Model Outlines: a Visual Language for DL Concept Descriptions

Fernando Náufel do Amaral

LLaRC – Laboratório de Lógica e Representação do Conhecimento,
Depto. de Física e Matemática, Pólo Universitário de Rio das Ostras,
Universidade Federal Fluminense, Rio das Ostras, RJ, Brazil
Email: fnaufel@gmail.com

Abstract. The development and use of ontologies may require users with no training in formal logic to handle complex concept descriptions. To aid such users, we propose a new visualization framework called “model outlines”, where more emphasis is placed on the semantics of concept descriptions than on their syntax. We present a rigorous definition of our visual language, as well as detailed algorithms for translating between model outlines and the description logic $\mathcal{ALCN}$. We have recently conducted a usability study comparing model outlines and Manchester OWL; here, we report on its results, which indicate the potential benefits of our visual language for understanding concept descriptions.

Keywords: Description logics, visual languages, diagrammatic reasoning, usability

1. Introduction

When working with formal ontologies, one often needs to formally represent conditions for membership in the defined classes. In this paper, we will call such conditions concept descriptions, following the description logic (DL) tradition [BCM+07].

Concept descriptions are important in many scenarios related to ontology development and use. For example, DL reasoners perform logical inferences by manipulating concept descriptions according to a specific deductive calculus. In many cases, users may be interested not only in the answers provided by such reasoners, but also in the chains of reasoning that led to those answers. In order to understand such chains of reasoning, users must be able to understand the meaning of the concept descriptions involved. This area of study is referred to as proof explanation [MdS04].

Another situation where concept descriptions play an important role is in the definition and use of ontology query languages [BBFS05]; here, building a query may include writing modified concept descriptions that contain free variables (representing individuals that must be returned by the query).

Because many users of formal ontologies have no specific training in logic, the problem of representing concept descriptions in a user-friendly fashion is an important one, and many researchers have proposed different ways of solving it: replacing logical symbols with keywords in DL languages [HDG+06], automatically generating natural language paraphrases of concept descriptions [FKS06], or using diagrammatic representations [KHL+07,Gai09].

As an example to make this discussion more concrete, consider the following concept description in DL syntax (to be formally introduced in Sect. 2 below), which appears in [BFH+99], a paper about proof explanation:

$$\exists \text{hasChild} \top \tag{1}$$

$$\square \forall \text{hasChild}.$$

$$(\exists \text{hasChild.~Doctor})$$

$$\sqcup (\exists \text{hasChild.~Lawyer})$$
Diagrammatic representations of concept descriptions have given rise to implementations of “visual” ontology browsers. One such example is the visualization tool GrOWL [KWV07], which produces the diagram in Figure 1 for the concept description in (1). As can be seen, the diagram is essentially an abstract syntax tree, which offers nonspecialist users little help in understanding the semantics of the description, especially if those users are not familiar with the DL symbols \( \exists \), \( \forall \), \( \neg \) and \( \sqcup \). In fact, we have found this to be a common phenomenon: many visualization frameworks for concept descriptions are too faithful to the syntax of the representation languages (e.g., DL, OWL), a feature which may prevent users from grasping the semantics of the concept descriptions.

This paper discusses model outlines, which depart from the syntax-based tradition in that they consist of diagrams characterizing the class of models of a given concept description. (Here, we use the term “model” in the logical sense.) The model outline for (1), produced after applying a carefully defined set of simplification rules to the original concept description, is presented in Figure 2. By adhering to some simple graphical conventions, a user can understand that the concept description represents a set of individuals having at least one child and having as grandchildren (if any) only doctors and non-lawyers.

Our previous papers [dAB08,dA08] introduced the first version of model outlines and compared them to natural language paraphrases of concept descriptions. Since then, we have reformulated the visual language so as to make it more intuitive (e.g., including optional labeled clusters, rendering cardinality restrictions as text and fine-tuning the placement of inner boxes). We have also altered the conversion algorithms to conform to the new visual language.

Most importantly, we have conducted a first usability test of model outlines, with promising results. Users from different backgrounds were shown concept descriptions in two formalisms: our model outlines and Manchester OWL (a textual notation for DL which uses keywords for logical symbols, infix notation for restrictions, syntax highlighting and indentation in order to make descriptions more readable for nonspecialists — see [HDG06]). We then tested ease of understanding for each formalism by asking the users questions about the concept descriptions shown.

This paper is structured as follows: Sect. 2 presents the syntax of model outlines for the description logic \( \mathcal{ALCN} \), at the concrete (token) and at the abstract (type) levels, as is recommended for diagrammatic systems [HMST02]; Sect. 3 defines the precise semantics of model outlines, in the form of algorithms that translate from model outlines to \( \mathcal{ALCN} \) concept descriptions; Sect. 4 discusses the translation of \( \mathcal{ALCN} \) concept descriptions to model outlines; Sect. 5 reports and analyzes the results of the usability test; Sect. 6 considers some specific aspects and possible applications of our visual language; Sect. 7 contains our concluding remarks.

