SuperDuper Schema: an OWL2+RIF DnS pattern

Aldo Gangemi
Semantic Technology Laboratory
CNR-ISTC Roma
Email: aldo.gangemi@cnr.it

I. Problem addressed

A. *Multiple descriptions*

Many problem-solving tasks require reasoning over alternative conceptualizations of a same situation or set of facts. Section I-B below includes typical cases when such reasoning is needed.

Descriptions and Situations (DnS)[1] is an expressive knowledge pattern usable as a representation framework for several complex patterns. The basic intuition goes to the ability to distinguish two levels of description:

- a "ground" description, using whatever vocabulary or axioms, which constitutes the representation of a **situation**. The situation section of DnS is basically a container for vocabulary and axioms that help refactoring a FOL n-ary (typically ≥ 2) relation into a class with *n* binary relations, i.e. the so-called *n-ary relation (logical) pattern*
- a "duper" description, using the DnS vocabulary or some specialization of it, which provides an alternative schema to describe a class of situations (that already have a ground description, or that can get one emerging out of existing data). The description part of DnS talks about (reified) relations, concepts, and the relations between them and between them and ground entities.

B. *Sample topics and sentences*

Topics and example sentences from the DeepKR list:

1) Abstraction and high level descriptions: e.g. "Enzymes proofread DNA during its replication and repair damage in existing DNA"
2) Explanation: "The logistics of carrying out metabolism set limits on cell size."
3) Hypotheticals, particularly as parts of explanations: e.g. "The lysosome provides a space where the cell can digest macromolecules safely, without the general destruction that would occur if hydrolytic enzymes roamed at large."
4) Representation of context: e.g. "In adults, the illness is rarely serious, but in infants it can be fatal". "In the elderly, pneumonia is a common cause of confusion."

Similarly, but from DnS story since 2003:

1) Interpretation of situations (e.g. in archaeology, literature, psychology)
2) Diagnostic interpretation of cases (diagnosis of diseases and syndromes against signs and findings)
3) Normative interpretation of cases (case abstraction, application of norms and meta-norms in Law)
4) Plan models to be satisfied on scheduling or execution
5) Plan models or schedules to be built based on available resources
6) Control checking (matching actual vs. expected facts, classifying unwanted or unexpected facts)

II. Solutions

A. *Talking about situations*

There is a long story of dealing with relations that range over more than two arguments, or have polymorphic arities. (Generic) use cases come from AI (e.g. situation calculus, reified solutions to otherwise HOL sentences, Minsky's frames, description logics, DLR, etc.), from philosophy (e.g. Davidson's events and the debate on the nature of events vs. propositions, four-dimensionalism in formal ontology), from linguistics (e.g. FrameNet frames), from conceptual modeling (e.g. association classes, Guizzardi's trope-based four-dimensionalism), etc.

B. *The `Situation` pattern*

In DnS, the `Situation` pattern[2] contributes a basic OWL vocabulary:
OWL Classes: *Situation*
OWL ObjectProperties: *isSettingFor* ($\subseteq$ Situation$\times$ owl:Thing)

Examples:

```
           DrivingCC055LEIn26July2011
                    rdf:type Situation  (1)
     {Driver_123,CC055LE,26July2010}
                    rdf:type owl:Thing  (2)
              DrivingMyCarIn26July2011
isSettingFor {Driver_123,CC055LE,26July2010}  (3)
```

When using `Situation` we need to specialize the *Situation* class and the *isSettingFor* property with domain-oriented vocabulary: for time, space, agents, etc. In OWL2, `Situation` is improved with identification constraints through the *owl:hasKey* metaproperty, useful when there are

[1]http://www.ontologydesignpatterns.org/descriptionandsituation.owl

[2]http://www.ontologydesignpatterns.org/cp/owl/situation.owl

multiple binary branches of an n-ary relation, so that each tuple of that relation will remain unique even after reification.

