_n)] &= \cap_{i=1}^{n} \xi[c_i]\\
\xi[(c_1 \vee \ldots \vee c_n)] &= \cup_{i=1}^{n} \xi[c_i]\\
\xi[(\exists_{\geq \mathsf{n}}\ \mathsf{r})] &= \{x |\ \|\{(x,y)|\ (x,y) \in \xi[r]\}\| \geq n\}\\
\xi[(\exists_{\geq \mathsf{n}}\ \mathsf{r}\ \mathsf{c})]] &= \{x |\ \|\{(x,y)|\ (x,y) \in \xi[r] \wedge y \in \xi[c]\}\| \geq n\}\\
\xi[(\exists_{\leq \mathsf{n}}\ \mathsf{r})]] &= \{x |\ \|\{(x,y)|\ (x,y) \in \xi[r]\}\| \leq n\}\\
\xi[(\exists_{\leq \mathsf{n}}\ \mathsf{r}\ \mathsf{c})] &= \{x |\ \|\{(x,y)|\ (x,y) \in \xi[r] \wedge y \in \xi[c]\}\| \leq n\}\\
\xi[(\exists_{= \mathsf{n}}\ \mathsf{r})]] &= \{x |\ \|\{(x,y)|\ (x,y) \in \xi[r]\}\| = n\}\\
\xi[(\exists_{= \mathsf{n}}\ \mathsf{r}\ \mathsf{c})] &= \{x |\ \|\{(x,y)|\ (x,y) \in \xi[r] \wedge y \in \xi[c]\}\| = n\}\\
\xi[(\forall\ \mathsf{r}\ \mathsf{c})] &= \{x |\ \forall y : (x,y) \in \xi[r] \Rightarrow y \in \xi[c]\}\\
\xi[(=\ \mathsf{r}\ \mathsf{i})] &= \{x |\ \forall y : (x,y) \in \xi[r] \Rightarrow y = \mathsf{i}\}\\
\xi[(\mathsf{r}\bullet\ \mathsf{c})] &= \xi[r] \cap \{(x,y)|\ y \in \xi[c]\}\\
\xi[\mathsf{r}_1 \circ \mathsf{r}_2] &= \{(x,y)|\ \exists z.(x,z) \in \xi[r_1] \wedge (z,y) \in \xi[r_2]\}
\end{aligned}$$

Figure 1: Semantics of DL Elements

support a query language as access to their state. Some query languages offer the expressiveness of the full first-order calculus.

2.4 Extension of DL: Concrete Domains

Existing DL systems usually cannot deal with concepts defined with the help of algebra. For instance, it is not possible to specify a *defined* concept SmallCircle that resembles every circle whose radius is less than 10mm. It is only possible to specify SmallCircle as *primitive* concept (which can never automatically be recognized) and to assert the concept membership externally. Some DL systems offer extra-logical, user-defined help functions that may assert the property (radius less than 10mm) automatically. However, these functions and their related concepts escape the DL semantics and prevent any reasoning. For instance, a concept VerySmallCircle resembling circles with a radius less than 5mm should be recognized as a specialization (subconcept) of SmallCircle. The idea of incorporating concrete domains into DL theory is to extend semantics and subsumption in a corresponding way (see [16]). CTL [17] is an example for a DL system that combines DL and concrete domains. CTL uses constraint logic programming (CLP) systems that can cope with systems of (in)equalities over (non)linear polynomials (e.g. CLP($\Re$) [18]). The above mentioned concepts SmallCircle and VerySmallCircle can be easily specified in CTL as *defined* concepts. CTL would immediately recognize the subsumption relationship between these concepts.

2.5 Application to VL

We argue that the main characteristics of DL systems are directly applicable to VL theory (see also [3] and Section 3 for a specific application):

- The TBox language is used to define VL elements as concept definitions. They are based on primitive concepts representing basic geometric objects (e.g. region, line, point). The primitive concepts form the roots of the taxonomy and are viewed as elementary lexical tokens. Defined concepts express (intermediate) semantic categories and are based on specializations of these primitive concepts.

- The classifier automatically constructs and maintains the specialization hierarchy of VL elements (defined as concepts). This hierarchy is used by the realizer to control the assertional reasoning process.

- Database-like assertion and query languages are used to state and retrieve spatial knowledge about individuals of VL programs. Example programs may be entered into the ABox by asserting primitive concept memberships for geometrical objects and spatial relationships between objects (as role fillers).