2. Syntax of model outlines

We consider the description logic \( \mathcal{ALCN} \), whose language of concept descriptions is specified in Figure 3, both in the DL syntax and in Manchester OWL. (Work is under way to define model outlines for more
expressive languages, such as the concept language underlying OWL 2 [HKS06].) In Figure 3, A stands for a class name (i.e., an atomic concept term), R stands for a property name (i.e., an atomic role term), and n represents a natural number. The (set-theoretical) meaning of these descriptions is given by a nonempty set $A$ (the universe or domain) along with an interpretation $I$ mapping each concept description $C$ to a set $I(C) \subseteq A$, and each role term $R$ to a binary relation $I(R) \subseteq A \times A$. An interpretation $I$ must map each description in the first two columns to the set in the third column. $\# S$ denotes the cardinality of a set $S$.

A literal is a description of the form $A$ or of the form $\neg A$, where $A$ is an atomic concept term.

The concrete syntax of model outlines defines their physical representation. What follows is an informal definition: a model outline contains clusters (solid or dashed), arrows (solid or dashed) and boxes. The root of the model outline is a solid cluster. A cluster may have an optional class label below it, consisting of a disjunction or of a conjunction of literals. So may a box. A box may also have an optional cardinality label below it, which may be of the form “(from $m$ thru $n$)”, “($m$ or more)”, or “(exactly $m$)”, with $m, n$ natural numbers, $m < n$. The source of an arrow may be a cluster or a box. The target of an arrow is always a box. Each box is the target of exactly one arrow. An arrow must have a role label above it, consisting of a role name. A box contains one or more clusters, according to constraints that we do not include in this informal description, but which will be made explicit in the abstract syntax below. A box may also contain at most one “among-which” inner box, which in turn contains one or more clusters, all of them solid. Inner boxes are never the source of arrows. A box or a cluster may have a case widget above it.

Figure 4 shows an example model outline. The target box of the arrow labeled “hasAttendance” has both a class label (“Enrolled”) and a cardinality label (“‘from 10 to 50’”). The target box of the arrow labeled “hasAttendance” also has an “among-which” inner box. This model outline does not have case widgets.

At this point, the reader should test the appropriateness of the choice of visual presentation of the components of model outlines. We suggest that the reader (without any further knowledge of the meaning of these components) formulate a natural language description of the constraints imposed upon the individuals of class GraduateCourse at the root of the outline. If the reader is knowledgeable in DL syntax, the reader should also produce an $ALCN^\prime$ concept description. In Sect. 3 below, we explain the precise meaning of this model outline, and in Sect. 4 we show the steps involved in its construction.

Case widgets indicate alternatives (i.e., disjunction). If a cluster or a box has a case widget above it, the user may browse the different cases interactively, one case at a time, by clicking on the triangles on either side of the case widget.

In Figure 5, for example, there are 4 cases altogether, specifying objects that are either (a) Books having all extras (if any) translated to Portuguese (and possibly other languages), or (b) Books having all extras (if any) in Audio format (and possibly other formats), or (c) ClassNotes having at least one Free copy in PDF format (and possibly other formats, and other copies), or (d) ClassNotes having at least one Low-priced copy (and possibly other copies).

<table>
<thead>
<tr>
<th>DL</th>
<th>Manchester</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C, D \rightarrow A$</td>
<td>$A$</td>
<td>$I(A)$</td>
</tr>
<tr>
<td>$\top$</td>
<td>THING</td>
<td>$\Delta$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>NOTHING</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>NOT $C$</td>
<td>$\Delta - I(C)$</td>
</tr>
<tr>
<td>$C \cap D$</td>
<td>C AND $D$</td>
<td>$I(C) \cap I(D)$</td>
</tr>
<tr>
<td>$C \cup D$</td>
<td>C OR $D$</td>
<td>$I(C) \cup I(D)$</td>
</tr>
<tr>
<td>$\forall R.C$</td>
<td>R ONLY $C$</td>
<td>${a \in \Delta \mid \forall b, [(a, b) \in I(R) \Rightarrow b \in I(C)]}$</td>
</tr>
<tr>
<td>$\exists R.C$</td>
<td>R SOME $C$</td>
<td>${a \in \Delta \mid \exists b, [(a, b) \in I(R) \land b \in I(C)]}$</td>
</tr>
<tr>
<td>$\leq n.R$</td>
<td>R MAX $n$</td>
<td>${a \in \Delta \mid #{b \mid (a, b) \in I(R)} \leq n}$</td>
</tr>
<tr>
<td>$\geq n.R$</td>
<td>R MIN $n$</td>
<td>${a \in \Delta \mid #{b \mid (a, b) \in I(R)} \geq n}$</td>
</tr>
<tr>
<td>$= n.R$</td>
<td>R EXACTLY $n$</td>
<td>${a \in \Delta \mid #{b \mid (a, b) \in I(R)} = n}$</td>
</tr>
</tbody>
</table>

