Ontology Patterns with DOWL: The Case of Blending[⋆]

Oliver Kutz[1], Fabian Neuhaus[2], Maria M. Hedblom[1,2], Till Mossakowski[2], and Mihai Codescu[1]

[1] KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy
[2] Institute for Intelligent Cooperative Systems
Otto-von-Guericke University of Magdeburg, Germany

Abstract. The Distributed Ontology, Model, and Specification Language DOL provides logic-independent structuring, linking, and modularity constructs. Its homogeneous OWL fragment, DOWL, we argue, can be seen as an ideal language for formalising ontology patterns in description logics. It naturally consumes earlier formalisms such as C-OWL or DDL, and extends these with various expressive means useful for the modelling of patterns. To substantiate this, we illustrate DOWL's expressive power with a number of examples, including ontology design patterns, networks of ontologies, and ontology combinations. The latter are used to formalise conceptual blending, based on DOWL features such as renaming, filtering, forgetting, interpretation, and colimit computation.

Keywords: OWL; DOL; DOWL; conceptual blending; ontology engineering; ontology design patterns;

1 Introduction

While the use of ontologies varies considerably, there are two recurring challenges: *reusability* and *interoperability.*

Reusability is an issue because the development of ontologies is typically done manually by experts and, thus, an expensive process. Hence, it is desirable to be able to reuse existing ontologies during the development of new ontologies. This presupposes a framework that allows to build *structured ontologies* by identifying modules and their relationships to each other. For example, it requires the ability to combine two existing ontologies in a way that handles the namespaces of the ontologies in an appropriate way. Further, the reuse of an existing ontology often requires that the ontology is *adapted* for its new purpose. For example, the adaption may require the extension of the ontology by new axioms, or the extraction of a subset of the ontology, or the change of its semantics from open world to closed world.

[⋆] This paper draws heavily on material from [21] (for the outline of DOL) and [13] (for the basics of conceptual blending).

The interoperability challenge is closely related to the reusability challenge. Since the development of ontologies is not an exact science and is usually driven by project specific requirements, two ontologies that have been developed independently will represent the same domain in different and, often, conflicting ways. Thus, in a situation where two independently developed ontologies are supposed to be reused as modules of a larger ontology, the differences between these ontologies will typically prevent them from working together properly. Overcoming this lack of interoperability may require an alignment or even an integration of these ontologies. This typically involves the identification of synonyms, homonyms, and the development of bridge axioms, which connect the two ontologies appropriately.

Addressing these two challenges, there is a diversity of notions providing design patterns for and interrelations among ontologies. The Distributed Ontology, Model and Specification Language (DOL) aims at providing a unified metalanguage for handling this diversity. In particular, DOL enjoys the following distinctive features:

– structuring constructs for building ontologies from existing ontologies, like imports, union, forgetting, interpolation, filtering, and open-world versus closed-world semantics;
– module extraction;
– mappings between ontologies, like interpretation of theories, conservative extensions etc.;
– alignments, interpretations, and networks of ontologies;
– combination of networks.

DOL has been partially approved as a standard of the Object Management Group (OMG), and its finalisation is planned for late 2016 [23].

DOL and its structuring language are designed as a multi-logic meta-language, already supporting all of the mainstream ontology languages in use today. In this paper, we outline the purely homogeneous DL-based OWL fragment of DOL, called DOWL. We illustrate that it provides substantial modelling support for the OWL user, and, moreover, encompasses and extends several well-known modelling approaches, namely in particular C-OWL and DDL, standard alignment techniques, as well as module extraction.

We illustrate some of these features here with two main use-cases that go beyond standard description logic or OWL modelling, namely (1) instantiable schematic ontology patterns, and (2) networks of ontologies and their combination, here applied to the computation of conceptual blends. We close with a discussion of reasoning and tool support, and an outline of future work.