C. Three approaches to relation or situation indexing

Pat Hayes recently stimulated the discussion with respect to how to attach temporal indexing to sentences of the form `R(a,b)`, and suggested three possible logical patterns, summarized as such (I quote him literally here):

- *Attach it to the sentence*, meaning that the sentence R(a,b) is true at the time t. This gives you a hybrid or context logic where the times are possible temporal worlds/indices or contexts, to which truth is relativized. But the sentences being so relativized do not themselves make any reference to time. Call this 3D.
- *Attach it to the relation as an extra argument*, and call the relation a 'fluent': R(a, b, t) This gives you the classical AI/KR approach which used to be called the situation calculus, where one quantifies over times in the KR language itself, but the object terms are still thought of as denoting 3D rather than 4D entities. Call this 3D+1.
- *Attach it to the object terms*, (using a suitable function, written here as an infix @): R(a@t, b@t) This requires us to make sense of this @ operation, and it seems natural to say that it means the t-slice of the thing named, which now has to be re-thought as a 4D entity. So the a, b things have morphed form being 3D (but lasting through time) to being genuinely 4D, and having temporal slices or parts. Call this 4D.

Hayes suggests that the different patterns are just syntactic variants, which (in principle) could be unified by some algorithm. Instead of attempting that algorithm, more modestly we have tried to represent the three approaches in OWL2, and have counted the resulting axioms for each of them, just to make it a concrete problem. Consider that the proposal of Hayes' came in the middle of a discussion about 3D vs. 4D ontologies,[3], and the approaches were tied to time indexing, but the problem is more general, and touches any attempt to include a new dimension in order to perspectivize facts or entities.

In order to exemplify the approaches in a computable language, the following solutions are reported by means of a leading example: the fact that Garibaldi landed in Sicily in 1861:

Solution 1: Sentence indexing (à la context/modal logic, 3D). As a sentence in FOL plus meta-level sugar an example is:

$$holdsAt((landsIn(Garibaldi, Sicily)), 1860) \quad (4)$$

while in OWL2 (*singleton property logical pattern with punning*) a possible refactoring is:

```
landsIn rdf:type owl:ObjectProperty        (5)
      GaribaldiLandsInSicilyIn1860
           rdfs:subPropertyOf landsIn      (6)
      GaribaldiLandsInSicilyIn1860
            rdfs:domain {Garibaldi}        (7)
      GaribaldiLandsInSicilyIn1860
                rdfs:range {Sicily}        (8)
      GaribaldiLandsInSicilyIn1860
                       holdsAt 1860        (9)
```

This solution requires a *singleton-universe* object property, which has valued domain and range, and is punned as an individual in order to assert the time indexing.

The axiom count for 100 FOL sentences with holdsAt after refactoring to OWL2 is as follows:

- 4*100 property declaration, subPropertyOf, domain, and range axioms
- 100 (time) data assertions
- In total, 500 axioms: 400 typically TBox axioms, 100 ABox axioms

This solution has the advantage of being able to talk about assertions, but the disadvantage of needing a lot of properties with singleton universes.

Solution 2: Relation indexing (à la frame logic, 3D+1). As a sentence in FOL an example is:

$$landsIn(Garibaldi, Sicily, 1860) \quad (10)$$

while in OWL2 (*situation knowledge pattern* with identification constraints when needed, no punning, ordering to be put on property names when needed), a possible refactoring is:

```
     Landing rdfs:subClassOf Situation (11)
 GaribaldiLandsInSicilyIn1860 rdf:type
                               Landing (12)
                     Landing owl:hasKey
    [hasEntity , hasPlace , hasTime]   (13)
GaribaldiLandsInSicilyIn1860 hasEntity
                             Garibaldi (14)
 GaribaldiLandsInSicilyIn1860 hasPlace
                                Sicily (15)
  GaribaldiLandsInSicilyIn1860 hasTime
                                  1860 (16)
```

This solution requires to specialize the *situation* knowledge pattern, and to possibly put hasKey axioms for identification constraints. Ordering can be put minimally in the name of properties when ambiguity arises. Punning is not needed. Solution (2) is a specialization of the `Situation` pattern.

The axiom count for 100 FOL sentences for landsIn, after refactoring to OWL2, is as follows:

- 5 TBox axioms: 3 property declarations, 1 rdfs:subClassOf axiom, 1 hasKey axiom
- 100 type assertions
- 300 object/data assertions

[3]Actually on a a thread of the Ontolog mailing list on 9 Feb 2011

- in total, 405 axioms: 5 TBox axioms, 400 ABox axioms

This solution has the advantage of being able to talk about assertions as (reifying) individuals, but the disadvantage of not being able to use them as properties as well (this could be accommodated by punning the individuals as object or data properties.

