Dually Structured Concepts in the Semantic Web: Answer Set Programming Approach

Patryk Burek and Rafał Graboś*

Department of Computer Science, University of Leipzig, Germany
{Burek, grabos}@informatik.uni-leipzig.de

Abstract. There is an ongoing discussion whether reasoning in the Semantic Web should be monotonic or not. However, it seems that the problem concerns not only reasoning over knowledge but knowledge itself, where apart from nondefeasible knowledge the defeasible knowledge can be distinguished. In the current paper we rely on the Dual Theory of Concepts, according to which concepts are dually structured into defeasible and nondefeasible parts. We develop a metaontology for representing both types of a concept's structure and apply it for annotating OWL axioms. The translation of annotated OWL axioms into a logic program under answer set semantics is provided. Hence the answer set solver Smodels may be used as reasoner for annotated ontologies, handling properly the distinction between monotonic and nonmonotonic reasoning.

Keywords: Ontology, Knowledge Representation, Reasoning in the Semantic Web, Semantic Web Inference Schemes

1 Introduction

Current standards and techniques used in the ontology and Semantic Web communities, like the Web Ontology Language (OWL) [25] and its underlying description logics (DL) with their reasoners, are basically monotonic. On the other hand there is an ongoing discussion whether reasoning in the Semantic Web should be monotonic or not. It seems however that the problem concerns not only the reasoning over the knowledge but the knowledge itself. The knowledge may be defeasible and as such may require nonmonotonic inferences on it. The main question then is what the knowledge is like and whether it is defeasible.

Since Wittgenstein [30] and Rosch [28] there is a debate in cognitive science concerning the prototypical structure of concepts. Both Wittgenstein and Rosch pointed out that concepts do not have to be specified by a set of necessary (and sufficient) conditions. They argued that often not all of a concept's referents share all of the properties assigned to the concept. Prototypically structured concepts then lack necessary characteristics but have only typical and therefore defeasible characteristics. On the other hand a prototypical concept structure raises several problems. For example it

* This paper was written fully collaboratively; the order of the authors' names is arbitrary.

A. Gómez-Pérez and J. Euzenat (Eds.): ESWC 2005, LNCS 3532, pp. 377–391, 2005.

is clear that not all concepts are fuzzy and prototypically structured but there are some that have at least necessary characteristics.

In the current paper we propose a framework for specifying concepts within ontologies based on the cognitive Dual Theory, according to which concepts are dually structured involving defeasible and nondefeasible parts.

To represent the structure of concepts and to delimit a defeasible part of a concept's definition from a nondefeasible one, we develop a simple metaonotology and use it for annotating OWL axioms. The translation of annotated OWL axioms into a logic program under answer set semantics is provided. Hence the answer set solver Smodels [22] may be used as reasoner for annotated ontologies, handling the distinction between monotonic and nonmonotonic reasoning properly.

For simplicity of reading, we do not represent annotated OWL axioms in OWL abstract syntax [25]. Instead we use description logic syntax extended by the provided meta-tags. This is justified since there is a correspondence between the syntaxes of OWL and description logics, such that each element of OWL can be represented by elements of the corresponding description logic[1]. However, one should note that the description logic syntax is used here for readability purpose only.

The paper is structured as follows. In the next section we introduce a running example. In the following two sections advantages and drawbacks of nonmonotonic solutions to the Semantic Web are presented. In the fifth section we present the foundations of our approach based on the cognitive Dual Theory. Moreover, in the fifth section the metaonology and the mechanism of annotating OWL axioms are presented. In the sixth section the translation algorithm of annotated OWL axioms into logic programming rules is introduced and illustrated by our running example. In the last two sections related work and conclusions are presented.

2 Example

Let us consider a hypothetical definition of desert. Desert could be defined as an area with little precipitation (the average amount of precipitation is less than 25 centimeters a year), partly covered by sand or gravel, having scanty vegetation or sometimes almost none, and capable of supporting only a limited and specially adapted animal population. The definition could be formally written in DL using the following form:

$$\texttt{Desert} \sqsubseteq \texttt{Area} \sqcap \exists \texttt{coveredby.(oneof\{Sand, Gravel\})} \sqcap \forall \texttt{precipitation.Low} \sqcap \forall \texttt{vegetation.Scanty} \sqcap \forall \texttt{populatedby.AdaptedAnimal}$$

The typical criticism raised by representatives of Prototype Theory here could be such, that there are deserts that do not fulfill all the conditions of the above definition but which nevertheless are treated as good representatives of deserts. Consider for example cold deserts covered rather with ice than sand. Moreover, in many deserts there are regions where the vegetation is not sparse at all.

If we would like to include exceptions like those above to a knowledge base containing our definition of desert, clearly we will end up with inconsistencies. Hence we

[1] For instance description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ underlie OWL Lite and OWL DL respectively.

see that there is a need to adopt less restrictive forms of knowledge representation that include prototypes and permit nonmonotonic reasoning.

3 Nonmonotonic Reasoning

Long since it has been recognized in AI that classical logic is not sufficiently robust to adequately reason as humans do. The main feature of classical logic (as well as DLs) is monotonicity, in the sense that one cannot reject conclusions by adding new premises. However, people tend to retract previous conclusions, when new evidences appear. This is called nonmonotonic reasoning, which's subject matter is developing reasoning systems, that model the way in which commonsense is used by humans.