2 DOWL in a nutshell

2.1 Structured DOWL ontologies

Structured DOWL ontologies are generated by the following grammar, where O is a basic OWL ontology, Σ is a signature (i.e. a set of entities: concepts, roles and individuals) and σ a signature morphism (i.e. a map between the entities of two ontologies):

```
Onto ::= O
       | IRI
       | Onto and Onto | Onto then [Anno] Onto
       | Onto with σ
       | Onto reveal Σ | Onto hide Σ
       | Onto keep Σ | Onto forget Σ
       | Onto extract Σ | Onto remove Σ
       | Onto select O | Onto reject O
       | minimize Onto | maximize Onto
       | combine Network | { Onto }
Anno ::= %def | %cons | %implied
```

A *basic ontology* O is written in some OWL serialisation, e.g. OWL Manchester syntax:

```
Class: Woman EquivalentTo: Person and Female
ObjectProperty: hasParent
```

As shown in this example, O can be an ontology fragment, which means that some of its entities are declared outside of O (e.g. in an imported ontology).

An *IRI reference* refers to an ontology existing on the Web, possibly abbreviated using prefixes, e.g.:

```
  <http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/pizza.owl>
```

or using prefixes:

```
%prefix(
  co-ode: <http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/> )%
co-ode:pizza.owl
```

An *extension* of an ontology by new entities and axioms is written O_1 **then** O_2, where O_2 is an ontology (fragment). An extension can optionally be marked as conservative (%cons after the "**then**"), stating that O_2 does not introduce any new constraints in terms of the language of O_1. In case that O_2 does not introduce any new entities, the keyword %implied can be used instead of %cons; the extension then merely states intended logical consequences. The keyword %def stands for definitional extensions, expressing that the interpretation of the new entities in O_2 is uniquely determined by the axioms for a given interpretation of O_1. The following OWL ontology is an example for the latter:[1]

```
  Class Person
  Class Female
then %def
  Class: Woman EquivalentTo: Person and Female
```

Similar to extension is the *union* of two self-contained ontologies, written O_1 **and** O_2. Compared to extensions, O_2 is restricted here, because it cannot be a fragment. On the other hand, O_2 can be an arbitrary structured ontology, and not just a basic one, as for extensions.

A *translation* of an ontology to a different signature is written O **with** σ, where σ is a signature morphism. This is particularly useful when disambiguating homonyms that may accidentially get identified when uniting ontologies:

```
FinancialOnto with Bank |-> FinancialBank and GeoOnto with Bank |-> RiverBank
```

[1] Annotations such as %cons, %then and %def, introduce so called 'proof obligations' on the meta-level. That is, what they claim to be the case may be true or false and therefore requires verification by proof (or sometimes sufficient syntactic criteria).

DOL features four different forms of reduction of a large ontology to a smaller signature. Assume that in some large medical ontology like SNOMED CT, we are interested only in facts about hearts and heart attacks. Then we can write one of:

```
SNOMED extract Heart, HeartAttack
SNOMED keep Heart, HeartAttack
SNOMED reveal Heart, HeartAttack
SNOMED select Heart, HeartAttack
```

With **extract**, we extract a SNOMED CT module[2], which is a sub-ontology of SNOMED CT capturing the same facts about hearts and heart as SNOMED CT itself. The signature of the extracted module may be larger than just the two specified entities (heart and heart attack). In extreme cases, we might get the whole original ontology (which is of course not desirable, because then no reduction has taken place). Using **keep**, we get a uniform interpolant, which is not necessarily a sub-ontology, but rather an ontology that may involve new axioms in order to capture the SNOMED CT facts about hearts and heart attacks in an ontology featuring exactly the two specified entities, heart and heart attack. However, such an ontology may be hard to compute, if it exists at all. Then, we also can use **reveal**, which essentially keeps the whole of SNOMED CT and provides some export interface consisting of heart and heart attack only. This can be useful when interfacing SNOMED CT with other ontologies, e.g. in an interpretation. Finally, the use of **select** simply removes all SNOMED CT axioms that involve other symbols than heart and heart attack. While this can be computed easily, it might leave the user with a poor ontology capturing only a small fraction and only the basic facts of SNOMED CT's knowledge about hearts and heart attacks. DOWL also adds language constructs to OWL to express (non-monotonic) minimisation (resp. maximisation) of concepts, borrowing from circumscription [19, 3]. A *minimisation* of an ontology, written **minimize { O }**, imposes a closed-world assumption on part of the ontology. It forces the entities declared in O to be interpreted in a minimal way. Entities declared before the minimised part are considered to be fixed for the minimisation. Symbols declared after the minimisation can be varied. For example, in the following OWL theory, B2 is a block that is not abnormal, because it is not specified to be abnormal, and hence it is also on the table.

```
  Class: Block
  Individual: B1 Types: Block
  Individual: B2 Types: Block DifferentFrom: B1
then minimize {
        Class: Abnormal
        Individual: B1 Types: Abnormal }
then
  Class: OnTable
  Class: BlockNotAbnormal EquivalentTo:
    Block and not Abnormal SubClassOf: OnTable
then %implied
  Individual: B2 Types: OnTable
```

