Visualizing the Semantics (Not the Syntax)
of Concept Descriptions[*]

Fernando Náufel do Amaral
LLARC – Laboratório de Lógica e Representação do Conhecimento
Departamento de Ciência e Tecnologia
Pólo Universitário de Rio das Ostras
Universidade Federal Fluminense
Rio das Ostras, RJ, Brasil
fnaufel@ic.uff.br

ABSTRACT

The development and use of ontologies may require users with no training in formal logic to handle complex concept descriptions (e.g., in the form of necessary conditions in class definitions). To aid such users, alternative representations of concept descriptions have been proposed, such as natural language paraphrases and visualization frameworks. We examine examples of those representations and propose a novel visual framework, where more emphasis is placed on the semantics of concept descriptions than on their syntax.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages

General Terms

Languages, Theory

1. INTRODUCTION

Not all people who use ontologies can write or understand the complex concept descriptions that appear as conditions in class definitions. In [9], for example, the authors show how difficult it may be for rather technically sophisticated users (e.g., Computer Science undergraduate and graduate students) to come up with correct specifications of a supposedly simple concept such as the class of Margherita pizzas.

End users of ontologies (including scholars from areas such as Computer Science, Linguistics, Engineering or Medicine) are usually not familiar with the various syntaxes used to represent concept descriptions (DL, OWL, RDF, etc.), let alone with the subtleties involved in their semantics. Since one of the aims of the use of ontologies in the Semantic Web is to make knowledge widely accessible, the problem

of finding easy-to-understand representations of concept descriptions is an important one.

Efforts to present concept descriptions in more user-friendly ways include (possibly controlled) natural language paraphrases. Take, for example, the complex description in DL syntax [1]:

$$\exists hasChild.\top \sqcap$$
$$\forall hasChild. \tag{1}$$
$$\neg((\exists hasChild.\neg Doctor) \sqcup (\exists hasChild.Lawyer))$$

This description denotes the class of individuals that have at least one child and all of whose children do not have non-doctors or lawyers as children. In other words, it represents individuals that have only doctors and non-lawyers as grandchildren. One way to make this description more readable is to substitute keywords for logical symbols and to write quantifiers in infix notation (e.g., *hasChild* **SOME** *Lawyer* instead of $\exists hasChild.Lawyer$). This is the approach taken by the the Manchester OWL syntax [6], which renders the concept description in (1) as

$$(hasChild \textbf{ SOME } Thing) \textbf{ AND }$$
$$hasChild \textbf{ ONLY NOT}$$
$$((hasChild \textbf{ SOME NOT } Doctor) \tag{2}$$
$$\textbf{OR } (hasChild \textbf{ SOME } Lawyer)))$$

A description in Manchester OWL syntax cannot be considered a true natural language paraphrase of a concept, as its proximity to DL can in some cases produce results as unintelligible to the untrained user as the original DL syntax. If one wants to convert descriptions like (1) into natural language, one must be careful enough to avoid imprecisions. To this end, researchers have devised *controlled natural languages* [10], whose vocabularies and grammars are restricted in order to reduce ambiguity. An example of such a language is ACE (Attempto Controlled English) [4], in which one paraphrase of (1) would be[1]

> *Every A hasChilds something. No A hasChilds*
> *something that hasChilds something that is not a* (3)
> *Doctor or that hasChilds a Lawyer.*

---

[1]Translation produced by the OWL Verbalizer
(`http://attempto.ifi.uzh.ch/site/docs/verbalizing_owl_in_controlled_english.html`). A class named *A* was defined as equivalent to the description in (1).
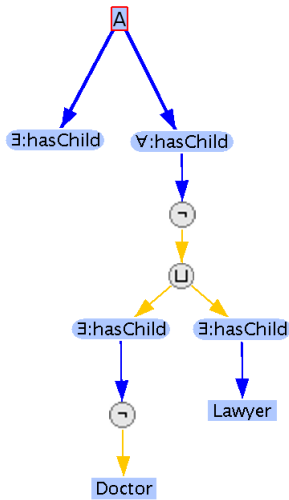
Figure 1: Visualization of description (1) produced by GrOWL.