- The forward-chaining realizer automatically recognizes the most specialized concept membership (i.e. semantic category of VL element) of individuals (e.g. input tokens). It is the main source for driving the recognition process and is utilized as general visual parser.

- The automatic detection of inconsistent concept definitions or individuals is an important advantage of this approach. It is used to detect unsound (e.g. inconsistent) formal specifications (TBox) or erroneous parser input (e.g. errors in syntax or semantics).

Other (but still non-standard) characteristics are also very useful:

- The retraction of facts (stated in the ABox) is useful for supporting incremental and predictive parsing techniques in the editing process. Non-monotonic changes of users are automatically recognized and obsolete deductions retracted.

- Default reasoning can make useful assumptions about parser input while incomplete information exists.

- A DL extended to handle concrete domains could be very useful. The definition of VL elements and the possible spatial relationships between them could be solely based on DL theory with a concrete domain over $\Re$. The need for an extra-logical component that recognizes geometric features and asserts them to the DL system would be obsolete.

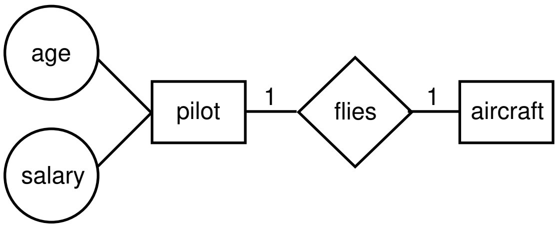
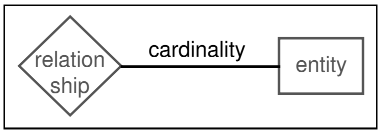


Figure 2: A simple entity-relationship diagram

3 Entity-Relationship Diagrams

The informal definition of Entity-Relationship (ER) diagrams is directly taken from [19] in order to demonstrate the expressiveness of our specification language and the reasoning capabilities of our DL system. Figure 2 shows an example ER diagram specifying a relationship between a pilot and an aircraft. We assume a few primitive concepts (denoted in *slanted* font) and spatial relations (touching, containing, linked_with, text_value) representing geometrical objects (rectangle, circle, diamond, line, text) and their relationships.

3.1 Connectors



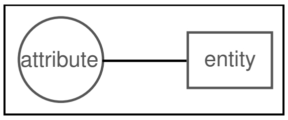
A *relationship-entity* connection is a line that touches exactly one text label (expressing cardinality) and two other regions (rectangle or diamond). A *cardinality* is a text string with values chosen from the set $\{1, m, n\}$.

relationship_entity $\equiv$
 (line $\wedge$
 ($\exists_{=3}$ touching) $\wedge$
 ($\exists_{=1}$ touching text) $\wedge$
 ($\exists_{=2}$ touching ($\mathit{rectangle} \vee \mathit{diamond}$)) $\wedge$
 ($\exists_{=1}$ touching $\mathit{rectangle}$) $\wedge$
 ($\exists_{=1}$ touching $\mathit{diamond}$))

cardinality $\equiv$

$(\mathit{text} \wedge$
$(\exists_{=1}\ \text{touching}) \wedge$
$(\forall\ \text{touching relationship\_entity}) \wedge$
$(\forall\ \text{text\_value}\ \{1, \mathsf{m}, \mathsf{n}\}))$

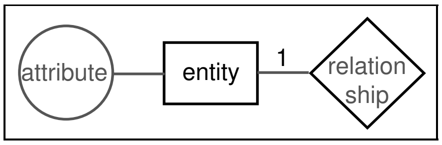
An *attribute-entity* connection is a line that touches only two regions (circle or rectangle) and no text string.



attribute_entity $\equiv$

$(\mathit{line} \wedge$
$(\exists_{=2}\ \text{touching}) \wedge$
$(\forall\ \text{touching}\ (\mathit{circle} \vee \mathit{rectangle})) \wedge$
$(\exists_{=1}\ \text{touching}\ \mathit{rectangle}) \wedge$
$(\exists_{=1}\ \text{touching}\ \mathit{circle}))$

3.2 Entities



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

named_region $\equiv$

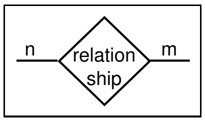
$(\mathit{region} \wedge$
$(\exists_{=1}\ \text{containing}) \wedge$
$(\forall\ \text{containing}\ \mathit{text}))$

entity $\equiv$

$(\mathit{rectangle} \wedge \text{named\_region} \wedge$
$(\exists_{=1}\ \text{touching relationship\_entity}) \wedge$
$(\forall\ \text{touching}\ (\text{attribute\_entity} \vee \text{relationship\_entity})) \wedge$
$(\exists_{=1}\ \text{linked\_with}\ \mathit{diamond}) \wedge$
$(\forall\ \text{linked\_with}\ (\mathit{circle} \vee \mathit{diamond})))$

3.3 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.