[2] DOL uses smallest depleting Σ-modules in the sense of [15] for the semantics of extractions.

Alternatively, we can maximise some entities. Using this, the example can be formulated in a more natural way, because now the concept of normal blocks is maximised:

```
ontology Blocks_Alternative2 =
  Class: Block
  Class: Normal
  Individual: B1 Types: Block, not Normal
  Individual: B2 Types: Block DifferentFrom: B1
            %% B1 and B2 are different blocks
            %% B1 is abnormal
  Class: Ontable
  Class: NormalBlock
        EquivalentTo: Block and Normal
        SubClassOf: Ontable
        %% Normally, a block is on the table
  maximize Normal vars Ontable BlockNotAbnormal
then %implied
  Individual: B2 Types: Ontable
     %% B2 is on the table
end
```

2.2 Alignments, Networks and Combinations in DOL and DOWL

DOWL comprises a comprehensive meta-language layer to express different kinds of ontology **alignments**, following the main established semantics, as well as **networks** of alignments.

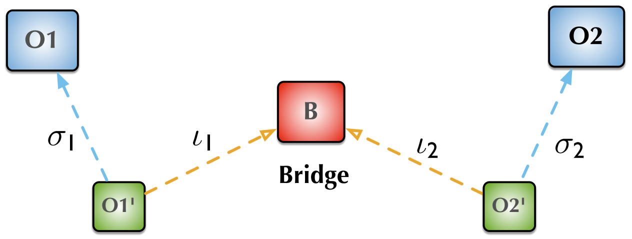
```
alignment A : $O_1$ to $O_2$ =
  $s_1^1$ $REL^1$ $s_2^1$,
  ...,
  $s_1^n$ $REL^n$ $s_2^n$,
  [assuming DOMAIN]
end
```

Fig. 1. Syntax of DOL Alignments

DOL represents the general alignment format introduced by the Alignment API [8] as in Fig. 1 where O_1 and O_2 are the ontologies to be aligned, s_1^i and s_2^i are O_1 and respectively O_2 symbols, for $i = 1, \ldots, n$, and s_1^i REL^i s_2^i is a *correspondence* which identifies a relation between the ontology symbols, either using a relation IRI or a symbol: $>$ (subsumes), $<$ (is subsumed), $=$ (equivalent), % (incompatible), $\in$ (instance) or $\ni$ (has instance). The user can specify the assumption about the universe where the relations in the correspondences are interpreted using the assuming clause, with possible values **SingleDomain** (all ontologies are interpreted over the same universe, which is also the default), **GlobalDomain** (the domains of the ontologies are reconciled w.r.t. a global domain of interpretation) and **ContextualisedDomain** (the domains are connected via relations). DOWL's treatment of bridge axioms in the so-called contextualised semantics closely mirrors the syntax and semantics of C-OWL [5] and DDL [4]. More details of DOL's alignment approach can be found in [6].