Solution 3: Individual indexing (à la 4D ontology). As a sentence in FOL an example is:

$$landsIn(Garibaldi@1860, Sicily@1860) \quad (17)$$

while in OWL2 (*phased individual logical pattern*), a possible refactoring is:

```
Garibaldi hasTemporalPart
                 Garibaldi@1860   (18)
Sicily hasTemporalPart Sicily@1860   (19)
Garibaldi@1860 landsIn Sicily@1860   (20)
Garibaldi@1860 occursAt 1860   (21)
Sicily@1860 occursAt 1860   (22)
```

This solution requires to introduce new entities: the phases of individuals referred to in the sentence.

The axiom count for 100 FOL facts for landsIn, after refactoring to OWL2, is as follows:

- 2*100 type assertions
- 5*100 object/data assertions
- in total, 700 ABox axioms

An advantage of this solution is that one can compactly (no additional axioms required) represent temporally heterogeneous relations, e.g. *John@2011 likes NewYork@1899*. However, the time of the sentence as different from the times of the elements cannot be represented.

A general objection to 4D approaches is that we could have *nD* entities, e.g. for place, orientation, state of matter, numerosity, etc.. In other words, we can perspectivize things within any dimension, not just time (cf. **??**).

To conclude the first part on approaches to relation indexing, it seems that solutions (1) and (3) suffer of limitations due either to the lack of meta-level axioms (solution (1): asserting about other axioms), or to the need to introduce new entities for phased individuals. Anyway, the main issue is maybe the difficulty to use (1) or (3) to refactor relations of arity ≥ 3, or polymorphic. Larger arities would require in (1) more punned singleton properties, e.g.:

```
GaribaldiLandsInSicilyWith1000Soldiers
                rdfs:subPropertyOf landsIn (23)
GaribaldiLandsInSicilyWith1000Soldiers
                rdfs:domain {Garibaldi} (24)
GaribaldiLandsInSicilyWith1000Soldiers
                rdfs:range {Sicily} (25)
GaribaldiLandsInSicilyWith1000Soldiers
                with 1000Soldiers (26)
```

in (3) more phased individuals, e.g.:

```
GaribaldiWith1000Soldiers
            with 1000Soldiers   (27)
Garibaldi hasCollectivePerspective
            GaribaldiWith1000Soldiers   (28)
```

while in (2) we should just add one global new property, e.g.:

```
GaribaldiLandsInSicilyIn1860
            with 1000Soldiers   (29)
```

In practice, the supposed only syntactic flavor of the different solutions dissolves once we consider a really multidimensional ontology. Why should we talk of 3D, 3D+1, or 4D, when the dimensional complexity of a modelling solution is decided by the requirements of the ontology, not by a philosophical choice? Solution (2) appears flexible enough to work with nD ontologies.

D. Some approaches to talk about descriptions

Descriptions are typically made in a logical language. For example, all solutions exemplified in the previous sections are descriptions of some facts or situations. Following logical good practices for preserving the integrity and complexity results about first-order languages, the vocabulary used to design those descriptions is disjoint from the vocabulary of the individuals described. Many workarounds have been proposed for mixing different logical layers without needing actual Higher-Order Logic (HOL), e.g. OWL2 punning [10], Hi-DL-Lite [2], Hayes' *pollarding* in Common Logic, and Kifer's F-Logic [7].

However, logical approaches do not provide any good practices to use such expressive capabilities, i.e. what such additional expressivity should be used for. Some use cases and solutions have been provided e.g. by the W3C task force on ontology patterns, limited to the capability of talking about concepts, for example when representing subjects of books. DnS [5] is the first proposal of a formal ontology for talking about concepts and (reified) relations. A detailed axiomatization, based on DOLCE, of a part of DnS is [8]. A lightweight example of an ontology for talking about concepts and the relations between them is SKOS [9]. LMM [4] is a semiotic ontology to talk about the meaning and reference of linguistic expressions, and makes wide usage of punning axioms.

