Abstract

The theory of conceptual blending has been applied very successfully in cognitive science to explain the process of concept generation. In previous work, we have shown a formalisation and implementation of conceptual blending that is borrowing techniques from logic, ontological engineering, and algebraic specification. However, so far our attempts (and similar approaches in the literature) have been reconstructive; that is they show how a particular concept (e.g., houseboat) can be 'discovered' by computationally blending two carefully selected input spaces (e.g, a house and a boat) and a suitable base. This paper describes an attempt to take the next step and automate the concept generation based on a database of input spaces. Particularly, we attempt to create monsters from a library of animals formalised as OWL ontologies.

Introduction

Conceptual blending in the spirit of Fauconnier and Turner operates by combining two input 'conceptual spaces', construed as rather minimal descriptions of some thematic domains, in a manner that creates new 'imaginative' configurations [**??**]. 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*. These examples illustrate that, typically, the blended spaces inherit some features from either space and combine them to something novel. The blending of the input spaces involves a base space, which contain shared structures between both input spaces. The structure in the base space is preserved in the blended space (the *blendoid*).

Goguen defines an approach that he terms *algebraic semiotics* in which certain structural aspects of semiotic systems are logically formalised in terms of algebraic theories, sign systems, and their mappings [**?**]. In **?**, algebraic semiotics has been applied to user interface design and conceptual 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. The formalisation of these blends can be formulated using the algebraic specification language OBJ3 [**?**].

In **???**, we have presented an approach to computational conceptual blending, which is in the tradition of Goguen's proposal. In these earlier papers, we suggested to represent the input spaces as ontologies (e.g., in the OWL Web Ontology Language[1]). We moreover presented how the Distributed Ontology Language (DOL) can be used to specify conceptual blends with the help of *blending diagrams*. These diagrams encode the relationships between the base space and the (two or more) input spaces. These *blending diagrams* can be executed by Hets, a proof management system. Hets is integrated into Ontohub,[2] an ontology repository which allows users to manage and collaboratively work on ontologies. DOL, Hets, and Ontohub provide a powerful set of tools, which make it easy to specify and computationally execute conceptual blends.

In this paper, we will discuss how we utilised DOL, Hets, and Ontohub in an attempt to build a prototype system that automates concept invention. The goal is to make the step from a reconstructive approach, where computational conceptual blending is illustrated by blending one concept (e.g., houseboat) with the help of some carefully selected input spaces (e.g, a house and a boat) to a system that autonomously selects two (or more) ontologies from a repository in Ontohub and attempts to blend them in a way that meets some given requirements. Within the extensive literature on conceptual blending, few attempts have been made at a (more or less) complete automation of the blending process, notable exceptions include **?**, **?**, **?**, and **??**.

In our experiment we use a repository of ontologies about animals as input spaces and try to blend them into monsters. In the next section, we are going to discuss the underlying model of our approach and how we simplified it in our prototype implementation. Finally, we summarise our initial results and discuss future work.

The Approach

We follow a variation of Goguen's approach to computational conceptual blending, which has been proposed within the COINVENT research project (see Figure **??** and `www.coinvent-project.eu`) and summarised by Marco Schorlemmer. In the next sections we discuss the various elements of this revised blending approach which we will refer to as the *Schorlemmer model*. Details of this revised model have not been published elsewhere so far, but the general background of the approach can be found in **?**.

(Weakened) Input Spaces

In this model there are two (or more) input spaces $I1$ and $I2$, which are in our case represented as OWL ontologies. These ontologies are randomly selected from a repository of animal ontologies in Ontohub. These ontologies are not 'fine-tuned' for a particular blend, but represent some features of types of animals, in particular their habitat, their diet, and some anatomical information (see Figure **??**). The goal is that in the future we will be able to easily both increase the depth of information that is provided in the animal ontologies as well as add new ontologies covering additional organisms.