We now illustrate how, using *bridge ontologies*, networks of alignments can be transformed into networks of ontology interpretations (morphisms), making them amenable to colimits. Let A be an alignment (using the notations above). The formal relations between the contributing ontologies can be given as a diagram in the shape of a W-alignment (see [25]) where, for **SingleDomain** O_1' and O_2' contain, respectively, all the symbols $O_1{:}s_1$ and $O_2{:}s_2$ that appear in a correspondence s_1 REL s_2 in A, $\sigma_i : O_i' \to O_i$ maps $O_i{:}s$ to s for $i = 1, 2$. B is a *bridge ontology*, whose signature Σ_B is the union of the signatures of O_1' and

O'_2 and whose set of sentences is determined by the union of all sentences that translate the correspondences of A in the underlying logical language.



For OWL, this means that $Class_1 < Class_2$ is translated to $O_1{:}Class_1 \sqsubseteq O_2{:}Class_2$, $Class_1 = Class_2$, to $O_1{:}Class_1 \equiv O_2{:}Class_2$ and so on. The signature morphisms ι_1 and ι_2 are signature inclusions.

Example 1. The foundational ontology (FO) repository Repository of Ontologies for MULtiple USes (ROMULUS)[3] contains alignments between a number of foundational ontologies. We present here the alignment of the FOs DOLCE[4] and BFO[5] using DOL syntax.

```
alignment DolceLite2BFO :
  <http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl>   to
  <http://www.ifomis.org/bfo/1.1> =
 endurant = IndependentContinuant,
 physical-endurant = MaterialEntity,
 physical-object = Object,
 perdurant = Occurrent,
 process = Process,
 quality = Quality,
 spatio-temporal-region = SpatiotemporalRegion,
 temporal-region = TemporalRegion,
 space-region = SpatialRegion
```

The bridge ontology of this alignment will contain only equivalence axioms between the matched symbols.

Another alignment, between Dolce and GFO [14], includes the correspondence `generic-dependent` $<$ `necessary_for`. This introduces in the bridge ontology the axiom `generic-dependent` $\sqsubseteq$ `necessary_for`.

While alignments capture relations between ontologies, interpretations (or ontology morphisms) capture the notion that one ontology can be completely mapped into another one. For example, mereology can be mapped into Euclidean space by interpreting parthood as containment between regions in space. See section 4.3 for further examples. The network construct itself is an essential ingredient for the idea of combination, which in turn is the fundamental operation enabling a formalisation of conceptual blending.

Networks of OWL ontologies are introduced by the following grammar:

```
NetworkDefn := network NAME = Network
Network ::= NAME* [ excluding NAME* ]
```

[3] See `http://www.thezfiles.co.za/ROMULUS/home.html`
[4] See `http://www.loa.istc.cnr.it/DOLCE.html`
[5] See `http://www.ifomis.org/bfo/`

Here, the NAMEs can name ontologies, alignments, interpretations or other networks. A network is specified as a list of network elements (ontologies, ontology mappings and sub-networks), followed by an optional list of excluded network elements.

DOL also provides means for combining a network of ontologies into a new ontology, such that the symbols related in the network are identified. The syntax of combinations is `combine N` where N is a network. The semantics of such a combination is given in terms of a *colimit*. We refrain from presenting the category-theoretic definition here (which can be found in [1]). The colimit of a network is similar to a disjoint union of its ontologies, with some identifications of shared parts as specified by the morphisms in the network.

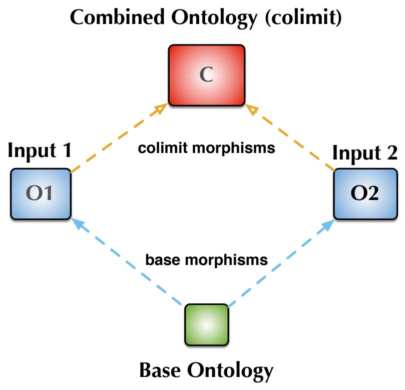


Fig. 2. Combined ontologies.

Fig. 2 shows the colimit of a diagram consisting of two morphisms with a common source. The colimit identifies the symbols of O_1 and O_2 that have a common origin in the base ontology and keeps distinct the symbols that do not share in the base. We can now put together the alignments between DOLCE and BFO and respectively DOLCE and GFO into one network:

```
network SpaceNetwork =
  DolceLite2BFO , DolceLite2GFO
```

We then can combine DolceLite and BFO taking into account the semantic relations specified in the alignment DolceLite2BFO given above:

```
ontology DOLCELiteAndBFO =
  combine DolceLite , BFO , DolceLite2BFO
```

The ontology combining the network of DolceLite2GFO will contain the axioms of DolceLite and BFO as well as the bridge axioms of the alignment between them.