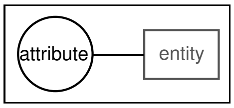


relationship $\equiv$

$(\mathit{diamond} \wedge \text{named\_region} \wedge$
$(\exists_{=2}\ \text{linked\_with}) \wedge (\forall\ \text{linked\_with entity}) \wedge$
$(\exists_{=2}\ \text{touching}) \wedge$
$(\forall\ \text{touching relationship\_entity}) \wedge$
$(\exists_{\leq 2}\ \text{touching}\ ((\text{touching} \circ \text{text\_value}) = 1)) \wedge$
$(\exists_{\leq 1}\ \text{touching}\ ((\text{touching} \circ \text{text\_value}) = \mathsf{m})) \wedge$
$(\exists_{\leq 1}\ \text{touching}\ ((\text{touching} \circ \text{text\_value}) = \mathsf{n})))$

Attributes

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



attribute $\equiv$

$(\mathit{circle} \wedge \text{named\_region} \wedge$
$(\exists_{=1}\ \text{linked\_with}) \wedge (\forall\ \text{linked\_with entity}))$

4 Related Work

There exist many approaches to specifying syntax (and to some degree semantics) of visual languages. Mostly, these are based on extensions of string grammar formalisms. A complete and recent overview is out of scope of this paper. However, we like to mention a few approaches: generalizations of attributed grammars (e.g. picture layout grammars [20]), positional grammars (e.g. [21]), and graph grammars (e.g. [22, 23, 24]). Other approaches closely related to this one use (constraint) logic or relational formalisms (e.g. [25, 26, 27, 28, 29, 30]) to represent spatial relationships. Experience has shown (reported by Wittenburg in [29]) that some grammar approaches have limitations (e.g. no arbitrary ordering of input, only special relations allowed, connected graphs necessary, bottom-up parsing applicable, no ambiguous grammars, etc.) which are sometimes unacceptable for particular application domains.

Helm and Marriott [26] developed a declarative specification and semantics for VLs. It is based on definite clause logic and implemented with the help of constraint logic programming. Marriott's recent approach is based on these ideas but utilizes constraint multiset grammars [30]. An advantage of our approach is the taxonomic hierarchy of concept definitions and the capabilities to reason about these specifications and their subsumption relationships.

Cohn and Gooday [31] applied the RCC theory to the VL domain and also developed formal semantics for Pictorial Janus. However, their specifications still rely on full predicate logic and cannot gain from the advantages of our DL approach. As far as we know, they currently do not support graphical construction of diagrammatic representations or mechanical verification processes. 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 theories about VLs are still designed by humans.

Citrin et al. [32] also present work on formal semantics of completely visual languages. They developed formal operational semantics for control in the object-oriented language VIPR but their semantics is not based on the graphical representation of the language elements.

Another approach to reasoning with pictorial concepts is based on a different, type-theoretic framework [33, 34, 35]. An important distinction is that our theory is more expressive with respect to concept definitions. For instance, in [33] the authors suggest to extend their type-theoretic approach by notions such as parameterization for construction of generic concepts and type dependency for describing pictures consisting of parts of other pictures. Our DL theory already handles the intended effects of parameterization and type dependency since its reasoning component automatically maintains a taxonomy of subsuming concept definitions which may share common subparts.

A principal advantage of our approach is the use of necessary and sufficient descriptions, i.e. *defined* concepts. Logic-based specifications using a Prolog-like style can only define sufficient but not necessary conditions. Our framework is suitable for recognizing (parsing) visual notations as well as constructing examples from specifications. Parsing can even hypothesize unknown information about notation elements. This can be accomplished with the help of ABox reasoning and the underlying model-theoretic semantics. The ABox reasoner verifies a notation example by creating a corresponding model and can automatically proof whether this model is still satisfiable if further assumption about elements are made. Our approach also support multi-level reasoning and can thus avoid problems with a combinatorial explosion of alternatives in specifications. For instance, imagine the specification of a triangle based on unordered sets of points (representing lines). We can avoid this problem since reasoning can take place about connectedness of points (low-level reasoning) as well as undirected lines (higher-level reasoning).