The Schorlemmer model differs from the model proposed in **?** by introducing an extra step: the ontologies $I1$ and $I2$ are not blended directly, but are first weakened to two theories $I1^*$ and $I2^*$ (see Figure **??**). There are different strategies that can be used to generate the weaker theories from the input ontologies; the only constraint is that input ontologies logically entail their weakened counterparts. The purpose of this extra step is to remove some of the information from the input spaces that is undesired for the blend. There are several reasons why such a step might be necessary. Firstly, when blending a concept from a given ontology, typically large parts of the ontology are in fact off-topic.

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

[2] `www.ontohub.org`

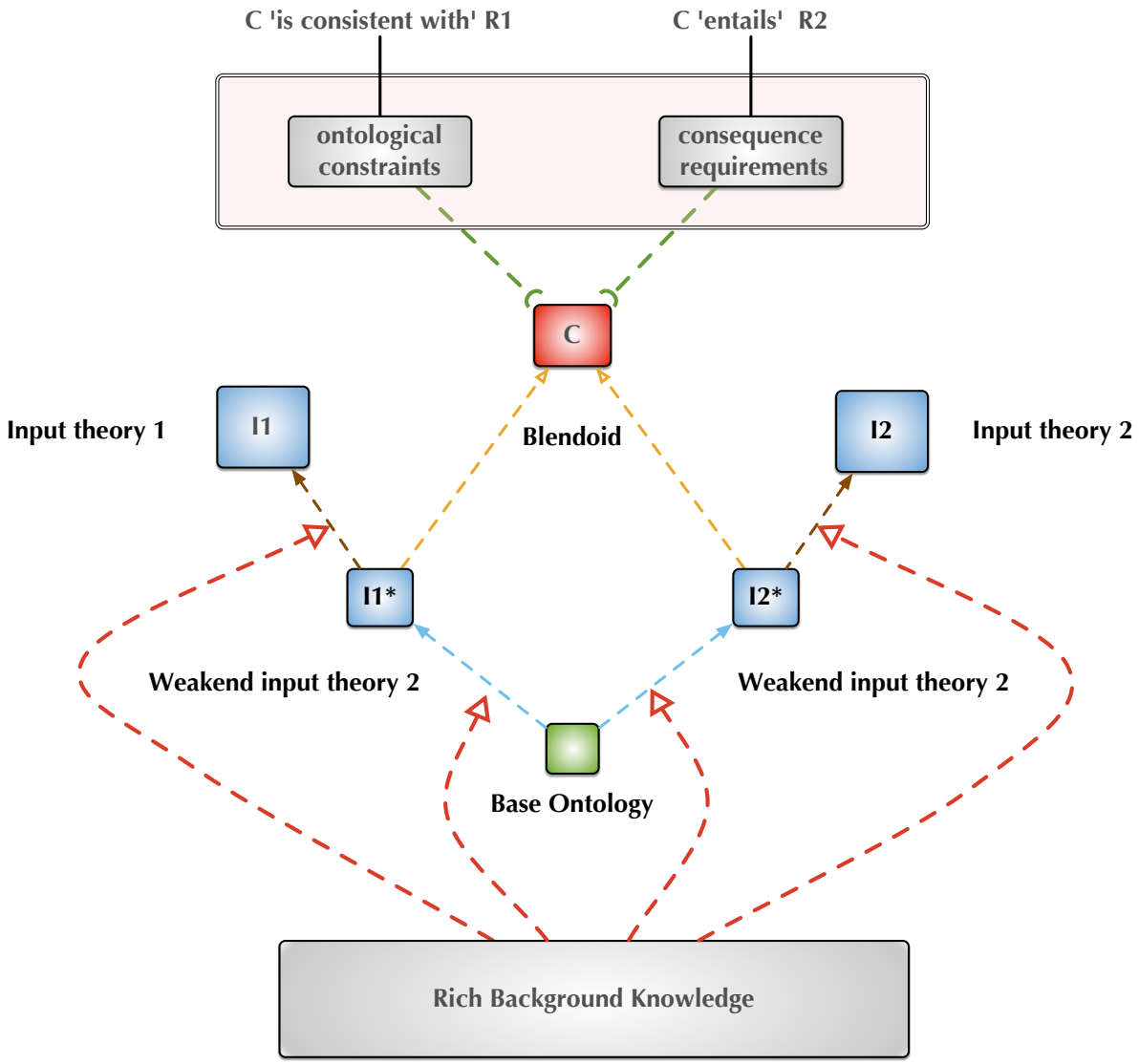

Figure 1: The core Schorlemmer model for computational blending enriched with evaluation and background layers

```
Class: Tiger
        SubClassOf: Mammal
        SubClassOf: Carnivore
        SubClassOf: has_habitat some Jungle
        SubClassOf: has_body_shape some QuadrupedShape
        SubClassOf: has_part some Fang
        SubClassOf: has_part exactly 4 Claw
        SubClassOf: has_part exactly 1 Tail
        SubClassOf: covered_by some Hair

Class: Viper
        SubClassOf: Reptile
        SubClassOf: Carnivore
        SubClassOf: has_habitat some
                        (Grasslands or Wetlands or Rocks)
        SubClassOf: has_body_shape some SnakeShape
        SubClassOf: has_part only (not Leg)
        SubClassOf: has_part some PoisonFang
        SubClassOf:  covered_by some Scales
```

Figure 2: Input space example

Logically speaking, when extracting a module for the concept in question, large parts of the ontology turn out to be logically irrelevant (module extraction is typically based on conservative extensions, see e.g. **?**). Secondly, when running the blend it may become obvious that the blendoid preserved too many properties from the input spaces. In this case, weakening the input spaces will lead to a better result.

We will discuss these issues in more detail below in the context of evaluation.

Base and Interpretations

The weakened input ontologies $I1^*$ and $I2^*$ are used to generate the base ontology. The base ontology is identifying some structure that is shared across $I1^*$ and $I2^*$. Or, to put it differently, the base ontology contains some theory, which can be found in both the input spaces, but it abstracts from the peculiarities of the input spaces and generalises the theory in some domain-independent way.