3 Use Case 1: Ontology Design Patterns in DOWL

Ontology Design Patterns (ODP) are solutions for reoccurring ontology modelling situations. [10] While there is a broad range of ODPs, many of the proposed ODPs are basically bits of OWL code. One challenge for their adoption was that there is no easy way to combine ODPs, or to integrate them with existing ontologies. DOWL provides solutions for these problems.

Let us consider an example of a popular ODP: the *reification of relations as events*. Within Semantic Web research contexts, this strategy is often employed because both RDF and OWL do not support n-ary relationships directly.[6] One use case is the representation of relationships that change over time.

[6] http://www.w3.org/TR/swbp-n-aryRelations/

We present below a simplified version of an ODP for temporally changing relationships.

```
Prefix: : <http://ex.com/odp/basicEvent#>
Ontology: <http://ex.com/odp/basicEvent>

Class: Occurrent
Class: Continuant  DisjointWith: Occurrent
Class: Time

ObjectProperty: has_agent   Domain: Occurrent  Range: Continuant
ObjectProperty: has_patient Domain: Occurrent  Range: Continuant
ObjectProperty: has_start_time  Range: Time
ObjectProperty: has_end_time    Range: Time

Class: DomainPTN SubClassOf: Continuant
Class: RangePTN  SubClassOf: Continuant

Class: ReifiedRelationPTN
        SubClassOf: has_agent exactly 1 DomainPTN
        SubClassOf: has_patient exactly 1 RangePTN
        SubClassOf: has_start_time exactly 1 Time
        SubClassOf: has_end_time exactly 1 Time
```

This pattern involves a reified relationship (a class) and two additional classes (DomainPTN, RangePTN), which correspond to the domain and the range of the non-reified relationship. These three classes are basically schematic placeholders within the ODP. The ODP is instantiated by replacing them with 'real' classes.

DOWL allows the reuse and modification of ODPs. For example, assuming we wanted to instantiate it for the 'loves' relationship. The following DOWL code defines the lovesOntology as an instantiation of the Basic-Event-ODP, where we have 'love events', which involves two people:

```
ontology lovesOntology = <http://ex.com/odp/basicEvent> with
  ReifiedRelationPTN |-> Love , DomainPTN |-> Person ,  RangePTN |-> Person
```

The resulting ontology contains all the axioms of the original ontology except that the generic pattern symbols have been replaced by *Love* and *Person*. Thus, the lovesOntology contains axioms like:

```
 Class: Person  SubClassOf: Continuant
 Class: Love    SubClassOf: has_agent exactly 1 Person
                    SubClassOf: has_patient exactly 1 Person
```

The lovesOntology inherits the properties `has_agent` and `has_patient` from the ODP. DOWL also allows the replacement of these generic properties by more pertinent ones. E.g., we may define the lovesOntologyv2 as the ontology that is the result of replacing

```
ontology lovesOntologyv2 = lovesOntology with
  has_agent  |-> has_lover ,   has_patient  |-> has_lovee
```

As one would expect, the lovesOntologyv2 contains the proper declarations of the object properties (with their domain and ranges) and the revised axioms:

```
 Class: Love
        SubClassOf: has_lover exactly 1 Person
        SubClassOf: has_lovee exactly 1 Person
```

4 Use Case 2: Conceptual Blending in DOWL

4.1 Conceptual Blending

Conceptual Blending is a theory for the cognitive process behind creative thinking and generation of novelty [9, 24]. The idea is that novel concepts are created when already known, and potentially conflicting, mental spaces are merged into a blended space, which, due to the unique combination of information, exhibits emergent properties. For example, the input concepts *mother* and *ship* may be blended into a new concept *mother ship* (see Fig. 3). According to the theory of conceptual blending, the blending process involves some shared structure, which is identified across the different input concepts (the so-called *base space*). The blended concept inherits the shared features from the base space and selected features from the input spaces.