Figure 2: Model outline for description (1)

tion. We follow the symbology established by the Protégé[2] user interfaces in representing individuals as dark-colored diamonds. When cardinality restrictions are not present, the number of individuals satisfying certain conditions is not important; therefore, so as not to mislead users into thinking that only one individual is allowed in a certain situation, we show a little cluster of diamonds.

The individuals in our model outlines are not named, since their identity is not important and since we do not want to clutter the display with unnecessary information. Instead, we may label a cluster of individuals with a concept description consisting of a conjunction of literals or of a disjunction of literals. An almost universally accepted graphical convention is that dashed lines and grayed-out regions invoke the idea of absence. We use this convention to indicate that an individual will satisfy (1) even if that individual has no grandchildren, a possibility that may be overlooked by someone with little or no knowledge of the semantics of DL.

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 suitable for diagrammatic systems [7]; Sect. 3 defines the precise semantics of model outlines, in the form of an algorithm that converts model outlines to $\mathcal{ALCN}$ concept descriptions[3]; Sect. 4 offers a simple example to illustrate the potential benefits of model outlines; and Sect. 5 contains a brief discussion.

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 the Manchester OWL syntax. In the grammar, 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 non-empty set Δ (the *universe* or *domain*) along with an interpretation $\mathcal{I}$ mapping each concept description C to a set $\mathcal{I}(C) \subseteq \Delta$, and each role term R to a binary relation $\mathcal{I}(R) \subseteq \Delta \times \Delta$. An interpretation $\mathcal{I}$ must map each description in the first column to the set given in the last 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* defines the physical representation of model outlines. What follows is an informal presentation: a model outline contains *clusters* (*solid* or *dashed*), *arrows*

Although more natural than (1) and (2), the nested relative clauses in (3) ("*that hasChilds something that is . . .*") may confuse the reader. The rendition of $\forall hasChild.\neg(\ldots)$ as "*No A hasChilds something that (. . .)*" is a good solution here, arguably more readable than "*Every A hasChilds something that not (. . .)*". A noticeable issue is the insertion, in the paraphrase, of class and property names (e.g., *hasChild*) that have been chosen by the author of the ontology according to criteria that do not take verbalization into account. The ACE translator assumes that all property names are verbs and handles such names accordingly, hence the occurrences of the awkward "*hasChilds*".

Alternatively, diagrammatic representations have been proposed, and implementations of ontology browsers have been based on these representations. However, many visualization frameworks for concept descriptions are too faithful to the *syntax* of the representation languages (e.g., DL, OWL, RDF), a feature which may prevent users from grasping the *semantics* of the concept descriptions. The visualization tool GrOWL [8], for example, will produce the abstract syntax tree in Figure 1 for the concept description in (1). While correct, such a diagram reflects only the syntax of the concept description. In order to understand the diagram, a user must be familiar with the notation of Description Logic and with the notion of syntax trees — more, in fact, than what is needed to understand the original concept description!

This paper presents *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. We hope this semantics-oriented visualization strategy will allow users to obtain deeper insights about the meaning of such descriptions, thereby preventing errors of design or of interpretation.

The model outline for the example description in (1) is shown in Figure 2. It depicts sets of individuals and the relations that must hold for them to satisfy the concept descrip-
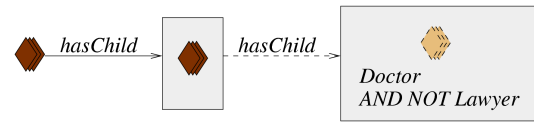
---

[2] http://protege.stanford.edu

[3] The *construction* of a model outline from a given concept description is not covered here. An algorithm for this task is presented in [3], along with further results.

