Blending in the Hub
Towards a computational concept invention platform

Oliver Kutz and **Fabian Neuhaus** and **Till Mossakowski** and **Mihai Codescu**

Institute of Knowledge and Language Engineering, Otto-von-Guericke University of Magdeburg, Germany

Abstract

Conceptual blending has been employed very successfully to understand the process of concept invention, studied particularly within cognitive psychology and linguistics. However, despite this influential research, within computational creativity little effort has been devoted to fully formalise these ideas and to make them amenable to computational techniques. We here present the basic formalisation of conceptual blending, as sketched by the late Joseph Goguen, and show how the Distributed Ontology Language DOL can be used to declaratively specify blending diagrams. Moreover, we discuss in detail how the workflow and creative act of generating and evaluating a new, blended concept can be managed and computationally supported within Ontohub, a DOL-enabled theory repository with support for a large number of logical languages and formal linking constructs.

Concept Invention via Blending

In the general methodology of conceptual blending introduced by Fauconnier and Turner (2003), the blending of two thematically rather different *conceptual spaces* yields a new conceptual space with emergent structure, selectively combining parts of the given spaces whilst respecting common structural properties.[1] The 'imaginative' aspect of blending is summarised as follows in Turner (2007):

> [...] the two inputs have different (and often clashing) organising frames, and the blend has an organising frame that receives projections from each of those organising frames. The blend also has emergent structure on its own that cannot be found in any of the inputs. Sharp differences between the organising frames of the inputs offer the possibility of rich clashes. Far from blocking the construction of the network, such clashes offer challenges to the imagination. The resulting blends can turn out to be highly imaginative.

A classic example for this is the blending of the concepts *house* and *boat*, yielding as most straightforward blends the concepts of a *houseboat* and a *boathouse*, but also an *amphibious vehicle* (Goguen and Harrell, 2009).

In the almost unlimited space of possibilities for combining existing ontologies to create new ontologies with emergent structure, conceptual blending can be built on to provide a structural and logic-based approach to 'creative' ontological engineering. This endeavour primarily raises the following two challenges: (1) when combining the terminologies of two ontologies, the shared semantic structure is of particular importance to steer possible combinations. This shared semantic structure leads to the notion of base ontology, which is closely related to the notion of 'tertium comparationis' found in the classic rhetoric and poetic theories, but also in more recent cognitive theories of metaphor (see, e.g., Jaszczolt (2003)); (2) having established a shared semantic structure, there is typically still a huge number of possibilities that can capitalise on this information in the combination process: here, structural optimality principles as well as ontology evaluation techniques take on a central role in selecting interesting blends.

We believe that the principles governing ontological blending are quite distinct from the rather informal principles employed in blending phenomena in language or poetry, or the rather strict principles ruling blending in mathematics, in particular in the way formal inconsistencies are dealt with. For instance, whilst blending in poetry might be particularly inventive or imaginative when the structure of the basic categories found in the input spaces is almost completely ignored, and whilst the opposite, i.e., rather strict adherence to sort structure, is important in areas such as mathematics in order to generate meaningful blends[2], ontological blending is situated somewhere in the middle: re-arrangement and new combination of basic categories can be rather interesting, but has to be finely controlled through corresponding interfaces, often regulated by or related to choices found in foundational or upper ontologies.

[1] The usage of the term 'conceptual space' in blending theory is not to be confused with the usage established by Gärdenfors (2000).

[2] For instance when creating the theory of transfinite cardinals by blending the perfective aspect of counting up to any fixed finite number with the imperfective aspect of 'endless counting' (Núñez, 2005).

The core contributions of the paper can be summarised as follows.[3] We:

- sketch the logical analysis of conceptual blending in terms of blending diagrams and colimits, as originally proposed by Joseph Goguen, and give an abstract definition of ontological blendoids capturing the basic intuitions of conceptual blending in the ontological setting;
- provide a formal language for declaratively specifying blending diagrams by employing the OWL[4] fragment of the distributed ontology language DOL for blending. This provides a structured approach to ontology languages and combines the simplicity and good tool support for OWL with the more complex blending facilities of OBJ3 (Goguen and Malcolm, 1996) or Haskell (Kuhn, 2002);
- discuss the capabilities of the Ontohub/Hets ecosystem with regard to collaboratively managing, creating, and evaluating blended concepts and theories; this includes an investigation of the evaluation problem in blending, together with a discussion of structural optimality principles and current automated reasoning support.

We close with a detailed discussion of open problems and future work.

Blending Computationalised

Goguen has created the field of *algebraic semiotics* which logically formalises the structural aspects of semiotic signs, sign systems, and their mappings (Goguen, 1999). In Goguen and Harrell (2009), algebraic semiotics has been applied to user interface design and blending. 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.

Goguen defines semiotic systems to be algebraic theories that can be formulated by using the algebraic specification language OBJ (Goguen and Malcolm, 1996). Moreover, a special case of a semiotic system is a *conceptual space*: it consists only of constants and relations, one sort, and axioms that define that certain relations hold on certain instances.