Departing from the DOLCE-oriented version of DnS, cDnS [3] is an attempt to cover most of the entities that lie in the grey area between typical, publicly recognized individuals (e.g. physical objects), and typical, publicly recognized predicates (e.g. classes, relations): concepts, collections, frames, communities, terms, meanings, interpretations (plans, norms, diagnoses, hypotheses, etc.).

Here we drastically simplify the problems in the literature, and present some simple patterns to:

- create a vocabulary for the intension of concepts and (reified) relations
- talk about descriptions, i.e. to *redescribe*, or *dupe* (in the sense of duplicating) a logical description (e.g. an assertional axiom), hence the name "duper"
- represent such vocabulary by exploiting OWL2 punning, i.e. *super-dupering* logical descriptions
- adding RIF rules in order to generalize the pattern

E. The `Description` pattern

The `Description` pattern[4] is very simple, only providing the means to associate descriptions (reifications of the intension of n-ary relations) with the concepts they either define or use, and a specialization property between either concepts or descriptions:
OWL Classes: *Description* (D), *Concept* (C)
OWL ObjectProperties: *defines* ($\subseteq D \times C$), *usesConcept* ($\subseteq D \times C$), *specializes* ($\subseteq (D \times D \cup C \times C)$)

Examples:

$$\texttt{ItalianConstitutionalLaw rdf:type D} \quad (30)$$

$$\texttt{Citizen rdf:type C} \quad (31)$$

$$\texttt{ItalianConstitutionalLaw defines Citizen} \quad (32)$$

$$\texttt{ItalianConstitutionalLaw usesConcept Republic} \quad (33)$$

$$\texttt{ParliamentaryRepublic specializesRepublic} \quad (34)$$

The advantages of the `Description` pattern are in the possibility of (lightly) talking about complex entities such as plans, norms, diagnoses, hypotheses, desires, etc., which are relational in nature, without resorting to modal logics. When a deep reasoning on these entities is required, we can wrap some specialized reasoner, which will benefit from the explicit typing of the entities to be treated in its logical language (e.g. a deontic logic or a constraint satisfaction language).

`Description` also nicely complements SKOS, since *Concept* can be aligned to *skos:Concept*, and *specializes* to *skos:broader*, while SKOS lacks notions like *Description* and *defines*.

Consider that `Description` alone is not addressing any of the Deep KR tasks mentioned in the Problem section. Next section is about full DnS, which addresses those tasks.

F. The DnS basic pattern

Traditionally, the universe of facts is not the same as the universe of concepts. With multiple descriptions, clarifying the disjoint universes is specially important in a reified world. In the past, KR deep reasoning tasks were dealt with specialized languages, but in an integration perspective we need clear universes of discourse. Some of those tasks address:

- legal knowledge (facts vs. cases vs. norms vs. meta-norms)
- services, planning and control (actual vs. expected facts)
- diagnoses and situation awareness (ground vs. (un)wanted facts)
- hypothesis testing and CBR (building/testing interpretations from facts)

The `Descriptions and Situations` pattern[5] (DnS) is a pattern to represent the entities involved in a common abstract task: *Classifying a situation*, according to possibly incomplete, alternative or loose constraints and concepts. When classifying a situation, constraints must be explicit and explicitly linked to the representation of a situation. DnS enables the representation of situations and descriptions, and their respective links: entities of a situation that play roles from a description, values of a situation that fit parameters from a description, events of a situation that accomplish a task defined by a description, etc. Given its general intuition, representing and reasoning with DnS gets to grips with its possible implementations, which are summarized in the following sections.