	DL	Manchester OWL	Meaning
$C, D \rightarrow$	A	A	$\mathcal{I}(A)$
|	$\top$	**THING**	Δ
|	$\bot$	**NOTHING**	$\varnothing$
|	$\neg C$	**NOT** C	$\Delta - \mathcal{I}(C)$
|	$C \sqcap D$	C **AND** D	$\mathcal{I}(C) \cap \mathcal{I}(D)$
|	$C \sqcup D$	C **OR** D	$\mathcal{I}(C) \cup \mathcal{I}(D)$
|	$\forall R.C$	R **ONLY** C	$\{a \in \Delta \mid \forall b.[(a,b) \in \mathcal{I}(R) \Rightarrow b \in \mathcal{I}(C)]\}$
|	$\exists R.C$	R **SOME** C	$\{a \in \Delta \mid \exists b.[(a,b) \in \mathcal{I}(R) \wedge b \in \mathcal{I}(C)]\}$
|	$\leq n.R$	R **MAX** n	$\{a \in \Delta \mid \#\{b \mid (a,b) \in \mathcal{I}(R)\} \leq n\}$
|	$\geq n.R$	R **MIN** n	$\{a \in \Delta \mid \#\{b \mid (a,b) \in \mathcal{I}(R)\} \geq n\}$
|	$= n.R$	R **EXACTLY** n	$\{a \in \Delta \mid \#\{b \mid (a,b) \in \mathcal{I}(R)\} = n\}$

Figure 3: $\mathcal{ALCN}$ **concept descriptions and their meaning**

Figure 4: Example model outline

Figure 5: Case widgets

Figure 4 shows an example model outline. The target box of the arrow labeled "R" has both a class label and a cardinality label (which includes a degenerate number interval consisting of the number 5). The target box of the arrow labeled "S" has a cardinality label including an infinite interval starting at 4. The target box of the arrow labeled "U" has an "among-which" oval. This model outline does not have case widgets.

Case widgets indicate alternatives (i.e., disjunction). If a cluster or a box has a case widget above it, we show the user one case at a time. In Figure 5, e.g., there are 4 cases altogether, specifying individuals that either (a) are members of C and are only related through R to individuals belonging to D or to E (if any), or (b) are members of F and are related through R to at least one individual belonging to G or to H. More formally, Figure 5 represents the description $(C \sqcap \forall R.(D \sqcup E)) \sqcup (F \sqcap \exists R.(G \sqcup H))$.