As we focus on standard ontology languages, namely OWL and first-order logic, we here replace the logical language OBJ. As structural aspects in the ontology language are necessary for blending, we augment these languages with structuring mechanisms known from algebraic specification theory (Kutz et al., 2008). This allows to translate most parts of Goguen's theory to these ontology languages. Goguen's main insight has been that semiotic systems and conceptual spaces can be related via *morphisms*, and that blending is comparable to *colimit computation*, a construction that abstracts the operation of disjoint unions modulo

[3]This paper elaborates on ideas first introduced in Hois et al. (2010); detailed technical definitions are given in Kutz et al. (2012).

[4]With 'OWL' we refer to OWL 2 DL, see `http://www.w3.org/TR/owl2-overview/`

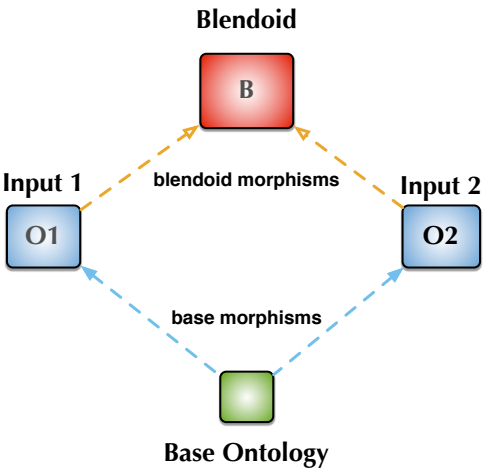

Figure 1: The basic integration network for blending: concepts in the base ontology are first refined to concepts in the input ontologies and then selectively blended into the blendoid.

the identification of certain parts, explained in more detail below. In particular, the blending of two concepts is often a *pushout* (also called a *blendoid* in this context).

Some basic definitions:[5] Non-logical symbols are grouped into **signatures**, which for our purposes can be regarded as collections of kinded symbols (e.g. concept names, relation names). **Signature morphisms** are maps between signatures that preserve (at least) kinds of symbols (i.e. map concept names to concept names, relations to relations, etc.). A **theory** or **ontology** pairs a signature with a set of sentences over that signature, and an **theory morphism** (or **interpretation**) between two theories is just a signature morphism between the underlying signatures that preserves logical consequence, that is, $\rho : T_1 \to T_2$ is a theory morphism if $T_2 \models \rho(T_1)$, i.e. all the translations of sentences of T_1 along ρ follow from T_2. This construction is completely logic independent.

Signature/theory morphisms are an essential ingredient for describing conceptual blending in a logical way.

We now give a general definition of ontological blending capturing the basic intuition that a blend of input ontologies shall partially preserve the structure imposed by base ontologies, but otherwise be an almost arbitrary extension or fragment of the disjoint union of the input ontologies with appropriately identified base space terms.

For the following definition, which we first introduced in Kutz et al. (2012), a diagram consists of a set of ontologies and a set of morphisms between them. The **colimit** of a diagram is similar to a disjoint union of its ontologies, with some identifications of shared parts as specified by the morphisms in the diagram. We refrain from presenting the category-theoretic definition here (which can be found in

[5]Note that these definitions apply to OWL, but also to many other logics. Indeed, they apply to any logic formalised as an *institution* (Goguen and Burstall, 1992).

Adámek, Herrlich, and Strecker (1990)), but explain the colimit operation using the examples below.

Definition 1 (Ontological Base Diagram) *An **ontological base diagram** is a diagram D for which the minimal nodes $(B_i)_{i \in D_{min} \subseteq |D|}$ are called **base ontologies**, the maximal nodes $(I_j)_{j \in D_{max} \subseteq |D|}$ called **input ontologies**, and where the theory morphisms $\mu_{ij} : B_i \rightarrow I_j$ are called the **base morphisms**. If there are exactly two inputs I_1, I_2, and one base $\mathcal{B}$, the diagram D is called **classical** and has the shape of a V. In this case, $\mathcal{B}$ is also called the **tertium comparationis**.*

Fig. 1 illustrates the basic, classical case of an ontological blending diagram. The lower part of the diagram shows the base space (tertium), i.e. the common generalisation of the two input spaces, which is connected to these via total (theory) morphisms, the base morphisms. The newly invented concept is at the top of this diagram, and is computed from the base diagram via a colimit. More precisely, any consistent subset of the colimit of the base diagram may be seen as a newly invented concept, a **blendoid** (a more precise definition of this notion is given in Kutz et al. (2012)). Note that, in general, ontological blending can deal with more than one base and two input ontologies.

Computing the Tertium Comparationis

To find candidates for base ontologies that could serve for the generation of ontological blendoids, much more shared semantic structure is required than the surface similarities that alignment approaches rely on. The common structural properties of the input ontologies that are encoded in the base ontology are typically of a more abstract nature. The standard example here relies on *image schemata*, such as the notion of a *container* (see e.g. Kuhn (2002)). Thus, in particular, foundational ontologies can support such selections. In analogical reasoning, 'structure' is (partially) mapped from a source domain to a target domain (Forbus, Falkenhainer, and Gentner, 1989; Schwering et al., 2009). Therefore, intuitively the operation of computing a base ontology can thus be seen as a bi-directional search for analogy or generalisation into a base ontology together with the corresponding mappings. Providing efficient means for finding a number of suitable such candidate generalisations is essential to making the entire blending process computationally feasible. Consider the example of blending 'house' with 'boat' discussed below in detail: even after fixing the base ontology itself, guessing the right mappings into the input ontologies means guessing within a space of approximately 1.4 Billion signature morphisms. Three promising candidates for finding generalisations are:

(1) **Ontology intersection:** Normann (2008) has studied the automatisation of theory interpretation search for formalised mathematics, implemented as part of the Heterogeneous Tool Set (HETS, see below). Kutz and Normann (2009) applied these ideas to ontologies by using the ontologies' axiomatisations for finding their shared structure. Accidental naming of concept and role names is deliberately ignored and such names are treated as arbitrary symbols (i.e., any concept may be matched with any other). By computing mutual theory interpretations between the inputs, the method allows to compute a base ontology as an *intersection* of the input ontologies together with corresponding theory morphisms. While this approach can be efficiently applied to ontologies with non-trivial axiomatisations, lightweight ontologies are less applicable, e.g., 'intersecting' a smaller taxonomy with a larger one clearly results in a huge number of possible taxonomy matches (Kutz and Normann, 2009). In this case, the following techniques are more appropriate.

(2) **Structure-based ontology matching:** matching and alignment approaches are often restricted to find simple correspondences between atomic entities of the ontology vocabulary. In contrast, work such as (Ritze et al., 2009; Walshe, 2012) focuses on defining a number of *complex correspondence patterns* that can be used together with standard alignments in order to relate complex expressions between two input ontologies. For instance, the 'Class by Attribute Type Pattern' may be employed to claim the equivalence of the atomic concept $\mathsf{PositiveReviewedPaper}$ in ontology O_1 with the complex concept $\exists \mathsf{hasEvaluation}.\mathsf{Positive}$ of O_2. Such an equivalence can be taken as an axiom of the base ontology; note, however, that it could typically not be found by intersecting the input ontologies. Giving such a library of design patterns may be seen as a variation of the idea of using image schemata.

(3) **Analogical Reasoning:** *Heuristic-driven theory projection* is a logic-based technique for analogical reasoning that can be employed for the task of computing a common generalisation of input theories. Schwering et al. (2009) establish an analogical relation between a source theory and a target theory (both first-order) by computing a common generalisation (called 'structural description'). They implement this by using anti-unification (Plotkin, 1970). A typical example is to find a generalisation (base ontology) formalising the structural commonalities between the Rutherford atomic model and a model of the solar system. This process may be assisted by a background knowledge base (in the ontological setting, a related domain or foundational ontology). Indeed, this idea has been further developed in Martinez et al. (2011).

Selecting the Blendoids: Optimality Principles

Having a common base ontology (computed or given), there is typically a large number of possible blendoids. For example, even in the rather simple case of combining House and Boat, allowing for blendoids which only partially maintain structure (called *non-primary* blendoids in Goguen and Harrell (2009)), i.e., where any subset of the axioms may be propagated to the resulting blendoid, the number of possible blendoids is in the magnitude of 1000. Clearly, from an ontological viewpoint, the overwhelming majority of these candidates is rather meaningless. A ranking therefore needs to be applied on the basis of specific ontological principles. In conceptual blending theory, a number of **optimality principles** are given in an informal and heuristic style (Fauconnier and Turner, 1998, 2003). While they provide useful guidelines for evaluating natural language blends, they do not suggest a direct algorithmic implementation, as also

analysed in Goguen and Harrell (2009). However, the importance of designing computational versions of optimality principles has been realised early on, and one such attempt may be found in the work of Pereira and Cardoso (2003), who proposed an implementation of the eight optimality principles presented in Fauconnier and Turner (1998) based on quantitative metrics for their more lightweight logical formalisation of blending. Such metrics, though, are not directly applicable to more expressive languages such as OWL or first-order logic. Moreover, the standard blending theory of Fauconnier and Turner (2003) does not assign types, which might make sense in the case of linguistic blends where type information is often ignored. A typical example of a type mismatch in language is the operation of *personification*, e.g., turning a boat into an 'inhabitant' of the 'boathouse'. However, in the case of blending in mathematics or ontology, this loss of information is often rather unacceptable: on contrary, a fine-grained control of type or sort information may be of the utmost importance.

Optimality principles for ontological blending are of two kinds.

(1) purely **structural/logical principles**: these extend and refine the criteria as given in Goguen and Harrell (2009), namely *degree of commutativity* of the blend diagram, *type casting* (preservation of taxonomical structure), *degree of partiality* (of signature morphisms), and *degree of axiom preservation*. In the context of OWL, typing needs to be replaced with preservation of specific axioms encoding the taxonomy.

(2) **heuristic principles**: these include introducing preference orders on morphisms (an idea that Goguen labelled $3/2$ pushouts (Goguen, 1999)) reflecting their 'quality' e.g. measured in terms of degree of type violation; specific ontological principles, e.g. adherence to the OntoClean methodology (Guarino and Welty, 2002), or general ontology evaluation techniques such as competency questions, further discussed below. Another set of heuristics is quantitative, statistical metrics, similar in style to those proposed in Pereira and Cardoso (2003).