Formally, a consequence relation $\models$ is nonmonotonic if there is a formula β and a set of formulas α such that $\alpha \models \beta$ holds, then $\alpha, \gamma \not\models \beta$ may hold when new information γ appears (where γ is a formula). Brewka pointed out in [8] that nonmonotonic reasoning, in its broadest sense, is reasoning to conclusions on the basis of incomplete information. In this sense *default* conclusions are accepted until new information arises.

Nonmonotonicity may be seen as a property of knowledge and property of reasoning over knowledge. We accent here the former view and suggest that first of all the knowledge is defeasible or not defeasible. Knowledge is defeasible, if it may be invalidated in the light of certain additional information, called defeaters.

Reasoning in the Semantic Web is usually monotonic. The formal semantics of Semantic Web languages like OWL DL is given by corresponding Description Logics [3], which are mainly monotonic. This means that there is no way for retracting any conclusions in the light of new information. This may cause knowledge on the Web to get inconsistent as new knowledge is added. In addition, it has been recognized in AI that a knowledge system should allow a construction of elaboration tolerant knowledge bases, i.e. bases in which small modification of the informal body of knowledge corresponds to small modifications of the formal base representing this knowledge [13]. Nonmonotonicity helps to satisfy this requirement.

In the current paper we propose answer set semantics to be added to the Semantic Web language OWL and useage of answer set solvers (such as Smodels[22]) as reasoners. To the best of our knowledge, only the work of Bertino [5] considers a similar idea, namely encoding the Semantic Web representation of knowledge by a logic program under answer set semantics[2] and thus allowing default reasoning. The mentioned approach consists in providing a new default semantics to RDF type inheritance primitives, in particular to DAML+OIL properties daml:Class and daml:subClassOf. Then DAML+OIL sentences have translations into a logic program. Answer set semantics to basic DAML+OIL constructor relations is provided and a direct translation of RDF statements into logic facts is done. At the same time the answer sets of the program are logically equivalent to the RDF statements inferred from the given KB. The main advantage of such a method is that ASP semantics permits any conclusion thrown by default to be dropped if contrary explicit knowledge is found. Since it

[2] Some authors propose a translation from DLs to logic programming under answer set semantics. An overview is given in a section 7 concerning the related work.

makes the conclusions defeasible, nonmonotonicity has been successfully adopted to DAML+OIL. However, we show in the next section that the above solution sometimes leads to unintuitive results.

4 Against Absolute Nonmonotonicity

Nonomonotonic solutions and prototype theory in general do not remain without problems. In [6], [7] problems are reported, which arise by the adoption of defaults to frame systems. For example, the cancellation and overriding of a concept's features results in mishmash of subsumption trees and disturbs the classification algorithm. If all properties of the concept can be canceled then the concept may be subsumed by anything, which gives rise to absurd cases like the one reported by Brachman: "a rock is an elephant except it has no trunk, it isn't alive, it has no legs… "[7].

Although some, or even most of our knowledge permits exceptions, there is also knowledge that does not. In other words not all concepts have prototypical structure, but there are some that per se exclude any form of cancellation of their properties. Consider for instance mathematical concepts. They are not prototypically structured, but instead they have provided at least necessary conditions and therefore no exceptions are permissible. Each triangle must have exactly three sides and therefore no atypical, not three sided triangles are allowed. Reasoning about such not prototypically structured concepts should remain monotonic.

Thus, for at least two above reasons the solutions like the one of Bertino [5], interpreting all of our knowledge as defeasible, seem to be problematic. Not in all cases daml:Class and daml:subClassOf require nonmonotonic interpretation. In the next section we present a different approach that tries to overcome the above problems but still permits defaults.

5 Dual Theory Approach

In [23] Osherson and Smith proposed the Dual Theory, which is a hybrid approach to the prototype problem. They found that the application of the Zadeh's fuzzy set theory [32] for representing prototypes leads to several problems while representing conjunctive concepts as well as logically empty and logically universal concepts. These problems of fuzzy-set theory, and ubiquitous prototypical categories forced Osherson and Smith to compose prototypes with definitions of necessary (and sufficient) conditions.

According to the Dual Theory, concepts have a binary structure and are composed of two types of information [20], [21]. One of these has a prototypical character and another that does not. The prototypes are considered to be used in an identification procedure of a concept's membership and are responsible for rapid decisions about concept's membership. Properties that form the prototype of a concept we will call here *typical* since they are those, which typically are considered to identify a concept's instances. But since not all of a concept's instances share such typical features, we treat typical features as being defeasible, in the sense that they ought to be assumed in the absence of any contrary information [26].

The second type of information constituting concepts consists of, as called here, *core* properties. Core properties are not defeasible and they provide the truth conditions to hold for concept membership. They are used for combinations of concepts and are responsible for our most considered categorization judgments [23]. Typical properties then may not fully determine the concept instances, but this may be done by the concept's core. Secondly not all concepts must have a prototypical part (for example mathematical concepts). The identification procedure in those cases is fully held by the core of the concept.