As for the *abstract syntax*, a model outline is formally defined as a LISP-style list generated by the grammar in Figure 6, given in extended BNF notation. For example, the model outline in Figure 5 corresponds to the list

```
(cases (cluster solid (C)((arrow dashed (R)
    (cases (box ( )( )(cases (cluster dashed (D)( )))( )( ))
      (box ( )( )(cases (cluster dashed (E)( )))( )( ))))))
  (cluster solid (F)((arrow solid (R)(cases (box ( )( )
    (cases (cluster solid (G)( ))
      (cluster solid (H)( )))(cluster dashed ( )( ))( )))))))
```
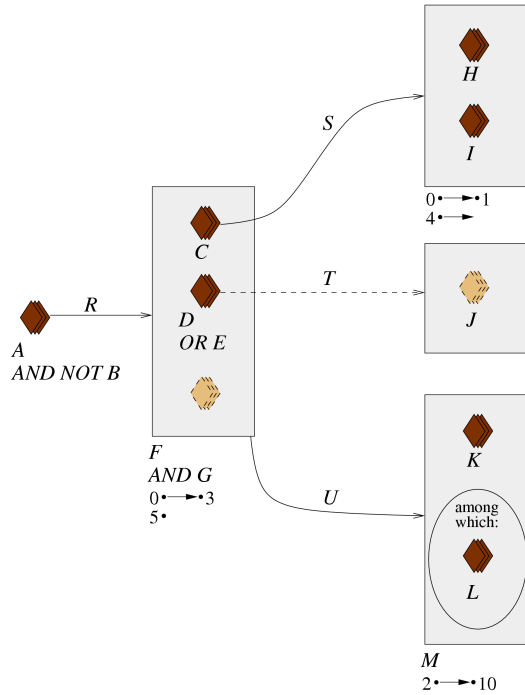
(*solid* or *dashed*) and *boxes*. The *root* of the model outline is a solid cluster. A cluster may have an optional *class label* below it, consisting of a disjunction or of a conjunction of literals. So may a box. A box may also have an optional *cardinality label* below it, consisting of a collection of *natural number intervals*. A natural number interval may contain a single number, in which case it is *degenerate*. A natural number interval may be *infinite*. 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" oval*, which in turn contains one or more clusters, all of them solid. A box or a cluster may have a *case widget* above it.
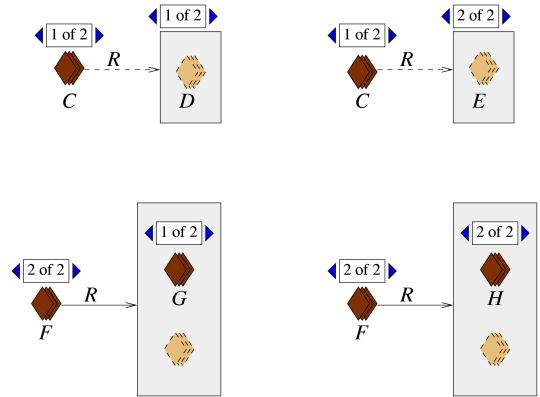
$$\langle outline \rangle \rightarrow \langle solidClusterCases \rangle$$
$$\langle solidClusterCases \rangle \rightarrow (\textbf{ cases } \langle solidCluster \rangle^{+})$$
$$\langle solidCluster \rangle \rightarrow (\textbf{ cluster solid } \langle classLabel \rangle (\langle arrow \rangle^{\star}))$$
$$\langle classLabel \rangle \rightarrow ( ) \mid (\langle literal \rangle) \mid (\textbf{ and } \langle literal \rangle \langle literal \rangle^{+}) \mid (\textbf{ or } \langle literal \rangle \langle literal \rangle^{+})$$
$$\langle literal \rangle \rightarrow \langle conceptName \rangle \mid (\textbf{ not } \langle conceptName \rangle)$$
$$\langle arrow \rangle \rightarrow (\textbf{ arrow solid } \langle arrowLabel \rangle \langle solidBoxCases \rangle)$$
$$\mid (\textbf{ arrow dashed } \langle arrowLabel \rangle \langle dashedBoxCases \rangle)$$
$$\langle arrowLabel \rangle \rightarrow (\langle roleName \rangle)$$
$$\langle solidBoxCases \rangle \rightarrow (\textbf{ cases } \langle solidBox \rangle^{+})$$
$$\langle solidBox \rangle \rightarrow (\textbf{ box } \langle classLabel \rangle (\langle interval \rangle^{\star}) (\langle cluster \rangle^{+}) \langle opt \rangle (\langle arrow \rangle^{\star}))$$
$$\langle interval \rangle \rightarrow (\langle number \rangle \langle number \rangle) \mid (\langle number \rangle \textbf{ infty })$$
$$\langle cluster \rangle \rightarrow \langle solidClusterCases \rangle$$
$$\langle opt \rangle \rightarrow ( ) \mid \langle unlabeledCluster \rangle \mid \langle oval \rangle$$
$$\langle unlabeledCluster \rangle \rightarrow (\textbf{ cluster dashed } ( ) ( ))$$
$$\langle oval \rangle \rightarrow (\textbf{ oval } \langle cluster \rangle^{+})$$
$$\langle dashedBoxCases \rangle \rightarrow (\textbf{ cases } \langle dashedBox \rangle^{+})$$
$$\langle dashedBox \rangle \rightarrow (\textbf{ box } \langle classLabel \rangle (\langle interval \rangle^{\star}) (\langle dashedCluster \rangle) ( ) (\langle arrow \rangle^{\star}))$$
$$\langle dashedCluster \rangle \rightarrow (\textbf{ cases (cluster dashed } \langle classLabel \rangle ( )))$$

Figure 6: Abstract, formal syntax for $\mathcal{ALCN}$ model outlines

Of course, the list representation is not meant for human consumption, but rather for automatic processing by algorithms such as the one presented in the next section.