G. The segregated DnS "duper" pattern

The first implemented versions of DnS were written in DAML+OIL and OWL1, where no punning is possible. Duping the concepts in the signature of the ontology was therefore needed. The additional axioms needed to abridge the `Situation` (S) and *Description* (D) axioms are the following:
OWL ObjectProperties: *classifies* ($\subseteq$ $(\mathit{Concept} \times \mathit{owl}{:}\mathit{Thing} \cup \mathit{Description} \times \mathit{Situation})$)
An **example** ontology for this pattern is the following:
[Generic TBox addons]

$$\texttt{CoreConcept rdfs:subClassOf C} \quad (35)$$

[Domain TBox]

$$\texttt{SalesModel rdf:type D} \quad (36)$$

$$\texttt{RegularSale rdfs:subClassOf S} \quad (37)$$

$$\texttt{RegularSale rdfs:subClassOf (classifies}^{-}\texttt{ value SalesModel)} \quad (38)$$

$$\texttt{RegularSale owl:equivalentClass (isSettingFor some (classifies}^{-}\texttt{ (CoreConcept and some (defines}^{-}\texttt{ SalesModel)))} \quad (39)$$

[ABox]

$$\texttt{SalesModel defines } \{\texttt{Seller, Buyer, Product, SaleTime}\} \quad (40)$$

$$\texttt{Seller rdf:type CoreConcept} \quad (41)$$

$$\texttt{Buyer rdf:type CoreConcept} \quad (42)$$

$$\texttt{Product rdf:type CoreConcept} \quad (43)$$

[4]http://http://www.ontologydesignpatterns.org/cp/owl/description.owl

[5]http://www.ontologydesignpatterns.org/cp/owl/descriptionandsituation.owl

```
                         SaleTime rdf:type C     (44)
                      Sale_c4598 isSettingFor
                      (Apple,Mustafa,iPad)       (45)
                    Seller classifies Apple      (46)
                 Buyer classifies Mustafa        (47)
                   Product classifies iPad       (48)
```

[Inferred axioms]

```
      ⊨ Sale_c4598 rdf:type RegularSale          (49)
   ⊨ Sale_c4598 classifies⁻ SalesModel           (50)
```

Duper DnS is good at representing the links between entities from a situation and concepts from a description, but as far as reasoning is concerned, the main tasks that can be carried out are limited to the knowledge bases where both situations and descriptions are sufficiently described and linked: representations of contexts (such as in the DeepKR scenario of age-oriented contextualization of diseases), executed plans with reference to plan models, applied (e.g. verified) interpretations and hypotheses, etc. Even in these cases, only some simple tasks can be performed in OWL(1).

In the example above, we are able to infer that a certain sale is an instance of *RegularSale*, and even that that sale is classified by the description *SalesModel*, but this inference power is quite weak, because we cannot state that *all and only* the things in the setting that are classified by a core concept that is defined by *SalesModel* can make the situation classified under *SalesModel*: we can say "some" of them; in practice, a situation with just one seller is enough to classify it as *RegularSale*!

In order to circumvent this problem, we might include explicitly all classification restrictions for core concepts in the equivalence axiom (51).

```
      RegularSale owl:equivalentClass
                  (isSettingFor some
       (classifies⁻ some (Seller and
       (defines⁻ value SalesModel))))
                                 etc.     (51)
```

This solution is really ad hoc, and definitely ugly. It'd be tempting here to use directly the Situation pattern alone, so as to create a catch-it-all *owl:equivalentClass* axiom (52).

```
      RegularSale owl:equivalentClass
          (isSettingFor some Seller))
           (isSettingFor some Buyer))
        (isSettingFor some Product))      (52)
```

However, the issue here is being able to *take into account also axioms made at the description layer*, such as the fact that some concepts are core, and others not, or that Seller is dual to Buyer (i.e. they are mutually dependent for a regular sale transaction). Moreover, how to state e.g. that *Seller* is a *CoreConcept* and a dual to *Buyer* only *in the context of a RegularSale*? In practice, we need equivalentClass axioms that are generalized to one holding for all classes of situations. E.g. something like the FOL axiom 53:

$$\begin{aligned} &\forall(s,e,c,d)((\\ &isSettingFor(s,e) \land CoreConcept(c) \land\\ &classifies(c,e) \land defines(d,c) \land\\ &\neg\exists(e_1,c_1)(\\ &isSettingFor(s,e_1) \land \neg CoreConcept(c_1) \land\\ &\neg classifies(c_1,e_1) \land \neg defines(d,c_1)))\\ &\rightarrow (classifies(d,s) \land S(s))) \end{aligned} \tag{53}$$

Additionally, with Duper DnS we are obliged to (manually or programmatically) create two distinct constants for the description (*SalesModel*), and the situation class (*RegularSale*), which are the intensional respectively extensional aspects of the same (reified) relation.