Considering our desert example in the context of the Dual Theory we see that the properties of being covered with sand and having scanty vegetation are the typical properties of a desert. Usually when meeting a region satisfying these conditions we recognize it as being a desert. Those features form the prototype of a desert but they do not fully determine the concept's instances. As mentioned above, there are deserts that do not fulfill those conditions. In such cases we need to refer to more subtle conditions to decide about concept's membership. In this case it could be the feature of having a low average amount of precipitation. That feature is not defeasible and it constitutes the core of the concept of desert.

Typical features are defeasible while core features are not. If in a knowledge base inconsistency concerning typical features appears, it should be treated differently from inconsistencies concerning core features. In the first case typical feature should be canceled so that the inconsistency is avoided, but in the second case the cancellation should not take place. As an example, let us consider a stony desert. Although assuming that a stony desert is not covered by sand it is still considered to be a desert. A classification of a stony desert under the concept desert would result in inconsistency of the knowledge base. However, since being covered by sand is not a core property but only a typical one, it may be canceled and this enables us to place stony deserts in the hierarchy under the concept of desert without falling into inconsistency. The same does not hold for the property of having low precipitation which is considered to be a core property. A region not having a low precipitation, even if covered by sand and having scanty vegetation should not be classified as desert without leading to inconsistency.

The typical features are considered in the Dual Theory to be identification features of a concept. This means that from the features an individual has, one can infer under what concept the individual falls. To enable the inference in this direction, the implication from features to a concept is needed. Features should then not be necessary but sufficient conditions for concept membership, which does not hold in our example, however. In our case we can infer from a concept to its features, which means that features are considered as necessary conditions for concept membership.

5.1 Annotation of Statements

To make it explicit which part of the concept definition is core and which is only typical we use a simple metaontology consisting of two disjoint classes *typical* and *core*. Corresponding tags `[core]` and `[typical]` are used to annotate OWL axioms. These tags are assigned neither to OWL classes or properties directly nor to whole OWL axioms. The class or property is not core or typical in general but only in the context of some axiom. For example a property of being covered by sand is typi-

cal in case of a desert but may be core for a sandy beach. On the other hand not the entire axiom must be typical or core but only its parts, here called characteristics. The tags are assigned to the rdfs:subClassOf construct[3], thus to RDF statements, where the resource rdfs:subClassOf is a predicate. We treat each such reified RDF statement as an instance of one of the two classes – core and typical. A reified statement is an instance of the class *core* iff the statement must always be satisfied for the given concept. A reified statement is typical iff it is satisfied for the concept, unless there is information to the contrary.

To illustrate this we can assign the tags to each statement of the exemplar definition of desert. We represent it in semi-description logics notation in the following way:

```
Desert ⊑ [core]Area ⊓
           [typical]∃coveredby.(oneof{Sand, Gravel}) ⊓
           [core]∀precipitation.Low  ⊓
           [typical]∀vegetation.Scanty ⊓
           [typical]∀populatedby.AdaptedAnimal
```

From the above we can read that the characteristics saying that desert is an area having low precipitation are core, whereas all others are only typical. The above definition now says that it may happen that there is a desert not covered by sand or gravel but it may not happen that a desert is not a subclass of area.

To assign metatags to OWL ontologies we use reified RDF statements and treat them as instances of one of the given metaclasses. For example in OWL abstract syntax the first tagged statement of the above definition looks as follows:

```
Individual(desertStatement1
  type(rdf:statement)
  type(metaontology:core)
  value(rdf:object Area)
  value(rdf:predicate rdfs:subClassOf)
  value(rdf:subject Desert))
```

`Metaontology` refers to the metaontology, where the classes: *core* and *typical* are defined.

6 Answer Set Programming for the Semantic Web

In this section the translation of annotated OWL axioms into a logic program under answer set semantics is provided and the answer set solver Smodels [22] is used to find a model for the given KB. For simplicity of reading we use a semi-description logic notation instead of the OWL abstract syntax.

[3] In the current paper we consider only partial definitions, but the approach can be extended to complete definitions as well. In that case also owl:equivalentClass should be tagged.

6.1 Answer Set Programming

Answer set programming (ASP) is a new logic programming paradigm for knowledge representation and problem solving in artificial intelligence [19]. Representation of a problem is purely declarative, suitable for many aspects of commonsense reasoning (diagnosis, configuration, planning etc.). Instead of a query technique (Prolog), it bases upon a notion of possible solution, called *answer set*.

Consider a propositional language L, with atomic symbols called atoms. A literal is an atom or a negated atom (by classical negation ¬). The symbol *not* is called epistemic negation and the expression *not a* is true, if there is no reason to believe that *a* is the case. Epistemic negation makes ASP a nonmonotonic system. Default rules, whose conclusions are defeasible in the light of certain knowledge, are represented using epistemic negation. Knowledge which is not defeasible is represented by means of the rules without negation *not*. The symbol ∨ is called epistemic disjunction and it is interpreted as follows: at least one literal is believed to be true. Formally, a rule *r* is an expression of the form:

$$c_1 \vee \ldots \vee c_k \leftarrow a_1, \ldots, a_m, \textit{not } b_{m+1}, \ldots, \textit{not } b_n \tag{1}$$

where $k \geq 0$, $n \geq m \geq 0$, c_i, a_l, b_k are ground literals *r*, $Body^{-}(r) = \{b_{m+1}, \ldots, b_n\}$, $Body^{+}(r) = \{a_1, \ldots, a_m\}$ and the disjunction $\{c_1 \vee \ldots \vee c_k\}$ is *Head*(r) of the rule *r*. A rule with an empty *Head* (i.e. a rule of the form: $\leftarrow Body$) is usually referred to as integrity constraint. A logic program is a finite set of the rules.