3. SEMANTICS OF MODEL OUTLINES

We define the precise semantics of a model outline by means of the recursive algorithm DESCR, which, when given a cluster C, yields the $\mathcal{ALCN}$ concept description taken as the meaning of C. Applying the algorithm to the root cluster of a model outline yields the concept description denoted by the entire model outline. Algorithm DESCR calls algorithm BOXDESCR to build the concept description denoted by a box. Both algorithms are given in Figure 10, at the end of the paper.

The reader should refer to the grammar in Figure 6 for the structure of the lists that the algorithms manipulate. These algorithms may be made more efficient, both in their running time and in the legibility of the output concept description. Here, their only purpose is to serve as the precise semantics of model outlines. As an example, when given as input the model outline in Figure 2, algorithm DESCR returns the description $\bot \sqcup (\top \sqcap \exists hasChild.(\bot \sqcup (\forall hasChild.(\bot \sqcup (Doctor \sqcap \neg Lawyer)) \sqcap \top)))$.

4. A SIMPLE EXAMPLE

To show how model outlines may help prevent modeling errors, we compare three attempts to specify a Margherita pizza (see the discussion in [9]). The correct model outline and DL concept description are shown in Figure 7, meaning that a Margherita pizza is any pizza having exactly two toppings: mozzarella and tomato.

If the specifier forgets to include the closure axiom (i.e., $\forall hasTopping.(Mozzarella \sqcup Tomato)$), the resulting specification will allow additional toppings, something that is indicated by the presence of the unlabeled cluster in the model outline in Figure 8.

Now, if the existential assertions are forgotten instead, the dashed/grayed out objects in the resulting model outline (Figure 9) will alert the user to the fact that this specification can be vacuously satisfied by a pizza with no toppings, and the label "*Mozzarella OR Tomato*" will make it clear that a pizza having only one of those toppings still satisfies this specification.

Figure 7: Correct Margherita pizza specification

$Pizza \sqcap$
$\exists hasTopping.Mozzarella$
$\sqcap \exists hasTopping.Tomato$
$\sqcap \forall hasTopping.$
$(Mozzarella \sqcup Tomato)$
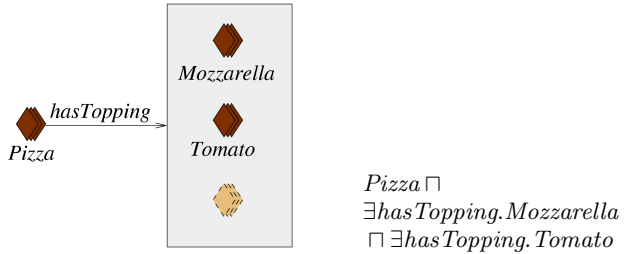
Figure 8: Incorrect Margherita pizza specification

$Pizza \sqcap$
$\exists hasTopping.Mozzarella$
$\sqcap \exists hasTopping.Tomato$
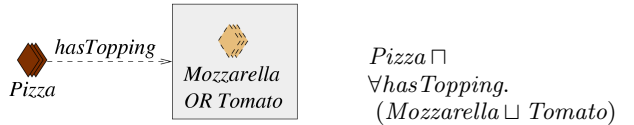
Figure 9: Incorrect Margherita pizza specification

$Pizza \sqcap$
$\forall hasTopping.$
$(Mozzarella \sqcup Tomato)$

For the sake of comparison with a natural-language based approach, we list below the ACE paraphrases of those three specifications, omitting the sentence "*Every MargheritaPizza is a Pizza*", common to all three paraphrases:

339

Fig. 7: *Every MargheritaPizza hasToppings a Tomato.*
Every MargheritaPizza hasToppings a Mozzarella.
No MargheritaPizza hasToppings something that is not
something that is a Mozzarella or that is a Tomato.