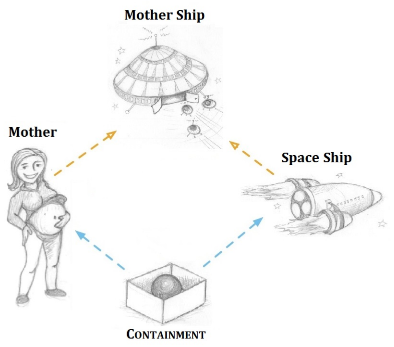


Fig. 3. The blending of mother ship

As mental spaces can be rich in information, the blends can take as many shapes as there are possible combinations. While humans can more or less automatically sort out the blends that make sense and are valuable, automatic blending needs guidance to avoid blends with conflicting or useless information.

4.2 Formalised Blending of Ontologies

Conceptual blending has been formalised using an approach based on Goguen's work on *algebraic semiotics* [12]. In this approach, the formal blending process is modelled by a colimit computation, a construction that abstracts the operation of disjoint unions modulo the identification of certain parts specified by the base and the interpretations, as discussed in detail in [11, 18, 17]. Algebraic semiotics does not claim to provide a comprehensive formal theory of blending. Indeed, Goguen and Harrell admit that many aspects of blending, in particular concerning the meaning of the involved notions, as well as the optimality principles for blending, cannot be captured formally. However, the structural aspects *can* be formalised and provide insights into the space of possible blends.

In [17], an approach to computational conceptual blending was presented, which is in the tradition of Goguen's proposal. The inputs for a blending process (input concepts, generic space, mappings between them) can be formally specified in a *blending network* represented in DOWL.

As illustrated in Figure 3, the process of blending involves two input concepts along some shared structure. The input concepts and the shared structure can all be represented as OWL ontologies. Together with the ontology morphisms that identify the shared structure, these ontologies form a DOWL network, which can be combined (compare Figure 2 above). Because the combination usually yields an ontology that contains too much information (often it is even inconsistent), it usually needs to be weakened by removing axioms or by using more sophisticated generalisation or debugging strategies [7].

```
Class: Dog SubClassOf: Mammal
           SubClassOf: has_habitat some Home
           SubClassOf: has_body_shape some QuadrupleShape
           SubClassOf: has_part exactly 2 Hindlegs
           SubClassOf: has_part exactly 2 Forelegs
           SubClassOf: covered_by some Hair

Class: Owl SubClassOf: Bird
           SubClassOf: has_habitat some Forest
           SubClassOf: has_shape some BirdShape
           SubClassOf: has_part exactly 2 Legs
           SubClassOf: has_part exactly 2 Wings
           SubClassOf: has_part exactly 1 Beak
           SubClassOf: covered_by some Feathers
```

Fig. 4. The two input spaces, Dog and Owl, represented as OWL theories.

We now give an example of a conceptual blending network specified in DOWL, and blending the Dog-Owl.

4.3 Blending the Dog-Owl

```
ontology base =
    ObjectProperty: has_habitat    ObjectProperty: has_body_shape
    ObjectProperty: has_part       ObjectProperty: covered_by
    Class: BackLimb               Class: ForeLimb
    Class: Animal  SubClassOf: has_part exactly 2 ForeLimb
                   SubClassOf: has_part exactly 2 BackLimb

interpretation base2dog: base to Dog =
    Animal |-> Dog,        ForeLimb |-> Foreleg,    BackLimb |-> Hindleg

interpretation base2owl: base to Owl =
    Animal |-> Owl,     ForeLimb |-> Wing,    BackLimb |-> Leg

ontology initialblend =
    {combine base, Dog, Owl, base2dog, base2owl} with Animal |-> Dowl

ontology dowlblend = initialblend reject
    {Class: Dowl
       SubClassOf: Bird
       SubClassOf: has_body_shape some QuadrupleShape
       SubClassOf: has_habitat some Home
       SubClassOf: covered_by some Feathers}
```