Intuitively the above rule *r* means that if $Body^{+}(r)$ of that rule is believed to be true and it is not the case that $Body^{-}(r)$ is believed to be true, then at least one literal of *Head*(r) must be believed to be true.

The semantics for ASP is defined by means of minimal set of literals satisfying all rules of the program P. Let us now assume now, that Lit_P is the set of all literals being present in the extended logic program P and *I* is an interpretation of P, $I \subseteq Lit_P$. We say that a set of literals *I* satisfies a rule of the form (1), if $\{a_1, \ldots, a_m\} \subseteq S$ and $\{b_{m+1}, \ldots, b_n\} \cap S = \emptyset$ imply that $\{c_1, \ldots, c_k\} \cap S \neq \emptyset$.

The Gelfond-Lifschitz (GL) transformation of P with respect to *I* is a positive logic program P′ which is obtained in two steps [19]:

- deletion of all rules *r* of P, for which $Body^{-}(r) \cap I \neq \emptyset$
- deletion of the negative bodies ($Body^{-}(r)$) from the remaining rules of P

Then, *I* is an *answer set* of the logic program P, if *I* is a minimal model (no proper subset of *I* is a model of P′) of the positive (without *not*) logic program P′; i.e. *I* is a minimal set of literals satisfying every rule in P' or if *I* contains a pair of complementary literals l and ¬l, then $I = Lit_P$.

Example. Consider the program $P_1 = \{b \leftarrow \text{not } a;\ a \leftarrow \text{not } b;\ f \leftarrow a\}$, *Lit* = (a, b, f), and let $I = \{a, f\}$. The GL reduction of the program P_1 w. r. t. *I* is the program $P_1' = \{a \leftarrow;\ f \leftarrow a\}$. According to the definition, *I* is an answer set for the program P, since *I* is a minimal set of literals satisfying all rules in P_1'. The second answer set of the program P is {b} and these are the only answer sets for this program.

In general, programs under answer set semantics describe a family of intended models. Therefore they encode possible solutions to a problem, being a constraint sat-

isfaction problem, and described by the program, where each rule is interpreted as a constraint.

Although answer set programs are basically propositional, it is possible to use a rule schemata containing variables. These schemata are representations of their ground instances, and answer set solvers [11], [22] use intelligent ground instantiation techniques before the actual answer set computation takes place[4]. In this case logic programs with variables may be used to represent more complex problems. We will use such a technique as a convenient representation of our running example.

6.2 Translation OWL to ASP

We translate the ontology represented in annotated OWL to a logic program under answer set semantics and use the answer set solver as reasoning tool. To handle defeasibility of typical knowledge, we use ASP rules with epistemic negation in the *body* for encoding default information. In the light of certain information being defeaters, the conclusions assumed to be defeasible may be retracted. Knowledge, which is not defeasible, i.e. the *core* properties in our case, is represented in a form of positive rules not allowing exceptions.

Any inconsistency w.r.t. the nondefeasible knowledge leads to the inconsistency of the whole KB indicating the fact that such conclusions may not be retracted. By that means nonmonotonic reasoning is enabled, but on the other hand the deletion of all conclusions, in opposite to [5], is avoided.

In general, we base on the work of Baral [1] who successfully translates DL $\mathcal{ALCQI}$ into declarative logic programming under answer set semantics. Our algorithm provides a translation of annotated RDF statements into suitable logical rules. Since, in case of the *core* properties of concepts the standard DL semantics is appropriate, the translation given in [1], except for step 6 can be used. Therefore the inference over the *core* knowledge is strictly monotonic. We are mainly interested in handling the default part of knowledge; therefore we show how statements annotated by the tag [typical] can be represented in ASP.

Below the general schema of the translation of default knowledge into corresponding logic program rules is given. In detail, we encode one type of formulae, namely default subsumption between an atomic concept and a formula. Instead of the standard OWL semantics, default ASP semantics is provided to such subsumption. The formula encoded in the semi DL notation: $C_1 \sqsubseteq$ [typical] C_2 has the following intuitive meaning: concept C_1 by default subsumes C_2 where C_2 may be more a complex formulae. In order to determine the defeaters (knowledge capable to cancel the default conclusions), for the typical subsumption, we use for each default rule a predicate ab_k where k is a unique index. The precise algorithm is provided in the current section.

For all steps, C_1 denotes an atomic concept, while C_2 corresponds to an arbitrary complex formula. The Herbrand universe (HU), which is a set of ground terms constructed from the constants and functions in the program, is divided into disjoint sets of sub-domains. This way the search space for a solver is restricted significantly,

[4] The answer set semantics of ground programs can be extended straightforwardly to programs with variables by employing the notion of Herbrand models.

since smaller the Herbrand base (HB, being the set of atomic ground formulae built form the HU and the predicate symbols of the program) is considered.

Since the translation of atomic concepts, roles and individuals is provided in [1] (Steps 1-5), we skip it in the algorithm presented below and demonstrate it only in the context of our running example.

Steps 1-5 remain unchanged as given in [1] with the exception that we divide HU (hence as well HB) into domains, containing disjoint elements.

Step 6.