Fig. 3. $ALCN^\prime$ concept descriptions and their meanings
Fig. 4. Example model outline

More formally, the model outline in Figure 5 corresponds to the description

\[
[\text{Textbook} \sqcap \forall \text{hasExtras}.] \\
(\exists \text{hasTranslation.Portuguese} \sqcup \\
\exists \text{hasFormat.Audio}) \sqcup \\
\{\text{ClassNotes} \sqcap \exists \text{hasCopy}.] \\
[(\text{Free} \sqcap \exists \text{hasFormat.PDF}) \sqcup \\
(\exists \text{hasPrice.Low} \sqcap \forall \text{hasPrice.Low})]
\]

As for the abstract syntax, a model outline is formally defined as a LISP-style list generated by the grammar in Figure 6, in extended BNF notation. The list representation is not meant for human consumption, but rather for automatic processing by algorithms such as the ones presented in the next section.

3. Semantics of model outlines

The appearance of the components of a model outline follows some (hopefully intuitive) graphical conventions:

- Individuals are represented by clusters of diamonds.
- The presence of a cluster (as opposed to a single diamond) emphasizes the idea that one or more individuals...
als may appear in a given situation. E.g., in Figure 4, the graduate courses in question may have as lecturers more than one tenured department professor holding a CompSci or Math PhD degree and supervising at least one graduate student from a total of 2 or more individuals.

Clusters of solid diamonds represent individuals that must exist. In Figure 4, it is mandatory that the graduate courses in question have as lecturer at least one tenured department professor holding a CompSci or Math PhD degree and supervising at least one graduate student from a total of 2 or more individuals. Likewise, the attendance must include students and graduate students.

Clusters of dashed diamonds represent optional individuals. If the cluster is labeled or has outgoing arrows, the individuals must belong to the corresponding class (e.g., “Guest” in Figure 4). If the cluster is unlabeled, the individuals may belong to any class, subject to the constraints stipulated by the label and the outgoing arrows of the outer box where the cluster is located (e.g., in Figure 4, the unlabeled cluster in the “hasLecturer” box represents lecturers that do not have to be tenured department professors, but that must hold a CompSci or Math PhD degree).

As indicated in the previous remark, box labels and arrows originating from boxes represent constraints that must be satisfied by all individuals corresponding to clusters in the box. In Figure 4, all individuals attending the graduate courses in question must belong to class “Enrolled”.

The absence of a dashed cluster in a box means that all the individuals represented in the box must belong to the classes specified by their respective labels and to the class specified by the box label and arrows (if present). This is evident in Figure 4, where it is re-
required that the lecturers hold a PhD degree only in CompSci or Math (a rather exclusivist and unfair requirement, but this is only an example).

**Dashed boxes**, always the target of dashed arrows, always contain a dashed cluster, representing optional individuals. In Figure 4, the graduate courses in question may or may not involve the use of (up to 2) department labs.

“Among which” **inner boxes** contain clusters representing individuals that belong to subclasses of one or more classes specified in the outer box. In Figure 4, the attendance of the graduate courses in question consists of students, some of which are required to be graduate students. Optionally, guests may attend.

The above remarks are included here only for pedagogical purposes. In fact, we define the precise semantics of model outlines by means of the **DESCR** procedure, which, when given a model outline *C* (in abstract syntax), yields the **ALCN** concept description taken as the meaning of *C*. The **DESCR** procedure calls **BOXDESCR** to build the concept description denoted by a box. Algorithm 1 shows both procedures in pseudocode.

The reader should refer to the grammar in Figure 6 for the structure of the lists that the algorithms manipulate. These algorithms can be modified to produce more legible output; here, their only purpose is to serve as the precise semantics of model outlines. When given as input the model outline in Figure 2, e.g., algorithm **DESCR** returns the following description, which is equivalent to (1):

\[ \bot \sqcup \{ T \sqcap \neg \text{hasChild}.(\bot \sqcup \bot \sqcup T) \sqcap T \sqcap \exists \text{hasChild}.(\bot \sqcup T) \sqcap \neg \text{hasChild}.[T \sqcap \neg \text{hasChild}.(\bot \sqcup \bot \sqcup (\text{Doctor} \sqcap \neg \text{Lawyer}))] \}\]

### 4. Constructing model outlines

In [dAB08] we presented a first algorithm for translating **ALCN** concept descriptions into model outlines. In this paper, we incorporate some important changes to the algorithm (e.g., to account for labeled optional clusters) and give a more informal explanation of the main steps involved in such a translation, using as a working example the concept description that originated the model outline in Figure 4.