Fig. 5. Base and Interpretations of the Dowl blend

To demonstrate the idea, we illustrate how two animals, a dog and an owl, can be merged into a monster of sorts, a 'dowl'. The input spaces are represented as simplified OWL ontologies in Figure 4. Naturally, the concepts are not fully represented, but the formalisations capture some of the important features of the animals. The base ontology contains information shared between the input spaces. Two interpretations map the base ontology onto the input spaces. See Fig. 5 for a formal representation of the base ontology and the interpretations to the input spaces. The ontology *initialblend* consists of the disjoint union of all the features from the input spaces modulo the shared features from the base space. Thus, in *initialblend* the blended concept *Dowl* is an animal that has two forelimbs and two backlimbs, which is covered by hair and feathers, lives both in homes and in the forest and has both the shape of a bird and a quadruped. To achieve a reasonable concept, we define a second ontology, *dowlblend*, where we selectively weaken *initialblend* by rejecting certain axioms.

```
Class: Dowl  SubClassOf: Mammal
             SubClassOf: has_habitat some Forest
             SubClassOf: has_body_shape some BirdShape
             SubClassOf: has_part exactly 2 HindLeg
             SubClassOf: has_part exactly 2 Wing
             SubClassOf: covered_by some Hair
```

Fig. 6. The blended concept Dowl.

The resulting ontology contains a new concept: a birdlike mammal with hair living in the forest (see Figure 6). Note that the resulting concept combines aspects of the original concepts selectively, which is something that could not be done in OWL. Naturally, we could choose a number of different combinations. Here, evaluation of the blends is essential and needs to be connected not only to logical consistency, but to a consideration of rich background knowledge ontologies that can help ensure the quality of the blends.

5 Discussion and Outlook

The blending diagrams (or networks) can be analysed and computed by the *Heterogeneous Tool Set* HETS, a proof management system. HETS is integrated into Ontohub[7], an ontology repository which allows users to manage and collaboratively work on ontologies. DOL, DOWL, HETS, and Ontohub provide a powerful set of tools making it easy to specify and computationally execute conceptual blends, as discussed in [16, 22]. Moreover, the structuring mechanisms of DOWL allow a new systematic approach to designing and reusing ontology design patterns in OWL, and to re-organise existing ontology patterns. An extensive introduction to the features and the formal semantics of the full DOL language can be found in [21].

Reasoning about DOWL ontologies and networks in many cases can be reduced to reasoning in OWL DL by using HETS for a) flattening out the structuring constructs, which means computing an equivalent basic ontology for any structured ontology, and b) taking combinations (colimits) of networks, also resulting in a flat OWL DL ontology. Concerning reasoning about the different forms of reduction, the easiest one is *select* and *reject*, which again can be flattened out. For *extract* and *remove*, module extraction methods already provide flat ontologies. For *hide* and *reveal*, the hiding can be uncovered (using colimits), also resulting in a flat OWL DL ontology.

Future work concerns reasoning about DOWL ontologies that cannot be flattened. In the case of *keep* and *forget*, this is addressed in current research about forgetting and uniform interpolation [20]. Likewise, *minimize* and *maximize* are difficult as well; they are related to fixpoints [2].

Acknowledgments. The project COINVENT acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant number: 611553.