Indeed, with Duper DnS it's very hard to figure out not only the following 2, 3, and 4 tasks, which feature incomplete information, so that soft, fuzzy or probabilistic rules would be possibly needed, but even 1, which is a deterministic task:

1) to decide if a situation can be (partly or fully) classified under a description by classifying relevant things in relevant concepts, in presence of description- or concept-level relations (diagnoses, control checking)
2) to evaluate what situation fits best a description (discovery, knowledge mining)
3) to make a situation emerge out of scattered facts (case-based reasoning, heuristic classification)
4) to evaluate what description fits best in order to unify a certain set of facts (norm application, hypothesis testing, explanation)

In the next section, we show novel patterns that employ OWL2 and RIF in order to solve tasks of type (1). Tasks of the other types would require soft rules (4), or hybridizing deterministic and probabilistic reasoners (2,3), whose patterns are not covered here.

H. Super-duper DnS

In OWL2, the *equivalentClass* axiom 39 could be more elegantly implemented with a property chain (axiom 54), while the duplicated vocabulary issue can be solved via punning, so that the overall model becomes as follows:
[DnS TBox add-ons]

```
(isSettingFor ∘ classifies⁻ ∘ defines⁻)
          owl:subPropertyOf classifies⁻ (54)
          CoreConcept rdfs:subClassOf C (55)
```

[Domain TBox]

```
                RegularSale rdf:type D      (56)
      RegularSale rdfs:subClassOf S         (57)
          RegularSale rdfs:subClassOf
   (classifies⁻ value RegularSale)          (58)
```

[ABox]

$$\texttt{Seller rdf:type CoreConcept} \tag{59}$$
$$\texttt{Buyer rdf:type CoreConcept} \tag{60}$$
$$\texttt{Product rdf:type CoreConcept} \tag{61}$$
$$\texttt{SaleTime rdf:type C} \tag{62}$$
$$\texttt{(Seller dualTo Buyer))} \tag{63}$$
$$\texttt{RegularSale defines}$$
$$\{\texttt{Seller, Buyer, Product}\} \tag{64}$$
$$\texttt{Sale}_{c4598}\ \texttt{isSettingFor}$$
$$\texttt{(Apple,Mustafa,iPad)} \tag{65}$$
$$\texttt{Seller classifies Apple} \tag{66}$$
$$\texttt{Buyer classifies Mustafa} \tag{67}$$
$$\texttt{Product classifies iPad} \tag{68}$$

[Inferred axioms]

$$\models \texttt{Sale}_{c4598}\ \texttt{rdf:type RegularSale} \tag{69}$$
$$\models \texttt{Sale}_{c4598}\ \texttt{classifies}^{-}\ \texttt{RegularSale} \tag{70}$$

Notice the “mapping” axiom (58) of *RegularSale* on being classified by itself. A variant was needed in OWL1 Duper DnS (axiom 39) in order to connect *RegularSale* to *SalesModel*, and making the classification be inherited to instances of *RegularSale*. However, even in OWL2 SuperDuper DnS it is needed, because punning does not substantially change the semantics of the OWL1 ontology.

For completeness, we might mention that a purely “situational” solution can be provided in OWL2, by adding punning axioms such as 71, 72, and 73 directly to axiom 52:

$$\texttt{(Seller rdf:type CoreConcept))} \tag{71}$$
$$\texttt{(Seller dualTo Buyer))} \tag{72}$$
$$\texttt{(Seller defines}^{-}\ \texttt{RegularSale))} \tag{73}$$

In practice, we would reconstruct the Description-level axioms as in the SuperDuper sample ABox above (axioms 59,63,64, but they would be semantically disjoint from the Situation-level axioms.

Therefore, besides being cleaner, the SuperDuper OWL2 implementation of the pattern does not provide a generalized solution as required by task (1). Nonetheless, punning allows us to easily hybridize OWL with RIF, as shown in the next section.