From the perspective of the workflow the base ontology is a more general theory that is generated from the (weakened) input ontologies. From a logical point of view, there exist two *interpretations* which embed the base ontology into $I1^*$ and $I2^*$. (In Figure **??** these are represented by the thinly dotted connectors between the base and $I1^*$ and $I2^*$.) These interpretations are a key element to make the automatic blending process work (see next section).

The Blend

The ontologies $I1^*$ and $I2^*$ together with the base ontology and the two interpretations that connect the base to $I1^*$ and $I2^*$ determine the blendoid. Informally, what happens is that the blendoid is a disjoint union of $I1^*$ and $I2^*$, where the shared structure from the base is identified.[3]

[3]Technically, the blendoid is the co-limit of the underlying diagram. For the formal details see **?** and **?**.

For example, assume one of our input ontologies is about tigers and the other about vipers. $I1^*$ and $I2^*$ are weakened versions of these input ontologies, where only some of the properties of tigers and vipers, respectively, are included. If the base ontology is empty, then the resulting blendoid consists of a theory that contains both tigers and snakes, but nothing is blended. If the base ontology identifies the tiger with the viper, the blendoid will be a monster that combines all the features of tigers and vipers that have been preserved in $I1^*$ and $I2^*$; e.g. you may get a tiger with a forked tongue and scales instead of hair. A different base may identify the viper with the tail of the tiger, in that case the resulting blend may consist of a tiger whose tail has eyes and poisonous fangs.

Background Knowledge and Requirements

To make the Schorlemmer model work in practice, the background knowledge and requirements have to play an essential role. Figure **??** represents a static view of how two input spaces are blended. However, since there are a vast number of potential blends, most of which are poor, computational concept blending is an iterative process. In each cycle, a new blendoid is created, and is evaluated against ontological constraints, i.e. a set $R1$ of axioms drawn from (common sense) background knowledge and with which a blendoid should not be in conflict, as well as a set $R2$ of consequence requirements, i.e. a collection of desired entailments a blendoid should yield. If the blendoid is rejected according to these criteria, the next cycle is started with different weakened input spaces and/or a different base. Ideally, the results of the evaluation is supposed to guide the changes in the next cycle. We will discuss the role of evaluation in more detail below.