C_2 is an atomic concept

$C_1 \sqsubseteq$ [typical]C_2:

$\leftarrow c_1(X)$, not $c_2(X)$, not $ab_1(X)$.

$ab_1(X) \leftarrow c_1(X), \neg c_2(X)$.

Step 7.

a) C_2 is of the form: $\neg C_3$:

$c_2(X) \leftarrow$ domain(X), not $c_3(X)$.

$C_1 \sqsubseteq$ [typical]C_2:

$\leftarrow c_1(X)$, not $c_2(X)$, not $ab_2(X)$.

$ab_2(X) \leftarrow c_1(X), \neg c_2(X)$.

b) C_2 is of the form $C_3 \sqcap C_4$:

$C_1 \sqsubseteq$ [typical]C_3

$C_1 \sqsubseteq$ [typical]C_4

and step 6

c) C_2 is of the form: $C_3 \sqcup C_4$:

$c_2(X) \leftarrow$ domain(X), $c_3(X)$.

$c_2(X) \leftarrow$ domain(X), $c_4(X)$.

$C_1 \sqsubseteq$ [typical]C_2:

$\leftarrow c_1(X)$, not $c_2(X)$, not $ab_3(X)$.

$ab_3(X) \leftarrow c_1(X), \neg c_3(X), \neg c_4(X)$.

d) C_2 is of the form: $\forall R.C_3$:

not_$c_2(X) \leftarrow$ domain(X), domain(Y), r(X,Y), not $c_3(Y)$.

$c_2(X) \leftarrow$ domain(X), domain(Y), not not_$c_2(Y)$, r(X,Y).

$C_1 \sqsubseteq$ [typical] C_2:

$\leftarrow c_1(X)$, not $c_2(X)$, not $ab_4(X)$.

$ab_4(X) \leftarrow c_1(X)$, r(X, Y), $\neg c_3(Y)$.

e) C_2 is of the form: $\exists R.C_3$:

$c_2(X) \leftarrow r(X,Y), c_3(Y)$.

$C_1 \sqsubseteq$ [typical]C_2:

$\leftarrow c_1(X)$, not $c_2(X)$, not $ab_4(X)$.

$ab_4(X) \leftarrow c_1(X)$, r(X,Y), $\neg c_3(Y)$.

f) C_2 is of the form: $\exists R^{\geq n}.C_3$:

$c_2(X) \leftarrow r(X,Y_1),\ldots, r(X,Y_n), c_3(Y_1),\ldots,c_3(Y_n), Y_1 \neq Y_2 \neq \ldots \neq Y_n$.

$C_1 \sqsubseteq$ [typical]C_2:

$\leftarrow c_1(X)$, not $c_2(X)$, not $ab_5(X)$.

$ab_5(X) \leftarrow$ not $c_2(X)$, $c_1(X)$.

g) C_2 is of the form: $\exists R^{\leq n}.C_3$:
$not\_c_2(X) \leftarrow r(X,Y_1),\ldots, r(X,Y_{n+1}), c_3(Y_1),\ldots,c_3(Y_{n+1}), Y_1 \neq Y_2 \neq \ldots \neq Y_{n+1}$.
$c_2(X) \leftarrow not\ not\_c_2(X), domain(X)$.
$C_1 \sqsubseteq [typical]C_2$:
$\leftarrow c_1(X), not\ c_2(X), not\ ab_6(X)$.
$ab_6(X) \leftarrow not\_c_2(X), c_1(X)$.
h) C_2 is of the form: oneOf $\{a_1,\ldots, a_n\}$
$c_2(X) \leftarrow domain(X), X=a_1$.
…
$c_2(X) \leftarrow domain(X), X=a_n$.
$C_1 \sqsubseteq [typical]C_2$:
$\leftarrow c_1(X), not\ c_2(X), not\ ab_7(X)$.
$ab_7(X) \leftarrow not\ c_2(X), c_1(X)$.

In order to explain the above algorithm, we consider our running example and show how it can be translated into a logic program under answer set semantics. Then the answer set solver Smodels [22] is used to compute answer sets of the logic program such that models of the program correspond to the conclusions of the given KB. Note that we consider the Herbrand universe only, thus models are restricted to Herbrand models only (subsets of the Herbrand base). Since our approach is nonmonotonic, multiple models of a program may exist. We distinguish between brave and cautious reasoning as follows:

- **Brave reasoning:** given a logic program P and a ground literal *l*, decide whether *l* is true in some answer set of P (denoted $P \models_b l$)
- **Cautious reasoning:** given a logic program P and a ground literal *l*, decide whether *l* is true in all answer sets of P (denoted $P \models_c l$)

Note that to decide whether *l* is true in answer set S means to check whether *l* belongs to answer set S. Thus, literals belonging to all answer sets may be called the cautious conclusions of the given KB (under HB), while literals belonging to some answer sets are called brave default conclusions of the given KB (under HB). Therefore, the cautious conclusions of a logic program P are consequences of the KB represented by P under the Herbrand base, while the brave conclusions of P are *possible* consequences of the KB represented by P, obtained under incomplete knowledge.