Given an **ALCN** concept description *C*, we start by converting *C* to modified disjunctive normal form (mDNF), applying simplification rules in the process. A concept description is in mDNF if it fits the pattern

\[ D_1 \sqcup \ldots \sqcup D_n \]

where each disjunct *D* is a conjunction of the form

\[ C_1 \sqcap \cdots \sqcap C_p \]
Algorithm 1. Conversion from model outlines to $\mathcal{ALCN}$

where each conjunct $C_j$ is either a literal, or a collection of "intervals" of natural numbers (whose upper bound may be $\infty$) associated to a role $R$, or a description of the form $\forall R.C'$ or of the form $\exists R.C'$, where $C'$ is itself in mDNF.

The modification is in the way number restrictions are represented: using appropriate rewrite rules, any conjunction of cardinality restrictions over a role $R_i$ can be converted to a collection of "intervals" of natural numbers; for role $R$, the interval $[m, n]$ represents the constraint ($\geq m.R$ and $\leq n.R$). Likewise, $[m, m]$ represents ($= m.R$), and $[0, m]$ represents ($\leq m.R$), and $[m, \infty]$ represents ($\geq m.R$).

To each $D_i$, we then apply the simplification rule

$$\forall R.C_1 \cap \ldots \cap \forall R.C_n \Rightarrow \forall R.\left( C_1 \cap \ldots \cap C_n \right)$$
As a result, we obtain $C'$, which is a disjunction $D'_1 \sqcup \ldots \sqcup D'_n$, where each $D'_i$ can be written as

$$L_1 \sqcap \ldots \sqcap L_m \sqcap C_1 \sqcap \ldots \sqcap C_p$$

where each $L_i$ is a literal, and each $C_j$ can be written as

$$\forall R.F \sqcap \exists R.G_1 \sqcap \ldots \sqcap \exists R.G_q \sqcap K$$

where $F$ and all the $G_i$ are in mDNF and $K$ is a collection of intervals of natural numbers representing cardinality restrictions over role $R$. Any (or all) of these elements may be absent. Note that we have grouped the conjuncts according to the role $R$ they act upon. Note also how the cardinality constraints in lines (2g), (2h), (2m) and (2o) have been written with (singleton) collections of intervals of natural numbers. Two transformations must be effected before the model outline can be built.

The first one concerns lines (2b)–(2d), where the set of objects related to the lecturers through $\text{holdsPhDIn}$ is closed; i.e., the lecturers must hold some PhD degree in CompSci or Math and only PhD degrees in CompSci or Math.

The algorithm detects such a closure whenever it finds conjuncts of the form

$$\forall R(C_1 \sqcup \ldots \sqcup C_n) \sqcap \exists R.C_1 \sqcap \ldots \sqcap \exists R.C_n$$

Here, we have $n = 1$ and $C_1 = \text{CompSci} \sqcup \text{Math}$. Then, to indicate the closure, the algorithm refrains from adding a dashed, unlabeled cluster to the target box of the $\text{holdsPhDIn}$ arrow (see Figure 4).

The second transformation is similar: in lines (2i)–(2l), we can see there is some sort of closure related to the role $\text{hasAttendance}$, but the situation is more complicated. In fact, this is the general case, which also includes the first transformation. Whenever the conjuncts for role $R$ are of the form

$$\forall R[(C_1 \sqcap D) \sqcup \ldots \sqcup (C_n \sqcap D)]$$

where $D$ is a conjunction (with $D = \top$ as the trivial case) it proceeds as follows:

- Solid clusters for $C_1, \ldots, C_n$ are created in the main target box for the $R$-arrow.
- The main target box for the $R$-arrow gets $D$ as a label. If $D = \top$, this label is not shown.
- The main target box for the $R$-arrow gets an “among which” inner box containing solid clusters for $F_1, \ldots, F_q$.
- Dashed clusters for $C_{n+1}, \ldots, C_p$ are created in the main target box for the $R$-arrow.

In our example description, in lines (2i)–(2l), we have that $n = 1$, and $C_1 = \text{Student}$, and $D = \text{Enrolled}$, and $p = 1$, and $C_2 = \text{Guest}$, and $q = 1$, and $F_1 = \text{GradStudent}$.

Algorithms 2–4 show the detailed pseudocode for converting an $\mathcal{ALCN}$ concept description $C$ in modified DNF to the abstract syntax of its corresponding model outline. Each procedure is accompanied by some remarks about its behavior.