Fig. 8: *Every MargheritaPizza hasToppings a Tomato.*
Every MargheritaPizza hasToppings a Mozzarella.

Fig. 9: *No MargheritaPizza hasToppings something that is*
not something that is a Mozzarella or that is a Tomato.

5. DISCUSSION

Visual vs. verbal. We favor the use of visual frameworks because they are, in a certain sense, more fundamental than verbal representations. Indeed, in [5], the author says that

> "*visual languages may provide the bridge between the verbal expression of ideas and their logical explication. Perhaps not, but it is still a worthy objective. [. . .] We also need to facilitate the adoption and use of formal visual languages and their associated inference capabilities in scholarly communities that may have little knowledge, or sympathy with, formal logic, but might find the insights it can provide of value to their own endeavors.*"

In [2], the authors express their belief that

> "*many of the problems people have putting their knowledge of logic to work, whether in machines or in their own lives, stems from the logocentricity that has pervaded its study for the past hundred years.*"

However, logocentric and diagrammatic approaches to the representation of concepts are not mutually exclusive; in fact, since human beings are fluent in both modes of communication (verbal and pictorial), we believe these modes can complement each other to great benefit. When the aim is to convey structure (such as in concept descriptions with many nested quantified subdescriptions), diagrams are clear and concise, whereas textual accounts usually turn out to be verbose and hard to follow, with many nested relative clauses. In this respect, it is interesting to notice that one common technique to make these textual descriptions readable is to present them as a list of items, using levels of indentation to make the structure apparent; well, these indented lists are, in a way, an attempt at giving a diagrammatic flavor to the text! On the other hand, simple situations (e.g., "a person that only has doctors and non-lawyers as grandchildren") may be easier to understand as natural language paraphrases. This suggests that our model outlines should be enhanced with textual descriptions: when the user hovers the mouse over a visual item (a box or a cluster), a tooltip may appear with a natural language paraphrase of the (local) constraints that the corresponding individual or set of individuals must satisfy. One advantage of this combination is that, along time, the coexistence of text and diagram will contribute to reinforce the user's understanding of our diagrammatic notation and of the peculiarities of complex textual descriptions.

Diagrammatic reasoning and explanation. Because this paper does not focus on the *construction* of model outlines, it has not emphasized the role of (logical) simplification. In building the model outline of Figure 2 for the description in (1), a suitable set of rewrite rules has been applied. If the user so desires, he may request to view the application of such rules in a dynamic, step-by-step fashion (e.g., through animation). This constitutes a form of diagrammatic reasoning. We believe that, with suitable presentation strategies, the application of diagrammatic rewrite rules can be rather intuitive, even for nonspecialist users. This indicates that model outlines could be useful in the context of *proof explanation* (besides concept construction and visualization), a possibility that we are currently investigating.

6. REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2nd edition, 2007.

[2] J. Barwise and J. Etchemendy. Heterogeneous logic. In J. Glasgow, N. H. Narayanan, and B. Chandrasekaran, editors, *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, pages 211–234. MIT Press, Cambridge, UK, 1995.

[3] F. N. do Amaral and C. Bazílio. Visualization of Description Logic models. In *The 21st International Workshop on Description Logics (DL2008), Dresden, Germany*, 2008.

[4] N. E. Fuchs, K. Kaljurand, and G. Schneider. Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In *FLAIRS 2006*, 2006.

[5] B. R. Gaines. Designing visual languages for Description Logics. *Journal of Logic, Language and Information*, 2008. To appear.