The Distributed Ontology Language DOL

The distributed ontology language DOL is an ideal formal language for specifying both ontologies, base diagrams, and their blends. DOL is a metalanguage in the sense that it enables the reuse of existing ontologies (written in some ontology language like OWL or Common Logic) as building blocks for new ontologies and, further, allows to specify intended relationships between ontologies. One important feature of DOL is the ability to combine ontologies that are written in different languages without changing their semantics. DOL is going to be submitted as response to the Object Management Group's (OMG) Ontology, Model and Specification Integration and Interoperability (OntoIOp) Request For Proposal.[6]

[6] http://www.omg.org/cgi-bin/doc?ad/2013-12-02

In this section, we introduce DOL only informally. A formal specification of the language and its model theoretic semantics can be found in Mossakowski et al. (2013).

For the purpose of ontology blending the following features of DOL are relevant:

- **DOL library**. A DOL library consists of basic and structured ontologies and ontology interpretations. A basic ontology is an ontology written in some ontology language (e.g., OWL or Common Logic). A structured ontology builds on basic ontologies with the help of ontology translations, ontology unions, and symbol hiding.
- **ontology translation** (written O_1 **with** σ). A translation takes an ontology O_1 and a renaming function (technically, signature morphism) σ. The result of a translation is an ontology O_2, which differs from the ontology O_1 only by substituting the symbols as specified by the renaming function.
- **ontology union** (written O_1 **and** O_2). The union of two ontologies O_1 and O_2 is a new ontology O_3, which combines the axioms of both ontologies.
- **symbol hiding** (written O_1 **hide** $\{s_1, ..., s_n\}$). A symbol hiding takes an ontology O_1 and a set of symbols $s_1, ..., s_n$. The result of the hiding is a new ontology O_2, which is the result of 'removing' the symbols $s_1, ..., s_n$ from the signature of ontology O_1. Nevertheless, O_2 keeps all semantic constraints from O_1.[7]
- **ontology interpretation** (written **interpretation** $INT\_NAME : O_1$ to $O_2 = \sigma$). An ontology interpretation is a claim about the relationship between two ontologies O_1 and O_2, giving some renaming function σ. It states that all the constraints that are the result of translating O_1 with σ can be proven by O_2.

Some additional features that are necessary for blending will be introduced in the next section.

Formalising Blending in DOL

The novelty proposed by DOL is that the user can specify the base diagram of the blendoid. This is a crucial task, as the resulting blendoid depends on the dependencies between symbols that are stored in the diagram. Ontohub, our web platform and repository engine for managing distributed heterogeneous ontologies and discussed in more detail below, is able to use the specification of a base diagram to automatically generate the colimit-blendoid. In this section, we illustrate the specification of base diagrams in DOL and the resulting blendoids by blending house and boat to houseboat and boathouse.

The main inputs for the blendings consist of two ontologies, one for HOUSE and the other for BOAT. We adapted them from Goguen and Harrell (2009) but gave a stronger axiomatisation, making them more realistic. The purpose of this exercise is to show, using this classic blend, that our

[7] By approximation, one could consider O_2 as the ontology that is the result of existentially quantifying $s_1, ..., s_n$ in O_1.

framework allows to blend in a generic way complex ontological theories, thus not being restricted theoretically to any particular domain or even logical language.

Fig. 2 shows the ontology for HOUSE in OWL Manchester Syntax.[8]

```
Class: Artifact
Class: Capability
ObjectProperty: has_function
    Range: Capability
ObjectProperty: executes
    Range: Capability
ObjectProperty: is_located_on
Class: Person
Class: Plot
ObjectProperty: is_inhabited_by
    Domain: House
    Range:  Person
Class: ServeAsResidence
    SubClassOf: Capability
Class: ArtifactThatExecutesResidenceFunction
    EquivalentTo: Artifact that executes
                  some ServeAsResidence
    SubClassOf: is_inhabited_by some Person
Class: House
    SubClassOf: Artifact
        that is_located_on some Plot
        and has_function some
                    ServeAsResidence
```

Figure 2: Ontology `House`

As discussed above, finding candidate base ontologies and base morphisms is a non-trivial task. For the purpose of this example, we created them manually. The base ontologies are both quite simple, they mostly introduce shared concepts and contain only weak axiomatisations. The second base ontology only differs from the first by replacing the class `Agent` by `Person` and two additional classes, namely `Object` and `Site`.

```
ontology base1 =
    Class: Artifact  [...]  Class: Agent
     end

ontology base2 =
    Class: Artifact [...]  Class: Person
    Class: Object       Class: Site
    end
```

The blending of boat and house to boathouse is achieved by turning the boat into a habitat and moving the house from a plot of land to a body of water. This can be represented by two interpretations `boat_habitable` and `house_floating`.

[8] In the examples, note that concepts such as 'ArtifactThatExecutesResidenceFunction' are auxiliary symbols that are needed because of limitation of the Manchester Syntax being used, which does not allow to use complex concepts on the left-hand side of subsumption statements. The ontology for BOAT is axiomatized similarly, it can be found at `http://www.ontohub.org/repositories/conceptportal`.

```
interpretation boat_habitable : base2 to Boat =
    Object |-> Boat,
    Site |-> BodyOfWater

interpretation house_floating : base2 to House =
    Object |-> House,
    Site |-> Plot
```

The base ontologies and the interpretations above provide the necessary ingredients for a blending of BOAT and HOUSE to BOATHOUSE. The syntax of combinations is

combine $O_1, \ldots, O_m,\ M_1, \ldots, M_n$

where the O_i are ontologies, and M_i are morphism names. The semantics of combinations is the colimit of the generated diagram. A colimit involves both pasting together (technically: disjoint union) and identification of shared parts (technically: a quotient).