In order to build the entire model outline for an $\mathcal{ALCN}$ concept description $C$, one should call the procedure as $\text{BUILDCLUSTERCASES}($solid, $C)$. 

---

**GraduateCourse**

1. $\forall \text{hasLecturer}$. (2a)
2. $\exists \text{supervises}. \text{GradStudent}$. (2e)
3. $\forall \text{hasLecturer}. \text{DeptProfessor}$. Tenured (2f)
4. $\forall \text{hasLecturer}. \text{GradStudent}$. (2i)
5. $\forall \text{hasLecturer}. \text{GradStudent}$. (2l)
6. $\forall \text{usesLab}. \text{DeptLab}$. (2m)
7. $\forall \text{usesLab}. \text{GradStudent}$. (2o)

Note how the constraints have been grouped by the roles they act upon. Note also how the cardinality constraints in lines (2g), (2h), (2m) and (2o) have been written with (singleton) collections of intervals of natural numbers.
The pseudocode of BUILDCLUS is detailed in Algorithm 2.

Algorithm 2. Conversion from ALCN to model outlines: BUILDCLUS

The remarks below will help the reader follow the pseudocode of BUILDCLUS in Algorithm 2:

- Lines 1–8 treat the base cases, where the description in modified DNF is \( \top, \bot, \) a disjunction of literals, or a conjunction of literals, respectively.
- In line 2, this piece of abstract syntax should be rendered visually in such a way as to make it clear that \( C \) is inconsistent (i.e., \( C \) denotes the empty set).
- If the algorithm does not exit before or at line 8, then we know \( C \) is of the form \( D_1 \sqcup \cdots \sqcup D_n \), with \( n \geq 1 \), at least one \( D_i \) not a literal.
- More precisely, \( C \) is of the form \( L_1 \sqcap \cdots \sqcap L_m \sqcup D_1 \sqcup \cdots \sqcup D_n \), with \( m \geq 0 \), \( n \geq 1 \), with all of the \( L_i \) literals, and all of the \( D_i \) nonliterals.
- Line 11 gathers all the literals in \( L_1 \sqcap \cdots \sqcap L_m \) into one single case.
- The loop in lines 12–18 produces a case rooted in the present cluster for each of the remaining disjuncts \( D_i \), each of which is of the form

\[
L_1 \sqcap \cdots \sqcap L_m \sqcap C_1 \sqcap \cdots \sqcap C_p
\]

subject to all of the following conditions:

* \( m + p > 0 \) (i.e., there must be at least one conjunct; otherwise, \( C \) would be \( \top \), and would have been treated in one of the base cases of the algorithm).
* If \( p = 0 \) then \( m \geq 2 \) (i.e., if there are only literals in the conjunction, there must be more than one literal; otherwise, \( D_i \) would be a literal, and would have been treated in line 11 above, which gathers all literals into a single case).
- In line 14, if \( m = 0 \), then \( \text{classLabel} \) should be empty. If \( m = 1 \) then, it should get \( L_1 \), with no \( \text{AND} \).
- Now, each each \( C_j \) consists of a conjunction of all restrictions over a single role \( R_i \), including cardinality restrictions.
- Then, in the loop consisting of lines 16–17, each \( C_j \) gives rise to an arrow leaving the present case, and in line 18 the present case is added to the set of cases associated to the present cluster.

Each arrow (along with its target box or box cases) is built by procedure BUILDARROW, whose pseudocode is detailed in Algorithm 3.

The following remarks are intended to help the reader follow the steps involved:

- Strictly speaking, the BUILDARROW procedure is non-deterministic, as there may be more than one way to parse the input concept description \( C \) in the form outlined in lines 1–6.
Algorithm 3. Conversion from $\mathcal{ALCN}$ to model outlines: BUILD\textsc{arrow}
Algorithm 4. Conversion from $\text{ALCN}$ to model outlines: BUILDBOXCASE

- More precisely, there may be more than one conjunct $D$ common to all disjuncts in the $\forall R$ universal restriction.
- Concept descriptions $C_1 \ldots C_n$ will correspond to solid clusters in the target box.
- Concept descriptions $C_{n+1} \ldots C_{n+p}$ will correspond to dashed clusters (i.e., optional individuals) in the target box.
- Concept description $D$ will correspond to the target box’s label and to arrows leaving the target box.
- Concept descriptions $F_1 \ldots F_q$ will correspond either to solid clusters in the “among which” inner box or to solid clusters in the target box, depending on circumstances explained below.
- The procedure is structured in 3 cases:

1. $n = 0 \land p = 0$ (starting on line 7): there is only one target box case. If present, concept descriptions $F_1 \ldots F_q$ give rise to solid clusters. If there are no universal restrictions, the target box will contain a dashed, unlabeled cluster, unless the cardinality restrictions $K$ force the existence of individuals in the target box, in which case, the unlabeled cluster will be solid.
2. $n > 0$ (starting on line 17): there is only one target box case. Concept descriptions $C_1 \ldots C_n$ will give rise to solid clusters. If present, concept descriptions $C_{n+1} \ldots C_{n+p}$ will give rise to dashed, labeled clusters corresponding to optional individuals. If present, concept descriptions $F_1 \ldots F_q$ will give rise to solid clusters in the “among which” inner box. $D$ will give rise to the target box’s label and outgoing arrows.
3. $n = 0 \land p > 0$ (starting on line 26): multiple box cases may be produced here. Each of $C_{n+1} \ldots C_{n+p}$ that is not a literal will give rise to a box case (literals will be gathered into one single box case and appear as a disjunction in the dashed cluster’s label). If present, concept descriptions $F_1 \ldots F_q$ will give rise to solid clusters in the “among which” inner box, which will be common to all box cases. $D$ will participate in the target box’s label and outgoing arrows.

Finally, procedure BUILDBOXCASE in Algorithm 4 is responsible for building each box case. Concept description $D$ gives rise to the box’s label and outgoing arrows. The contents of the box case are generated from parameters $E$ (solid clusters) and $opt$ (dashed clusters and/or an “among which” inner box). Note that the stroke of the box (dashed or solid) and the cardinality restrictions are actually associated to the arrow incident to the box, which is computed by the BUILDARROW procedure.

5. Usability Evaluation

We have conducted a usability study in order to evaluate our proposed diagrammatic notation. The main aim was to test the usefulness of model outlines for the understanding of complex concept descriptions.

Note that it is the model outline notation itself that is being evaluated, not a specific graphical user interface (GUI) implementing the notation. Thus, the focus of the study is on understanding, not on interaction. We find this to be an advantage, as changes can be made to the notation before we are committed to a specific GUI, and problems can be identified in relation to spe-
pecific features of the notation, so that special attention can be given to these problems in order to solve or mitigate them through the use of appropriate human interaction techniques. From a practical point of view, this potentially reduces the need for radical, costly changes after implementation.

Likewise, we have chosen model outlines for the simpler ALCN language so we could find out early if something needs to be changed in our most basic assumptions. The result of this test will help us design the extensions of model outlines to deal with more expressive concept languages.

Following [DR94], we defined our main goal as: Model outlines can help users with little or no training in Logic to understand complex concept descriptions. In particular, model outlines are more effective than Manchester OWL for this task.

Manchester OWL (see Figure 3 and also [HDG+06]) is a textual notation for DL which uses keywords for logical symbols (e.g., “SOME” for “∃”), infix notation for restrictions (e.g., “hasChild SOME Man” for “∃hasChild.Man”), syntax highlighting and indentation in order to make descriptions more readable for nonspecialists. So, we are comparing our diagrammatic notation with a textual notation designed for the same target audience. (As the test participants were all Brazilians, we used Portuguese translations of the Manchester OWL keywords.)

Next, we defined a set of concerns, in the form of questions like: Can users understand the meaning of X? where X is one of the elements present in model outlines (solid clusters, dashed clusters, arrows, boxes, inner boxes, case widgets, etc.). Specific concerns were also formulated (e.g., “Can users understand that individuals in “among which” inner boxes are mandatory?”).

We selected 10 participants for our study (note that [DR94] recommends 6 to 12). These participants come from several backgrounds and occupations, as detailed below. All received detailed information on the procedures and on their rights as participants. All signed terms of informed consent.

One session of the study consisted of the following activities: a pre-test questionnaire, a tutorial on notation A, a specification on domain X using notation A, 15 questions, a post-task questionnaire, a tutorial on notation B, a specification on domain Y using notation B, 15 questions, a post-task questionnaire, and a post-test questionnaire. Notations A and B alternated between model outlines and Manchester OWL. Domains X and Y alternated between graduate courses (which included Figure 4 of this paper) and family relations. Each participant answered 15 questions for each domain. The questions for each domain were fixed, regardless of the notation used. For each domain, half the participants answered questions on model outlines, and half answered questions on Manchester OWL specifications. Half the participants saw model outlines before Manchester OWL, and half saw Manchester OWL before model outlines.

The number of correct answers and the time to answer were measured. Additional information was obtained in the form of comments collected through the “thinking out loud” protocol [DR94] and through questionnaires. Table 1 shows the occupation and the number of correct answers for each participant.