[6] M. Horridge, N. Drummond, J. Goodwin, A. Rector, R. Stevens, and H. Wan. The Manchester OWL syntax. In *OWL: Experiences and Directions*, 2006.

[7] J. Howse, F. Molina, S.-J. Shin, and J. Taylor. On diagram tokens and types. In *DIAGRAMS '02: Proceedings of the Second International Conference on Diagrammatic Representation and Inference*, pages 146–160, London, UK, 2002. Springer-Verlag.

[8] S. Krivov, R. Williams, and F. Villa. GrOWL: A tool for visualization and editing of OWL ontologies. *Journal of Web Semantics*, 5(2):54–57, 2007.

[9] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In *Engineering Knowledge in the Age of the SemanticWeb*, volume 3257 of *LNCS*, 2004.

[10] R. Schwitter, K. Kaljurand, A. Cregan, C. Dolbear, and G. Hart. A Comparison of three Controlled Natural Languages for OWL 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC)*, Washington, 2008.

DESCR(C)

    ▷ C is of the form (**cases** $C_1 \cdots C_m$)
1 $Descr \leftarrow \bot$
2 **for** each C_i in $C_1, \ldots, C_m$
    ▷ C_i is of the form (**cluster solid** L ($A_1 \cdots A_n$))
3   **do if** $L = ($ $)$
4   **then** $Case \leftarrow \top$
5   **else** $Case \leftarrow L$
6   **for** each A_j in $A_1, \ldots, A_n$
7   **do** $Case \leftarrow Case \sqcap \text{BoxDescr}(A_j)$
8         $Descr \leftarrow Descr \sqcup Case$
9 **return** $Descr$

BOXDESCR(A)

    ▷ A is of the form (**arrow** $S\,AL$ (**cases** $B_1 \cdots B_m$))
1 $BDescr \leftarrow \bot$
2 **for** each B_i in $B_1 \cdots B_m$
    ▷ B_i is of the form (**box** BL ($I_1 \cdots I_n$) ($C_1 \cdots C_p$) Opt ($A'_1 \cdots A'_q$))
3   **do** $Universal \leftarrow \bot$; $Existentials \leftarrow \top$
4   **for** each C_j in $C_1, \ldots, C_p$
5   **do** $Universal \leftarrow Universal \sqcup \text{DESCR}(C_j)$
6   **if** C_j is of the form (**cluster solid** $\ldots$)
7   **then** $Existentials \leftarrow Existentials \sqcap \exists AL.\text{DESCR}(C_j)$
8        $Universal \leftarrow \forall AL.(Universal)$
9   **if** Opt is of the form (**oval** $C'_1 \cdots C'_r$)
10   **then for** each C'_j in $C'_1, \ldots, C'_r$
11   **do** $Existentials \leftarrow Existentials \sqcap \exists AL.\text{DESCR}(C'_j)$
12   **if** $BL = ($ $)$
13   **then** $BCase \leftarrow Universal \sqcap Existentials$
14   **else** $BCase \leftarrow BL \sqcap Universal \sqcap Existentials$
15   **if** $n > 0$
16   **then** $Card \leftarrow \bot$
17   **for** each I_j in $I_1, \ldots, I_n$
            ▷ I_j is of the form ($X\,Y$)
18   **do if** $Y = \mathbf{infty}$
19   **then** $Card \leftarrow (Card \sqcup \,\geq X.AL)$
20   **else** $Card \leftarrow (Card \sqcup (\geq X.AL \sqcap \,\leq Y.AL))$
21         $BCase \leftarrow BCase \sqcap Card$
22   **for** each A'_j in $A'_1, \ldots, A'_q$
23   **do** $BCase \leftarrow BCase \sqcap \forall AL.\text{BoxDescr}(A'_j)$
24       $BDescr \leftarrow BDescr \sqcup BCase$
25 **return** $BDescr$

Figure 10: Algorithms to produce a concept description for a cluster C