5 Conclusion and Future Work

We like to note that our approach has no restrictions about the ordering of input and the type of allowed relations if we incorporate concrete domains. We do not rely on special parsing techniques since our approach is purely declarative. We can even deal with ambiguous grammars since the DL realizer can compute every model satisfying the specifications. A problem with our approach could be the worst-case time complexity of the underlying classification algorithms. However, almost every logical or constraint approach with an interesting expressiveness has to deal with tractability and decidability. It is also important to note that complexity issues of DLs are well understood and analyzed. A forthcoming paper will investigate these properties and and try to apply them to building a taxonomic hierarchy of visual notations/languages.

We are currently investigating the suitability of our approach for several visual languages and notations (e.g. venn diagrams, petri nets, flow charts, etc). This investigation is facilitated by our generic editor [5] whose semantics are based on our DL theory. We are planning to incorporate description logics with concrete domains over the algebra of simple reals. The relationship between constraint approaches and description logics with concrete domains has to be more thoroughly analyzed.

References

[1] K.M. Kahn and V.A. Saraswat, "Complete Visualizations of Concurrent Programs and their Executions", in *1990 IEEE Workshop on Visual Languages, Skokie, Illinois, Oct. 4-6.* Oct. 1990, pp. 7–14, IEEE Computer Society Press.

[2] K.M. Kahn, V.A. Saraswat, and V. Haarslev, "Pictorial Janus: A Completely Visual Programming Language and its Environment (in German)", in *GI-Fachgespräch Programmieren multimedialer Anwendungen der GI-Jahrestagung 1991, Darmstadt, Oktober 1991*, J. Encarnacao, Ed. 1991, pp. 427–436, Springer Verlag, Berlin.

[3] V. Haarslev, "Formal Semantics of Visual Languages using Spatial Reasoning", In VL'95 [36], pp. 156–163.

[4] V. Haarslev, "Formal Semantics of (Completely) Visual Languages", Technical Report, in preparation, 1996.

[5] V. Haarslev and M. Wessel, "GenEd – An Editor with Generic Semantics for Formal Reasoning about Visual Notations", Technical Report, in preparation, 1996.

[6] E. Clementini, P. Di Felice, and P. van Oosterom, "A Small Set of Formal Topological Relationships Suitable for End-User Interaction", in *Advances in Spatial Databases, Third International Symposium, SSD'93, Singapore, June 23-25, 1993*, D. Abel and B.C. Ooi, Eds. June 1993, vol. 692 of *Lecture Notes in Computer Science*, pp. 277–295, Springer Verlag, Berlin.

[7] E. Spanier, *Algebraic Topology*, McGraw-Hill Book Company, New York, N.Y., 1966.

[8] M.J. Egenhofer, "Reasoning about Binary Topological Relations", in *Advances in Spatial Databeses, Second Symposium, SSD'91, Zurich, Aug. 28-30, 1991*, O. Günther and H.-J. Schek, Eds. Aug. 1991, vol. 525 of *Lecture Notes in Computer Science*, pp. 143–160, Springer Verlag, Berlin.

[9] B.L. Clarke, "A Calculus of Individuals Based on 'Connection' ", *Notre Dame Journal of Formal Logic*, vol. 22, no. 3, pp. 204–218, July 1981.

[10] B.L. Clarke, "Individuals and Points", *Notre Dame Journal of Formal Logic*, vol. 26, no. 1, pp. 204–218, Jan. 1985.

[11] D.A. Randell and A.G. Cohn, "Exploiting Lattices in a Theory of Space and Time", In Lehmann [37], pp. 459–476.

[12] R.M. MacGregor, "The Evolving Technology of Classification-based Knowledge Representation Systems", In Sowa [38], pp. 385–400.

[13] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Reswnick, and A. Borgida, "Living with Classic: When and How to Use a KL-ONE-like Language", In Sowa [38], pp. 401–456.

[14] W.A. Woods and J.G. Schmolze, "The KL-ONE Family", In Lehmann [37], pp. 133–177.

[15] R.J. Brachman and J.G. Schmolze, "An overview of the KL-ONE knowledge representation system", *Cognitive Science*, pp. 171–216, Aug. 1985.

[16] P. Hanschke, *A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning*, PhD thesis, University of Kaiserslautern, 1993.

[17] G. Kamp and H. Wache, "CTL – A Description Logic with Expressive Concrete Domains", submitted for publication, 1996.

[18] J. Jaffar, S. Michaylov, P.J. Stuckey, and R.H.C. Yap, "The CLP($\Re$) Language and System", *ACM Transactions on Programming Languages and Systems*, vol. 14, no. 3, pp. 339–395, July 1992.