<table>
<thead>
<tr>
<th>Occupation</th>
<th>Correct answers (model outlines)</th>
<th>Correct answers (Manchester OWL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logician</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>Theoretical physicist</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>Software engineer</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>Secretary</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Nurse</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Graphics designer</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Social worker</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>Comp. Science undergrad</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>Production engineer</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>Mathematician</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Totals</td>
<td>137</td>
<td>116</td>
</tr>
<tr>
<td>Percentages</td>
<td>91.3%</td>
<td>77.3%</td>
</tr>
</tbody>
</table>

For the graduate courses domain, we note the following highlights:

Question 8 was related to Figure 4 of this paper, and elicited 5 errors using Manchester OWL, and no errors using model outlines. The question was: “If a course is attended only by students that are not graduate students, does the course meet the specification?” The error was probably induced by the abbreviation recommended in [HDG+06]:

\[
\text{hasAttendance SOME [Student, GradStudent]}
\]

which seems to have evoked the idea that the bracketed list consisted of a set of alternatives. This question was answered correctly by all participants using
model outlines, which indicates that users understood the meaning of “among which” inner boxes.

Question 14 elicited 4 errors using Manchester OWL, and 3 errors using model outlines. This question was about a specification consisting of 4 cases. The situation proposed in the question satisfied exactly one of the 4 cases. With Manchester OWL, the participants had difficulty in finding their way among multiple parentheses and complex disjunctions. With model outlines, they apparently thought that the proposed situation had to satisfy all cases.

For the family relations domain, we note the following highlights:

Question 6 elicited 4 errors using Manchester OWL, and no errors using model outlines. This question asked if a person satisfying the given specification could have jobless children. The specification in Manchester OWL included the sentence

\textit{hasChild SOME (Man AND worksAt ONLY Hospital)}

Apparently, the users forgot that “ONLY” (which stands for “\(\forall\)”) does not imply the existence of objects. In the model outline, the presence of a dashed cluster, a dashed box and a dashed arrow made it clear that existence was not required.

Question 8 elicited 3 errors using Manchester OWL, and 1 error using model outlines. This question asked if a person satisfying the given specification had to have a grandchild working as a surgeon. Some users found it confusing to follow the composition of roles (\textit{hasChild–hasChild}), and were again, as in question 8 about graduate courses, confused by the Manchester OWL abbreviation “\textit{SOME \[ \cdots \]}”. In the model outline, the presence of a solid cluster labeled \textit{Surgeon} inside an “among which” inner box made the correct answer more clear.

One trend was clearly observed in both domains: specifications that involve cases (i.e., complex disjunctions), such as the one in Figure 5 of this paper, are more difficult to understand than those that do not, as Table 2 indicates.

Among the comments offered by the participants, many indicated confusion due to the way cases were presented in model outlines (like in Figure 5 of this paper, the layout consisted of 4 diagrams on a single page). Some users thought that all 4 diagrams had to be satisfied. This is clearly one weakness of model outlines (on paper) that we must try to eliminate in the GUI implementation. We predict that such confusion will not arise if the user interacts with the model outline (e.g., dynamically expanding and collapsing cases). The GUI should also make clear when clusters in different cases actually correspond to the same cluster, by showing one single cluster which can be expanded in different ways.

As for time: in the courses domain, each user took on average 28 seconds per question, regardless of the notation. In the family relations domain, each user took on average 26 seconds per question with model outlines, but 40 seconds per question with Manchester OWL.

Of the 10 participants, 5 said they preferred model outlines, 4 said they liked both notations equally well, and 1 said both notations were equally bad.

6. Discussion

Among several questions related to our proposed visual framework, we examine here the issue of possible applications and a more specific question regarding the interaction of concept descriptions with the axioms in the TBox of a formal ontology.

6.1. Applications

Apart from the usability issues discussed in Section 5, we can examine the question of \textit{usefulness}; more precisely, in what applications and tasks can model outlines be of help to users?

The algorithms presented in Sections 3 and 4 provide a translation-based formal semantics to our visual language. Such algorithms also suggest two basic ways of using model outlines:

In a first scenario, an ontology browser can show nonspecialist users model outlines corresponding to complex concept descriptions (such as necessary and sufficient conditions in the definitions of classes). This was our original motivation, and this is the application the usability test was designed to take into account (investigating how well nonspecialist users could un-
understand concept descriptions rendered as model outlines). Here, a concept description is translated into a model outline, and the user takes on a more or less passive stance, interacting with a finished diagram only to collapse and expand elements, switch the focus to certain cases, etc.

In a second situation, the user may be expected to create and edit model outlines. We can envision a dedicated graphical editor which will guide the user in the task of assembling a diagram from basic elements such as clusters, boxes and arrows, providing autocompletion and other facilities to help the user find the desired class and role names to include in labels. In this scenario, the user takes on a much more active role, and the interaction provided by the graphical editor must be carefully planned in order to prevent the user from building nonsensical diagrams or, even worse, to prevent the user from becoming confused because a certain operation he is trying to perform (and which he finds to be intuitive) is not being allowed by the editor. In other words, the manual construction of model outlines by the user presents interaction and usability problems that seem to be harder than those involved in the mere exhibition of finished model outlines. If such a graphical editor is to be made available, more specific usability tests must be conducted first.