It is clear that two features of desert are *core* properties: *area* and *precipitation. low*. Since they are not defeasible knowledge, we represent them in a form of not defeasible logic program rules, as showed in [1]. The remaining properties can be seen as the typical properties of desert, thus they are encoded as default rules by means of the method given above. Each atomic concept, role and fact assertion as well as elements of HU are encoded as logical rules and domain predicates respectively. Let us assume that HU = {sturt, stone, gravel, sand, few_cactus, twenty_cm, camel}. We use here Smodel's syntax to encode the above example.

Step 1. Elements of HU are encoded as facts belonging to the domains:

```
d_cover(gravel;sand;stone). d_desert(sturt). d_animal(camel).
d_vegetation(few_cactus). d_prec_level(twenty_cm).
```

Step 2. Atomic concepts are represented by the following rules:

```
area(X)  :- d_desert(X), not not_area(X).
not_area(X) :- d_desert(X), not area(X).
low(X) :- d_prec_level(X), not high(X).
high(X) :- d_prec_level(X), not low(X).
scanty(X) :- d_vegetation(X), not not_scanty(X).
not_scanty(X) :-d_vegetation(X), not scanty(X).
adopted_animal(X) :- d_animal(X), not not_adopted_animal(X).
not_adopted_animal(X) :- d_animal(X), not adopted_animal(X).
```

Step 3. Atomic roles are represented by the following rules:

```
coveredby(X,Y) :- d_desert(X), d_cover(Y), not not_coveredby(X,Y).
not_coveredby(X,Y) :- d_desert(X), d_cover(Y), not coveredby(X,Y).
precipitation(X,Y)  :- d_desert(X),d_prec_level(Y), not
not_precipitation(X,Y).
not_precipitation(X,Y) :-  d_desert(X),d_prec_level(Y), not precipi-
tation(X,Y).
vegetation (X,Y) :- d_desert(X),  d_vegetation(Y), not
not_vegetation(X,Y).
not_vegetation (X,Y) :- d_desert(X), d_vegetation(Y), not
vegetation(X,Y).
populatedby (X,Y) :- d_desert(X),  d_animal(Y), not
not_populatedby(X,Y).
not_populatedby (X,Y) :- d_desert(X),  d_animal(Y), not
populatedby(X,Y).
```

Step 4. Fact assertions are encoded as logic programming facts:

```
desert(sturt). low(twenty_cm).
```

Step 5. Each role assertion is translated to the facts:

```
coveredby(sturt,stone).
```

Step 7. A subsumption, where a super concept is a complex concept in form of a conjunction, is split first into the conjunction of subsumptions (step 7 (b) of our procedure):

(I) desert $\sqsubseteq$ [core] area

(II) desert $\sqsubseteq$ [typical] $\exists$coveredby.(oneof{Sand, Gravel})

(III) desert $\sqsubseteq$ [core] $\forall$precipitation.Low

(IV) desert $\sqsubseteq$ [typical] $\forall$vegetation.Scanty

(V) desert $\sqsubseteq$ [typical] $\forall$populatedby.AdoptedAnimal

Since a *core* subsumption corresponds to the semantics of standard DL subsumption, we translate it according to the method given in [1]. Then:

(I) triggers translation of the core subsumption:

```
:- desert(X), not area(X).
```

(II) appeals first step 7 (e):

```
ok_coveredby(X) :- coveredby(X,Y), oneOf(Y), d_desert(X).
```

which triggers step 7 (h):

```
oneOf(X) :- d_cover(X), X=sand.
oneOf(X) :- d_cover(X), X=gravel.
```

and step 7 (e) for default subsumption:

```
:- desert(X), not ok_coveredby(X), not ab_2(X).
ab_2(X) :-  desert(X), coveredby(X,Y), not oneOf(Y), d_cover(Y).
```

(III) is translated according to step 7 (d):

```
not_precipitation_ok (X) :- precipitation(X, Y), not low(Y),
d_desert(X), d_prec_level(Y).
precipitation_ok (X) :- not not_precipitation_ok (X), precipita-
tion(X,Y), d_desert(X), d_prec_level(Y).
:- desert(X), not precipitation_ok(X).
```

(IV) is represented due to step 7 (d) as:

```
not_vegetation_ok (X) :- vegetation(X, Y), not scanty (Y),
d_desert(X),d_vegetation(Y).
vegetation_ok (X) :- not not_vegetation_ok (X), vegetation(X,Y),
d_desert(X),d_vegetation(Y).
```

and step 7 (d) for default subsumption:

```
:- desert(X), not vegetation_ok(X), not ab_3 (X).
ab_3(X) :-  desert(X), vegetation(X, Y), -scanty(Y).
```

(V) is encoded similar to the previous formulae by means of step 7 (d) :

```
not_populatedby_ok (X) :- populatedby(X, Y), not adopted_animal(Y),
d_desert(X),d_animal(Y).
populatedby_ok (X) :- not not_populatedby_ok (X), populatedby(X,Y),
d_desert(X),d_animal(Y).
```

and step 7 (d) for default subsumption:

```
:- desert(X), not populatedby_ok(X), not ab_5 (X).
ab_5(X) :-  desert(X), populatedby(X, Y), -adopted_animal(Y).
```

Smodels computes 4 answer sets, where each contains the following:

```
desert(sturt),area(sturt), ab_2(sturt), populatedby(sturt,camel),
scanty(few_cactus), vegetation(sturt,few_cactus), precipita-
tion(sturt,twenty_cm).
```

We can see that Sturt is an abnormal desert w.r.t. the covering (ab_2) since it is not covered by sand or gravel but only by stones. The condition of being covered by sand or by gravel is not fulfilled in the case of Sturt desert, nevertheless Sturt is considered being a desert, as intended. It would not be the case if the core property of having low precipitation is not satisfied. Then we would obtain no model which corresponds to the intuition that there is no desert without low precipitation.