In our example, houseboat can be defined by the colimit based on the interpretations. To make the result easier to read, some of the classes are renamed:

```
ontology house_boat =
    combine boat_habitable, house_floating
    with Object |-> HouseBoat, Site |-> BodyOfWater
```

Ontohub is able to compute the colimit, which combines both the boat and house ontologies along the morphism. The colimit inherits most of the axioms of the ontologies and the base. Here we just show the declaration of the blended class `Houseboat`:

```
Class: HouseBoat
  SubClassOf: Artifact
    and has_function some MeansOfTransportation
    and has_function some Floating
    and is_navigated_by some Agent
  SubClassOf: Artifact
    and is_located_on some BodyOfWater
    and has_function some ServeAsResidence
```

In the case of blending of BOAT and HOUSE to BOATHOUSE, the crucial part in this blend is to view a boat as a kind of "person" that lives in a house. The two ontologies `House` and `Boat` presented above can be blended by selecting a base, which here provides (among others) a class `Agent`, and two interpretations, mapping `Agent` to `Boat` and `Person`, respectively. In this way, we let a boat play the role of a person (that inhabits a house).

```
interpretation boat_personification :
    base1 to Boat =
    Agent |-> Boat

interpretation house_import :
base1 to House =
    Agent |-> Person

ontology boat_house =
 combine boat_personification, house_import
 with Agent ↦ Boat, House ↦ BoatHouse
```

As before, Ontohub is able to compute the colimit. As above, we present here only the relevant declarations of the blended concept.

```
Class: BoatHouse
  SubClassOf: Artifact
    and is_located_on some Plot
    and has_function some ServeAsResidence
Class: ArtifactThatExecutesResidenceFunction
       EquivalentTo: Artifact
           and executes some ServeAsResidence
       SubClassOf: is_inhabited_by some Boat
```

Of course, the possibilities for blending the two concepts do not stop here. For example, we could map the `agent` in the base ontology to `person` in the boat ontology. This can be achieved by first defining an additional interpretation and by blending all three interpretations.

```
interpretation boat_import :
    base1 to Boat =
        Agent |-> Person

ontology boat_house =
        combine boat_personification, house_import,
boat_import
 with Agent ↦ Boat, House ↦ BoatHouse
```

The resulting blendoid is consistent, but it contains some strange consequences. For example, in the blendoid boats are driven by boats. However, if we are interested both in hosting boats and a hub for autonomous vehicles, this would count as an interesting result. In general, whether such more creative aspects of blendoids are desirable or not will depend on the context of the blending. We will address this issue in the section on evaluation below.

Blending in the Hub

Representation and Computation

Indeed, combinations and colimits can be computed by our web platform Ontohub. Ontohub is a repository engine for managing distributed heterogeneous ontologies. Ontohub supports a wide range of formal logical and ontology languages and allows for complex inter-theory (concept) mappings and relationships with formal semantics, as well as ontology alignments and blending. Ontohub understands various input languages, among them OWL and DOL.

We describe the basic design and features of Ontohub in general, and outline the extended feature-set that we pursue for conceptportal.org - a specialised repository within the distributed ontohub architecture.

The back-end of Ontohub is the Heterogeneous Tool Set HETS, which is used by Ontohub for parsing, static analysis and proof management of ontologies. HETS can also compute colimits of OWL diagrams and even approximations of colimits in the case where the input ontologies live in different ontology languages (Codescu and Mossakowski, 2008).

Computation of colimits in HETS is based on HETS' general colimit algorithm for diagrams of sets and functions (note that signatures in most cases are structured sets, and signature morphisms structure preserving functions). Such a colimit of sets and functions is computed by taking the disjoint union of all sets, and quotienting it by the equivalence relation generated by the diagram, which more precisely is obtained by the rule that given any element x of an involved set, any images of x under the involved functions are identified. The quotient is computed by selecting a representative of each equivalence class.

A difficulty that arises is that we have to make a choice of these representatives, and therefore of names for the symbols in the colimit, as a symbol may be not always identically mapped in the base diagram of the blendoid. The convention in HETS is that in case of ambiguity, the name of the symbol is chosen to be the most frequently occurring one. This gives the user control over the namespace, such that the symbols of the colimit can be later renamed. We can see this for our boathouse example above, where `Agent` appears most often in the diagram and therefore the symbol has been explicitly renamed.

Evaluating the Blending Space

Optimality principles, in particular structural ones, can be used to rank candidate blendoids on-the-fly during the ontology blending process. However, even if they improve on existing logical and heuristic methods, optimality principles will only narrow down the potential candidates and not tell us whether the result is a 'successful' blend of the ontologies. For example, assume that we had optimality principles that would show that from the roughly 1000 candidate blendoids of *House* and *Boat* that Goguen computed, only two candidates $\mathfrak{B}_{hb}$ and $\mathfrak{B}_{bh}$ are optimal. Is either $\mathfrak{B}_{hb}$ or $\mathfrak{B}_{bh}$ any good? And, if so, which of them should we use? To answer these question, it seems natural to apply ontology evaluation techniques.