The manual construction of model outlines is an activity that may be required in at least two tasks: the user may want to define concept descriptions to serve as necessary and sufficient conditions in class definitions, or the user may want to construct a diagram to submit as a visual query on the ontology he is working on. In this latter case, the definition of model outlines must be altered to provide features that are typical of query languages, such as clusters marked as variables to be bound by the query results. One possibility that seems worth investigating is the formatting of the query’s results themselves as model outlines, allowing the same visual language to be used for queries and for the data returned by them.

The use of model outlines to exhibit instance data leads to a third possible application: model exploration [Bau09], where a user may interact with generated models of the formal ontology he is working on, so as to gain better understanding of the axioms in the ontology and their logical consequences and/or to test conjectures.

6.2. Model outlines and the TBox

Model outlines have been originally designed to denote “standalone” concept descriptions, which means that the atomic concepts and the role names that appear in the diagram are not constrained in any way other than what is shown in the diagram itself. However, in real life, concept descriptions usually appear in the context of a formal ontology, whose TBox axioms affect the interpretation of atomic concepts and role names.

A simple example is the fact that a solid cluster labeled “Class1 AND Class2” will denote nonexistent individuals if the ontology’s TBox contains the axiom

\[ \text{Class1} \sqsubseteq \neg \text{Class2} \]

which states that the atomic concepts Class1 and Class2 are disjoint.

A slightly more complex situation is illustrated by Figure 4, which seems to suggest that GradStudent is a subclass of Student, but this type of knowledge belongs in the TBox, and should not be conveyed (erroneously) by a model outline! To be exact, the “among which” inner box in Figure 4 is only saying that if GradStudent and Student are disjoint, then the GraduateCourses at the root of the diagram can have no attendance.

This is a general phenomenon related to “among which” inner boxes. A question that naturally arises, then, is whether these inner boxes are really necessary in the visual language.

Consider the model outline in Figure 7. It corresponds to the ALCN description

\[ \forall R. (A \sqcup B) \sqcap \exists R.A \sqcap \exists R.B \sqcap \exists R.C \]

The modified model outline obtained by removing the inner box and placing the cluster labeled C in the main box, along with the clusters labeled A and B, shown in Figure 8, corresponds to the ALCN description

\[ \forall R. (A \sqcup B \sqcup C) \sqcap \exists R.A \sqcap \exists R.B \sqcap \exists R.C \]

It can be shown that these two descriptions are equivalent if and only if the interpretation of C is contained in the interpretation of A \sqcup B, but this containment is not the kind of information we have designed model outlines to convey.
In the context of a model-outline-based tool used for ontology browsing, one solution would be to invoke a DL reasoner to check the consistency of each cluster label and each box, using the knowledge present in the TBox. Then the tool could somehow visually highlight the inconsistent elements of the diagram, alerting the user.

In the context of a model-outline-based visual query language, it seems that “among which” inner boxes can be eliminated without essentially compromising the range of queries that can be formulated. This is because the user could build a query corresponding to the modified version in Figure 8 and be informed, in the results, of the possibility that individuals could appear at the same time as instances of $A \sqcup B$ and of $C$. If, on the other hand, the TBox implies that $C$ and $A \sqcup B$ are inconsistent concepts, no results would be returned by the query.

7. Conclusions

The main achievements of the work related here are the reformulation of our model outline visual language and its related algorithms, as well as the results of our first usability test, comparing model outlines to Manchester OWL.

Ontology visualization is a very active field of study. The survey [KHL+07] discusses over 40 ontology visualization tools, all of them developed in the past 10 years. All of those tools are general, in the sense that they use one single visualization framework to show several types of information about the ontology: the subsumption hierarchy, roles, etc. In particular, those tools show concept descriptions either textually (e.g., Protégé) or in the form of abstract syntax trees (as in Figure 1 of this paper).

Model outlines, on the other hand, are specialized, having been designed specifically to show concept descriptions. Although the notation used is new, our usability test indicates it is intuitive enough to be understood by non-specialists. The specialized nature of model outlines suggests that they can be integrated with a more general tool, so that users can easily switch views, e.g., from the subsumption hierarchy as a tree to the definition of a class as a model outline.

We are currently implementing a concept description browser based on model outlines, as a Protégé plugin. We are taking special care to rely on graphical conventions and interaction techniques that profit from the vast body of knowledge related to visual perception and cognitive principles, as described, e.g., in [War04].

Work is also under way to extend model outlines to the concept language associated to OWL 2 [HKS06].

References