7 Related Work

Related work can be organized into two groups. The first group contains approaches that explicitly distinguish between defeasible and nondefeasible parts of a concept's structure. In MultiNet [15], which is a knowledge representation paradigm along the line of Semantic Networks, three types of concept descriptions are distinguished: categorical, prototypical and situational. Categorical and prototypical knowledge corresponds to our core and typical characteristics, respectively.

The idea of partial nonmonotonicity for the Semantic Web is introduced in [4]. The system called DR-DEVICE is capable of reasoning about RDF metadata over multiple Web sources using defeasible logic rules. The implementation is declarative because it interprets the *not* operator using well-founded semantics. Compared to our one, this approach uses well-founded semantics instead of answer set semantics. In fact, the second is more robust and general in the sense that every well-founded model is an answer set but not conversely.

On the other hand many authors investigate the problem of integrating DLs with other defeasible logics. Baader in [2] considers the problem of integrating Reiter's default logic into a terminological representation system. In [24] a general framework for Preferential Default Description Logics (PDDL) is developed. Recently Heymans in [16] extended the description logic $\mathcal{SHOQ}(\mathbf{D})$ by a preference order on the axioms and thus effectively introduced nonmonotonicity into $\mathcal{SHOQ}(\mathbf{D})$.

As we are interested in adding defeasibility to Semantic Web languages directly by means of answer set semantics, only the work of Bertino [5] and colleagues developed a similar idea. However, instead of giving a non-standard interpretation to the properties of DAML+OIL, we divide knowledge into defeasible and not defeasible parts, thus avoiding all conclusions to be canceled.

A second group of related work forms proposals for translating DLs to various logic programming languages. In [14] Grosof investigates many possible interactions among DLs and Datalog logic programs (def-LPs). Similar approaches are presented in [10] and [18] where plain Datalog is combined with certain DL paradigms and hybrid languages are proposed. Finally, Rosati in [27] combines disjunctive Datalog with $\mathcal{ALC}$ based on a generalized answer set semantics.

On the other hand some authors discuss approaches combining DLs with a more expressive logic programming paradigm, namely Answer Set Programming. In [17] the DL $\mathcal{SHIF}$ is simulated by free disjunctive logic programs (DLP). Alsac and Baral in [1] show a translation, used as the foundation of presented framework, of DL $\mathcal{ALCQI}$ into declarative logic programming under answer set semantics. Swift [29] reduces inference in the description logic $\mathcal{ALCQI}$to query answering from answer sets of logic programs. In the paper [12] the integration of rules and ontologies in the Semantic Web in a form of combining logic programming under the answer set semantics with description logics is presented. The paper is focused on $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$.

Mainly, the intention of the papers mentioned above is either to combine the semantic and computational strengths of the two systems or to use powerful logic programming technology for inference in description logics, as noticed in [12]. The idea behind our work is similar to the former view, since the answer set solver is used as reasoner. On the other hand, we are close to the spirit of [2], [16] and [4], where conclusions are divided explicitly into defeasible and not defeasible ones.

8 Conclusions and Future Work

In the current paper we have presented a framework that permits to delimit the defeasible part of a concept here called *typical* from the nondefeasible one - *core*. It is a balanced way between purely monotonic approaches to representing ontologies in the Semantic Web on the one hand and purely nonmonotonic on the other. Both types of knowledge, defeasible and not defeasible, can be coherently represented by means of the developed framework.

We developed a system of metatags for annotating characteristics of concepts. The tags are defined in form of a metaontology. The tags `[core]` and `[typical]` assigned to each of a concept's characteristic provide the information whether the characteristic has a defeasible or nondefeasible nature.

A translation of annotated OWL axioms into a logic program under answer set semantics is given. ASP provides an intuitive semantics to the Semantic Web languages and extends them by nonmonotonicity. On the other hand a powerful logic programming technology is used for inference purposes. We use the answer set solver Smodels as reasoner for annotated ontologies, thus handling properly the distinction between monotonic and nonmonotonic reasoning.

The framework is grounded on the cognitive theory of a concept's dual structure. Adoption of the Dual Theory provides a cognitive foundation for concept specifications in ontologies. The concepts in ontologies may then be cognitively more adequate, which means that when formalized they still preserve the structure of their originals. It seems, that cognitive adequacy of concepts reduces the deformation of knowledge which often takes place in the process of knowledge modeling and formalization [31].

Moreover, the approach permits handling inconsistency in the knowledge base, which, as presented, should be treated differently in the context of defeasible or nondefeaisble knowledge. The inconsistency of the core, i.e. of nondefeasible knowledge is not allowed but when concerning only a defeasible part of a concept's specification, it may be reduced to abnormality. As far as inconsistency among default conclusions is concerned, it is handled in answer set semantics by means of multiple models, representing possible solutions. In case of incomplete information, we admit default conclusions as long as no defeating information arises.