Ontologies are human-intelligible and machine-interpretable representations of some portions and aspects of a domain that are used as part of information systems. To be more specific, ontology is a logical theory written in some knowledge representation language, which is associated with some intended interpretation. The intended interpretation is partially captured in the choice of symbols and natural language text (often in the form of annotations or comments). The evaluation of an ontology covers both the logical theory and the intended interpretation, their relationship to each other, and how they relate to the requirements that are derived from the intended use within a given information system. Therefore, ontology evaluation is concerned not only with formal properties of logical theories (e.g., logical consistency), but, among other aspects, with the *fidelity* of an ontology; that is whether the formal theory accurately represents the intended domain (Neuhaus et al., 2013). For example, if $\mathfrak{B}_{hb}$ is an excellent representation of the concept *houseboat*, then $\mathfrak{B}_{hb}$ provides a poor representation of the concept *boathouses*. Thus, any evaluation of the blend $\mathfrak{B}_{hb}$ depends on what domain $\mathfrak{B}_{hb}$ is intended to represent.

The lesson is that the evaluation of the results of ontology blending is dependent on the intended goal and, more generally, on the requirements that one expects the outcome of the blending process to meet. One way to capture these requirement is similar to competency questions, which are widely used in ontology engineering (Grüninger and Fox, 1995). Competency questions are usually initially captured in natural language, they specify examples for questions that

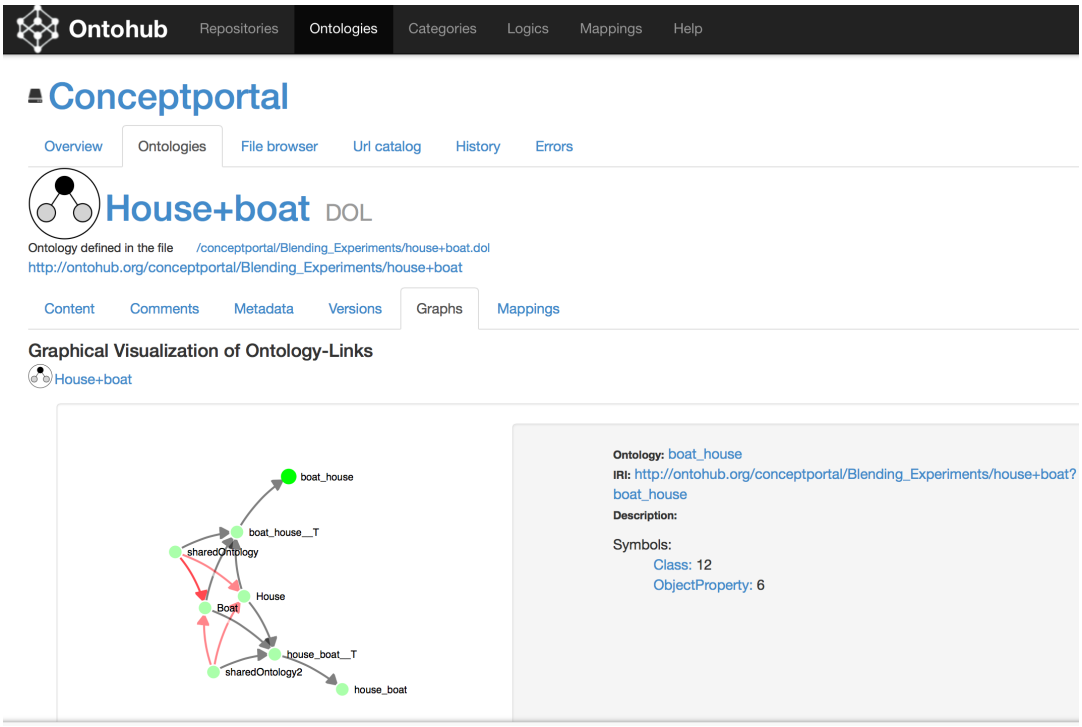

Figure 3: Blendoid representation and colimit computation via Hets/Ontohub: the screenshot of Ontohub shows the heterogeneous ontology house+boat.dol, hosted in the Conceptportal repository. The entire double-blend of house and boat into boathouse and houseboat is shown in the Graph to the left. The red arrows denote the interpretations of the shared ontologies into the blend. The concept boat_house is selected and shown on the right: its theory can be inspected by following the link to the respective ontology specification.

an ontology needs to be able to answer in a given scenario. By formalising the competency questions one can use automatic theorem provers to evaluate whether the ontology meets the intended interpretation.

The requirements that are used to select between the different blends fall, roughly, into two categories. ontological constraints and consequence requirements. Ontological constraints prevent the blends from becoming 'too creative' by narrowing the space for conceptual blending. E.g., it may be desirable to ensure that the `is_inhabited_by` relationship is asymmetric and that `is_navigated_by` is irreflexive. To achieve that any blendoid can be checked for logical consistency with the following ontology:

```
ontology OntologicalConstraints =
  ObjectProperty: is_inhabited_by
    Characteristics: Asymmetric
  ObjectProperty: is_navigated_by
    Characteristics: Irreflexive
```

Given these requirements, any blendoid that involves a house that lives in itself, or any boat navigated by itself (see the blendoid `boat_house1` above) would be discarded.