I. Super-duper DnS + RIF

The last solution proposed here hybridizes the OWL2 version of the DnS pattern with a RIF rule. Rule 74 makes, on a global base, a rule reasoner infer the classification axiom between a situation and a description whereas all core concepts defined by the description are classified by entities in the setting of a situation. The consequent part of the rule also makes us infer that that description is subclass of the class Situation. The double interpretation of the description can be made thanks to punning.

```
                forall ?s ?e ?c ?d
                 ?s [rdf:type ?d] |
   ?d [rdfs:subClassOf Situation]
                                ←
                           and (
              ?s [isSettingFor ?e]
                  ?e [rdf:type ?c]
                 ?c [defines⁻ ?d]
        ?c [rdf:type CoreConcept]
                (not (exists ?e1
                           and (
              ?s [isSettingFor ?e]
          (not (?e [rdf:type ?c]))
    (not (?c [defines⁻ ?d]))))))
```
(74)

The RIF rule presented here lets us accomplish some tasks of type (1), since it complements the OWL2 pattern to reach the expressivity of the axiom 53. The most important novelty with respect to previous DnS implementations is that the SuperDuper schema puns the names of descriptions and situation types (the intensional vs. extensional parts of a reified n-ary relation), and cleanly links to rules 74 in order to making *rdf:type* and *classifies*$^{-}$ isomorphic in the scope of the DnS pattern.

III. Conclusion

In this review of design patterns for representing situations, and the interplay between descriptions and situations, we have shown that a classic reified approach to represent situations, together with description-level axioms, can be used as a design pattern to approach several deep KR problems. Such a pattern requires a rich expressivity in order to accomplish even basic useful reasoning tasks. The solution proposed includes some novel constructs introduced by OWL2 (punning, keys, property chains), as well as a rule language like RIF Core, which extends SWRL towards FOL with fully quantified rules.[6] Other rule languages could be used for this task, so that the solution introduced can be also used as a bridge between DL-based ontologies, and AI languages such as KM, OCML, PowerLoom, Alchemy, etc. In particular, we have tested OCML, and it allows a clean implementation of the RIF rule.

IV. The ODP portal

In the presentation, we will exemplify the support provided by the ODP wiki[7] for collecting, reviewing, annotating and

[6]Such an extension of SWRL was proposed already in 2005 within W3C, cf. http://www.w3.org/Submission/SWRL-FOL.

[7]http://www.ontologydesignpatterns.org

publishing knowledge patterns like the ones introduced here [6], [11], [1].

References

[1] E. Blomqvist, V. Presutti, E. Daga, and A. Gangemi. Experimenting with extreme design. In *EKAW*, pages 120–134, 2010.

[2] G. De Giacomo, M. Lenzerini, and R. Rosati. Towards Higher-Order DL-Lite. In *Proc. of DL2008*, Dresden, Germany, 2008.

[3] A. Gangemi. Norms and Plans as Unification Criteria for Social Collectives. *Journal of Autonomous Agents and Multi-Agent Systems*, 16(3), 2008.

[4] A. Gangemi. *What's in a schema? A formal metamodel for ECG and FrameNet*. Studies in Natural Language Processing. Cambridge University Press, 2010.

[5] A. Gangemi and P. Mika. Understanding the Semantic Web through Descriptions and Situations. In *CoopIS/DOA/ODBASE*, pages 689–706, 2003.

[6] A. Gangemi and V. Presutti. Ontology Design Patterns. In S. Staab and R. Studer, editors, *Handbook on Ontologies, 2nd Edition*. Springer Verlag, 2008.

[7] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of Association for Computing Machinery*, 42:4:741–843, 1995.

[8] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social roles and their descriptions. In C. Welty and D. Dubois, editors, *Proc. of Knowledge Representation and Reasoning (KR)*, pages 267–277, 2004.

[9] A. Miles and D. Brickley. SKOS Core Vocabulary Specification. Technical report, World Wide Web Consortium (W3C), November 2005. http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/.

[10] B. Motik. On the properties of metamodeling in owl. *J. Log. Comput.*, 17(4):617–637, 2007.

[11] V. Presutti, E. Daga, A. Gangemi, and A. Salvati. http: //ontologydesign-patterns.org [odp]. In *International Semantic Web Conference (Posters & Demos)*, 2008.