[7] `www.ontohub.org`

References

1. J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. Wiley, New York, 1990.
2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
3. Piero A. Bonatti, Carsten Lutz, and Frank Wolter. The complexity of circumscription in DLs. *J. Artif. Intell. Res. (JAIR)*, 35:717–773, 2009.
4. A. Borgida and L. Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *Journal of Data Semantics*, 1:153–184, 2003.
5. Paolo Bouquet, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-owl: Contextualizing ontologies. In *The Semantic Web-ISWC 2003*, pages 164–179. Springer, 2003.
6. M. Codescu, T. Mossakowski, and O. Kutz. A Categorical Approach to Ontology Alignment. In *Proc. of the 9th International Workshop on Ontology Matching (OM-2014)*, ISWC-2014, Riva del Garda, Trentino, Italy, 2014. CEUR-WS.
7. Roberto Confalonieri, Marco Schorlemmer, Enric Plaza, Manfred Eppe, Oliver Kutz, and Rafael Peñaloza. Upward Refinement for Conceptual Blending in Description Logic: An ASP-based Approach and Case Study in EL++. In Odile Papini, Salem Benferhat, Laurent Garcia, Marie-Laure Mugnier, Eduardo L. Ferm, Thomas Meyer, Renata Wassermann, Torsten Hahmann, Ken Baclawski, Adila Krisnadhi, Pavel Klinov, Stefano Borgo, Oliver Kutz, and Daniele Porello, editors, *JOWO@IJCAI*, volume 1517 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.
8. Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The alignment API 4.0. *Semantic Web*, 2(1):3–10, 2011.
9. Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.
10. Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on ontologies*, pages 221–243. Springer, 2009.
11. Joseph Goguen. Semiotic morphisms, representations and blending for interface design. In *In Proceedings, AMAST Workshop on Algebraic Methods in Language Processing*, pages 1–15. AMAST Press, 2003.
12. Joseph A. Goguen. An Introduction to Algebraic Semiotics, with Applications to User Interface Design. In Chrystopher L. Nehaniv, editor, *Computation for Metaphors, Analogy and Agents*, number 1562 in Lecture Notes in Computer Science, pages 242–291. Springer, 1999.
13. Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus. Image schemas in computational conceptual blending. *Cognitive Systems Research*, 2016. In print.
14. Heinrich Herre. General Formal Ontology (GFO) : A Foundational Ontology for Conceptual Modelling. In Roberto Poli and Leo Obrst, editors, *Theory and Applications of Ontology*, volume 2. Springer, Berlin, 2010.
15. Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Logic-based ontology comparison and module extraction, with an application to dl-lite. *Artif. Intell.*, 174(15):1093–1141, 2010.
16. Oliver Kutz, John Bateman, Fabian Neuhaus, Till Mossakowski, and Mehul Bhatt. E pluribus unum: Formalisation, Use-Cases, and Computational Support for Conceptual Blending. In Tarek R. Besold, Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, Thinking Machines. Atlantis/Springer, 2014.

17. Oliver Kutz, Till Mossakowski, Joana Hois, Mehul Bhatt, and John Bateman. Ontological Blending in DOL. In Tarek R. Besold, Kai-Uwe Kühnberger, Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity, Concept Invention, and General Intelligence, Proc. of the 1st International Workshop C3GI@ECAI*, volume 01, Montpellier, France, August 2012. Publications of the Institute of Cognitive Science, Osnabrück.
18. Oliver Kutz, Till Mossakowski, and Dominik Lücke. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. *Logica Universalis*, 4(2):255–333, 2010. Special Issue on 'Is Logic Universal?'.
19. V. Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1994.
20. Carsten Lutz and Frank Wolter. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In Toby Walsh, editor, *IJCAI*, pages 989–995. IJCAI/AAAI, 2011.
21. Till Mossakowski, Mihai Codescu, Fabian Neuhaus, and Oliver Kutz. *The Road to Universal Logic–Festschrift for 50th birthday of Jean-Yves Beziau, Volume II*, chapter The distributed ontology, modelling and specification language - DOL. Studies in Universal Logic. Birkhäuser, 2015.
22. Fabian Neuhaus, Oliver Kutz, Mihai Codescu, and Till Mossakowski. Fabricating Monsters is Hard - Towards the Automation of Conceptual Blending. In *Proc. of Computational Creativity, Concept Invention, and General Intelligence (C3GI-14)*, volume 1-2014, pages 2–5, Prague, 2014. Publications of the Institute of Cognitive Science, Osnabrück.
23. Object Management Group. The distributed ontology, modeling, and specification language (DOL), 2015. Draft answer to RFP available at `https://ontoiop.org`.
24. Mark Turner. *The Origin of Ideas: Blending, Creativity, and the Human Spark*. Oxford University Press, 2014.
25. Antoine Zimmermann, Markus Krötzsch, J. Euzenat, and Pascal Hitzler. Formalizing Ontology Alignment and its Operations with Category Theory. In *Proc. of FOIS-06*, pages 277–288, 2006.