Consequence requirements specify the kind of characteristics the blendoid is supposed to have. E.g., assume the purpose of the conceptual blending is to find alternative housing arrangements, because high land prices make newly build houses unaffordable. In this case, the requirement could be 'a residence that is not located on a plot of land', which can be expressed in OWL as follows:

```
ontology ConsequenceRequirements =
[...]
  Class PlotFreeResidence
    EquivalentTo: Residence
      and (is_located_on only (not (Plot)))
```

Ontohub allows to use ontological constraints and consequence requirements to evaluate blended concepts automatically. The requirements are managed as DOL files, which allow to express that a given blendoid is logically consistent with a set of ontological constraints or that it entails some consequence requirements. The requirements themselves may be stored as regular ontology files (e.g., in OWL Manchester syntax). Ontohub executes the DOL files with the help of integrated automatic theorem provers, and is able to detect whether a blendoid meets the specified requirements.

At this time, the evaluation of blendoids for ontological constraints and consequence requirements depends on the use of DOL files. We are planning to integrate this functionality into the GUI of Ontohub to make it more convenient for the user.

Another way to evaluate a blendoid is to analyse its structure for typical ontological errors. For this purpose, Ontohub has integrated OOPS!. OOPS! automatically analy-

ses ontologies for common pitfalls, which is developed by the Ontology Engineering Group at the Technical University of Madrid (Poveda-Villalón, Suárez-Figueroa, and Gómez-Pérez, 2012). We are planning to add additional evaluation tools to Ontohub in the future.

Outlook

Our work in this paper follows a research line in which blending processes are primarily controlled through mappings and their properties (Gentner, 1983; Forbus, Falkenhainer, and Gentner, 1989; Veale, 1997; Pereira, 2007). By introducing blending techniques to ontology languages, we have provided a method which allows us to combine two thematically different ontologies into a newly created ontology, the blendoid, describing a novel concept or domain. The blendoid creatively mixes information from both input ontologies on the basis of structural commonalities of the inputs and combines their axiomatisations.

We have illustrated that the tool HETS and the DOL language (Mossakowski et al., 2013) provide an excellent starting point for developing the theory and practice of ontology blending further. They: (1) support various ontology language and their heterogeneous integration (Kutz et al., 2008); (2) allow to specify theory interpretations and other morphisms between ontologies (Kutz, Mossakowski, and Lücke, 2010); (3) support the computation of colimits as well as the approximation of colimits in the heterogeneous case (Codescu and Mossakowski, 2008); (4) provide (first) solutions for automatically computing a base ontology through ontology intersection (Kutz and Normann, 2009) and blendoid evaluation using requirements or tools such as OOPS!.

In particular, we have shown that the blending of ontologies can be declaratively encoded in a DOL ontology representing the respective blending diagram—here, employing the homogeneous fragment of DOL just using OWL ontologies. Blendoid ontologies, as well as their components, i.e. input and base ontologies, can be stored, formally related, and checked for consistency within Conceptportal, a repository node within Ontohub dedicated to blending experiments carried out in the European FP7 Project COINVENT. Ontohub moreover gives access to thousands of ontologies from a large number of different scientific and common sense domains. They are searchable via rich metadata annotation, logics used, formality level, and other dimensions, to provide not only a rich pool of ontologies for blending experiments, but also for the evaluation of newly created concepts. Ontohub also supports a growing set of collaborative features, including online editing of ontologies, commenting, version control, and group and permission management.

To make concept invention via ontological blending feasible in practice from within Ontohub, a number of further plugins into the architecture are planned covering in particular the automatic creation of base ontologies together with their mappings, the implementation of filtering blendoids by structural optimality principles and preference orders on morphisms, as well as the addition of more ontologically motivated evaluation techniques as discussed above.

Acknowledgements

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.

References

Adámek, J.; Herrlich, H.; and Strecker, G. 1990. *Abstract and Concrete Categories*. Wiley, New York.

Codescu, M., and Mossakowski, T. 2008. Heterogeneous colimits. In Boulanger, F.; Gaston, C.; and Schobbens, P.-Y., eds., *MoVaH'08*.

Fauconnier, G., and Turner, M. 1998. Conceptual integration networks. *Cognitive Science* 22(2):133—187.

Fauconnier, G., and Turner, M. 2003. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.

Forbus, K.; Falkenhainer, B.; and Gentner, D. 1989. The structure-mapping engine. *Artificial Intelligence* 41:1–63.

Gärdenfors, P. 2000. *Conceptual Spaces - The Geometry of Thought*. Bradford Books. MIT Press.

Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7:155–170.

Goguen, J. A., and Burstall, R. M. 1992. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the Association for Computing Machinery* 39(1):95–146. Predecessor in: LNCS 164, 221–256, 1984.

Goguen, J. A., and Harrell, D. F. 2009. Style: A Computational and Conceptual Blending-Based Approach. In *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*. Springer.