[19] J.A. Serrano, "The Use of Semantic Constraints on Diagram Editors", In VL'95 [36], pp. 211–216.

[20] E.J. Golin, "Parsing Visual Languages with Picture Layout Grammars", *Journal of Visual Languages and Computing*, vol. 2, no. 4, pp. 371–393, Dec. 1991.

[21] G. Costagliola, M. Tomita, and S.K. Chang, "A Generalized Parser for 2-D Languages", in *1991 IEEE Workshop on Visual Languages, Kobe, Japan, Oct. 8-11.* Oct. 1991, pp. 98–104, IEEE Computer Society Press.

[22] H. Göttler, "Graph Grammars, a new Paradigm for Implementing Visual Languages", in *Rewriting Techniques and Applications, 3rd International Conference, RTA-89, 3-5 April 1989, Chapel Hill, NC.* Apr. 1989, pp. 152–166, Springer Verlag, Berlin.

[23] M.A. Najork and S.M. Kaplan, "Specifying Visual Languages with Conditional Set Rewrite Systems", in *1993 IEEE Symposium on Visual Languages, Bergen, Norway, Aug. 24-27.* Aug. 1993, pp. 12–17, IEEE Computer Society Press.

[24] J. Rekers and A. Schürr, "A Graph Grammar Approach to Graphical Parsing", In VL'95 [36], pp. 195–202.

[25] C. Crimi, A. Guercio, G. Nota, G. Pacini, G. Tortora, and M. Tucci, "Relation Grammars

and their Application to Multi-dimensional Languages", *Journal of Visual Languages and Computing*, vol. 2, no. 4, pp. 333–346, Dec. 1991.

[26] R. Helm and K. Marriott, "A Declarative Specification and Semantics for Visual Languages", *Journal of Visual Languages and Computing*, vol. 2, no. 4, pp. 311–331, Dec. 1991.

[27] B. Meyer, "Pictures Depicting Pictures: On the Specification of Visual Languages by Visual Grammars", in *1992 IEEE Workshop on Visual Languages, Seattle, Washington, Sept. 15-18*. Sept. 1992, pp. 41–47, IEEE Computer Society Press.

[28] K. Wittenburg, L. Weitzman, and J. Talley, "Unification-based Grammars and Tabular Parsing for Graphical Languages", *Journal of Visual Languages and Computing*, vol. 2, no. 4, pp. 347–370, Dec. 1991.

[29] K. Wittenburg, "Adventures in Multi-dimensional Parsing: Cycles and Disorders", in *1993 International Workshop on Parsing Technologies, Tilburg, Netherlands and Durbuy, Belgium, Aug. 8-10*, Aug. 1993.

[30] K. Marriott, "Constraint Multiset Grammars", In VL'94 [39], pp. 118–125.

[31] A.G. Cohn and J.M. Gooday, "Defining the Syntax and the Semantics of a Visual Programming Language in a Spatial Logic", in *AAAI-94, Spatial and Temporal Reasoning Workshop*, 1994, Preprint.

[32] W. Citrin, M. Doherty, and B. Zorn, "Formal Semantics of Control in a Completely Visual Programming Language", In VL'94 [39], pp. 208–215.

[33] D. Wang and J.R. Lee, "Pictorial concepts and a concept-supporting graphical system", *Journal of Visual Languages and Computing*, vol. 4, no. 2, pp. 177–199, June 1993.

[34] D. Wang and J.R. Lee, "Visual Reasoning: its Formal Semantics and Applications", *Journal of Visual Languages and Computing*, vol. 4, no. 4, pp. 327–356, Dec. 1993.

[35] D. Wang, J.R. Lee, and H. Zeevat, "Reasoning with Diagrammatic Representations", in *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, J. Glasgow, N.H. Narayanan, and B. Chandrasekaran, Eds., pp. 339–393. AAAI Press / The MIT Press, Menlo Park, California, 1995.

[36] *1995 IEEE Symposium on Visual Languages, Darmstadt, Germany, Sep. 5-9*. IEEE Computer Society Press, Sept. 1995.

[37] F. Lehmann, Ed., *Semantic Networks in Artificial Intelligence*, Pergamon Press, Oxford, England, 1992.

[38] J.F. Sowa, Ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo, California, 1991. Morgan Kaufmann Publishers.

[39] *1994 IEEE Symposium on Visual Languages, St. Louis, Missouri, Oct. 4-7*. IEEE Computer Society Press, Oct. 1994.