In future work the presented annotations can be extended so that not only the distinction of defeasible and not defeasible knowledge is possible, but also levels of defeasibility may be introduced. This may be done by adopting one among many approaches, extending answer set programming by priorities. One of the major proposals is Logic Programming with Ordered Disjunction, proposed by Brewka in [9]. In this framework, priorities among literals may be expressed such that degrees of defeasibility among abnormalities may be formulated. Moreover, the translation should be extended so the OWL datatypes are handled.

Acknowledgements. We are indebted to Hesham Khalil, Sören Auer, Frank Loebe and Sebastian Dietzold for fruitful discussions and to the anonymous reviewers for feedback on earlier versions of this paper.

References

1. Alsac, G., Baral, Ch.: Reasoning in description logics using declarative logic programming. Abstract, ASU Technical Report (2001-2002)
2. Baader, F., Hollunder, B.: Embedding Defaults into Terminological Representation Systems. Automated Reasoning 14 (1995) 149–180
3. Baader F. et.al.: The Description Logic Handbook. Cambridge University Press, 2003
4. Bassiliades, N. et. al.: A Defeasible Logic Reasoner for the Semantic Web. RuleML (2004) 49-64
5. Bertino E. et. al.: Local Closed-World Assumptions for reasoning about Semantic Webdata. Proc. of AGP'03 APPIA-GULP-PRODE 2003
6. Brachman, R.J., Schmolze, J.G.: An overview of the KL-ONE knowledge representation system. Cognitive Science 9(2) (1985) 171-216
7. Brachman, R.J.: 'I Lied about the Trees' or, Defaults and Definifons in Knowledge Representation. AI Magazine 6(3) (1985) 80-93
8. Brewka G.: Nonmonotonic Reasoning: Logical Foundations of Commonsense Cambridge: Cambridge University Press, 1991
9. Brewka, G.: Logic Programming with Ordered Disjunction. Proc. of AAAI'02, Canada (2002) 100-105

10. Dionini, F.M. et. al.: AL-log: integrating datalog and description logics. Journal of Intelligent and Cooperative Information Systems 10 (1998) 227-252
11. Eiter, T. et.al.: The DLV system for knowledge representation and reasoning: Infsys Research Report, Austria (2002)
12. Eiter, T., et.al.: Combining Answer Set Programming with Description Logics for the Semantic Web. Proc. of KR'04, Canada (2004) 141-151
13. Gelfond M. and N. Leone: Logic Programming and Knowledge Representation - A-Prolog perspective , AI 138 (2002) 3-38
14. Grofof, B.N. et. al.: Description Logic Programs: Combining Logic Programs with Description Logic. Porc. of WWW'03 Hungary (2003) 48-57
15. Helbig, H., Gnörlich, C.: Multilayered Extended Semantic Networks as a Language for Meaning Representation in NLP Systems. In: Gelbukh, A. F. (eds.): Computational Linguistics and Intelligent Text Processing Proceedings of the Third International Conference, CICLING'02, Mexico (2002) 17-23
16. Heymans, S., Vermeir, D.: A Defeasible Ontology Language. Proc. of ODBASE'02, USA (2002) 1033-1046
17. Heymans, S., Vermir, D.: Integrating ontology languages and answer set programming, Proc. of DEXA'03, Czech Republic (2003) 584
18. Levy, A., Rousset, M.: CARIN: A representation language combining Horn rules and description logics. Proc. of ECAI'96, Hungary (1996) 323-327
19. Lifschitz, V.: Answer set programming and plan generation. AI 138 (2002) 39-54
20. Margolis E., Laurence S.: Concepts and Cognitive Science. In E. Margolis E., Laurence S. (eds.): Concepts: Core Readings. Cambridge, MA.: Bradford Books/MIT Press, 1999.
21. Margolis E., Laurence S.: Concepts. In Warfield T., Stich S. (eds): The Blackwell Guide to the Philosophy of Mind. Blackwell. 2003
22. Niemelä, I., Simons, P.: Smodels-an implementation of the stable model and well-founded semantics for normal logic programs. Proc. Of LPNMR'97, Germany (1997) 420-429
23. Osherson D.N., Smith E.E.: On the adequacy of prototype theory as a theory of concepts. Cognition 9(1) (1981) 35-58
24. Quantz, J.J., Ryan, M.: Preferential Default Description Logics. KIT-Report 110. Technische Universitat Berlin (1993)
25. Patel-Schneider, P.F. et. al.: Web ontology language (owl) abstract syntax and semantics. W3C Recommendation (2004)
26. Reiter, R.: On reasoning by default. Proc. of Theoretical Issues in Natural Language Processing USA (1978) 210 - 218
27. Rosati, R.: Towards expressive KR systems integrating datalog and description logics: Preliminary report. Proc. of DL'99, Sweden (1999) 160–164
28. Rosch, E., Mervis, C.: Family Resemblances: Studies in the Internal Structure of Categories. Cognitive Psychology 7 (1975) 573-603
29. Swift, T.: Deduction in ontologies via ASP. Proc. of LPNMR'04, USA (2004) 275-288
30. Wittgenstein, L.: Philosophical Investigations. Blackwell. Oxford. 1953
31. Zhang, J.: Representation of Health Concepts: Cognitive Perspective. Journal of Biomedical Informatics 35 (2002) 17-24
32. Zadeh, L.: Fuzzy Sets. Inform.control 8 (1965) 338-353