Goguen, J., and Malcolm, G. 1996. *Algebraic Semantics of Imperative Programs*. MIT Press.

Goguen, J. 1999. An introduction to algebraic semiotics, with applications to user interface design. In Nehaniv, C. L., ed., *Computation for Metaphors, Analogy, and Agents*, volume 1562 of *Lecture Notes in Computer Science*. Springer. 242–291.

Grüninger, M., and Fox, M. S. 1995. The role of competency questions in enterprise engineering. In *Benchmarking—Theory and Practice*. Springer. 22–31.

Guarino, N., and Welty, C. 2002. Evaluating ontological decisions with OntoClean. *Commun. ACM* 45(2):61–65.

Hois, J.; Kutz, O.; Mossakowski, T.; and Bateman, J. 2010. Towards Ontological Blending. In *Proc. of the The 14th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA-2010)*.

Jaszczolt, K. M. 2003. On Translating 'What Is Said': *Tertium Comparationis* in Contrastive Semantics and Pragmatics. In *Meaning Through Language Contrast Vol. 2*. J. Benjamins. 441–462.

Kuhn, W. 2002. Modeling the Semantics of Geographic Categories through Conceptual Integration. In *Proc. of GIScience 2002*, 108–118. Springer.

Kutz, O., and Normann, I. 2009. Context Discovery via Theory Interpretation. In *Proc. of the IJCAI Workshop on Automated Reasoning about Context and Ontology Evolution, ARCOE-09, Pasadena, California*.

Kutz, O.; Lücke, D.; Mossakowski, T.; and Normann, I. 2008. The OWL in the CASL—Designing Ontologies Across Logics. In *Proc. of OWLED-08*, volume 432. CEUR.

Kutz, O.; Mossakowski, T.; Hois, J.; Bhatt, M.; and Bateman, J. 2012. Ontological Blending in DOL. In Besold, T.; Kuehnberger, K.-U.; Schorlemmer, M.; and Smaill, A., eds., *Computational Creativity, Concept Invention, and General Intelligence, Proc. of the 1st Int. Workshop C3GI@ECAI*, volume 01-2012. Montpellier, France: Publications of the Institute of Cognitive Science, Osnabrück.

Kutz, O.; Mossakowski, T.; and Lücke, D. 2010. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. *Logica Universalis* 4(2):255–333. Special Issue on 'Is Logic Universal?'.

Martinez, M.; Besold, T. R.; Abdel-Fattah, A.; Kühnberger, K.-U.; Gust, H.; Schmidt, M.; and Krumnack, U. 2011. Towards a Domain-Independent Computational Framework for Theory Blending. In *Proc. of the AAAI Fall 2011 Symposium on Advances in Cognitive Systems*.

Mossakowski, T.; Kutz, O.; Codescu, M.; and Lange, C. 2013. The Distributed Ontology, Modeling and Specification Language. In et al., C. D. V., ed., *Proceedings of the 7th International Workshop on Modular Ontologies (WoMO-13)*, volume 1081. CEUR-WS.

Neuhaus, F.; Vizedom, A.; Baclawski, K.; Bennett, M.; Dean, M.; Denny, M.; Grüninger, M.; Hashemi, A.; Longstreth, T.; Obrst, L.; et al. 2013. Towards ontology evaluation across the life cycle: The Ontology Summit 2013. *Applied Ontology* 8(3):179–194.

Normann, I. 2008. *Automated Theory Interpretation*. Ph.D. Dissertation, Department of Computer Science, Jacobs University, Bremen.

Núñez, R. E. 2005. Creating mathematical infinities: Metaphor, blending, and the beauty of transfinite cardinals. *Journal of Pragmatics* 37:1717–1741.

Pereira, F. C., and Cardoso, A. 2003. Optimality Principles for Conceptual Blending: A First Computational Approach. *AISB Journal* 1(4).

Pereira, F. C. 2007. *Creativity and Artificial Intelligence*, volume 4 of *Applications of Cognitive Linguistics*. Mouton de Bruyter.

Plotkin, G. D. 1970. A note on inductive generalization. *Machine Intelligence* 5:153–163.

Poveda-Villalón, M.; Suárez-Figueroa, M. C.; and Gómez-Pérez, A. 2012. Validating Ontologies with OOPS! In *Knowledge Engineering and Knowledge Management*. Springer. 267–281.

Ritze, D.; Meilicke, C.; Šváb Zamazal, O.; and Stuckenschmidt, H. 2009. A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences. In *OM-09*, volume 551 of *CEUR*.

Schwering, A.; Krumnack, U.; Kühnberger, K.-U.; and Gust, H. 2009. Syntactic Principles of Heuristic-Driven Theory Projection. *Cognitive Systems Research* 10(3):251–269.

Turner, M. 2007. The Way We Imagine. In Roth, I., ed., *Imaginative Minds - Proc. of the British Academy*. Oxford: OUP. 213–236.

Veale, T. 1997. Creativity as pastiche: A computational treatment of metaphoric blends, with special reference to cinematic "borrowing". In *Proc. of Mind II: Computational Models of Creative Cognition*.

Walshe, B. 2012. Identifying complex semantic matches. In *The Semantic Web: Research and Applications*. Springer. 849–853.