Our Initial Implementation

For the purpose of our experiment we created a small library of ontologies of animals, describing some of their anatomy, their habitats, and their diets. Further, we developed several ontologies that contained background information. All ontologies were written in OWL Manchester Syntax.

In order to automate the blending process as modelled in Figure **??**, we had to provide the following functionality: given two selected ontologies from the repository, repeat the following steps until the blend is successful:

(i) weaken the input spaces and generate $I1^*$ and $I2^*$,

(ii) create the base ontology and the interpretations that link the base to $I1^*$ and $I2^*$,

(iii) execute the blend and generate the blendoid, and

(iv) evaluate the blendoid.

Weakening the input space. For the purpose of the initial implementation we wrote a simple script that removes some axioms in an OWL file. The script preserves the declaration of classes, individuals, and properties (thus, the signature of the ontology is not changed). The selection of the axioms that are deleted is randomised.

Generating the base ontology. For the purpose of an initial implementation we are currently working with a very simplified approach, where the bases consist basically of the shared signature of $I1^*$ and $I2^*$, which allows for trivial interpretations from the base. The only exception is the class Monster, which is mapped to the animals within the input spaces. E.g., Monster in the base ontology is mapped to the input ontologies $I1^*$ and $I2^*$, namely to Tiger and Viper, respectively.[4] Figure **??** shows the complete base and both interpretations for our running example.

```
ontology base =
        ObjectProperty: has_habitat
        ObjectProperty: has_body_shape
        ObjectProperty: has_part
        ObjectProperty: covered_by
        Class: Carnivore
        Class: Monster

interpretation base2Viper: base to viper  =
        Monster |-> Viper

interpretation base2Tiger: base to tiger  =
        Monster |-> Tiger

ontology monsterblend  =
    combine base2Viper, base2Tiger
```

Figure 3: Base and Interpretations

While this approach works, it limits the number of interesting blends severely. E.g., in our example of blending Tiger with Viper, the approach allows blendoids like a tiger with poisonous fangs and scales, but no tigers with a viper-like tail, because this would require the base to identify the tail of the tiger with the snake. This, however, can be easily obtained by allowing more complex base mappings.

Running the Blend. Hets provides the capability to run the blend. Technically, this is 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 **???**.

Figure **??** shows an example of a blendoid that is derived from the input spaces in Figure **??** with a weakening of both input spaces. In this case the monster inherits most of the tiger qualities, but it has poisonous fangs and is (partially) covered by scales.

Evaluation. Hets integrates a number of theorem prover and consistency checkers. We used Pellet and Darwin for the evaluation of the blendoids.

We evaluate blendoids both by considering its internal consistency and by looking for potential clashes with our background knowledge. For this purpose we use DOL to specify a new ontology that combines a blendoid with the background knowledge. This combined ontology is evaluated for logical consistency via Hets. The reason why the background knowledge is essential here is that the detection

[4]This approach presupposes that the same terminology is used consistently across the animal ontologies. However, since the ontologies were all developed in-house, this was not an issue.

```
Class: Monster
       SubClassOf: Carnivore
       SubClassOf: has_part some PoisonFang
       SubClassOf: covered_by some Scales
       SubClassOf: Mammal
       SubClassOf: has_habitat some Jungle
       SubClassOf: has_body_shape some QuadrupedShape
       SubClassOf: has_part some Fang
       SubClassOf: has_part exactly 4 Claw
       SubClassOf: has_part exactly 1 Tail
       SubClassOf: covered_by some Hair
```

Figure 4: Example Blendoid

of problems often requires more information than is contained within the blendoid. To return to our example about tigers and vipers, assume the input spaces in Figure **??** were weakened and that $I1^*$ contains the information that tigers have four claws, and $I2^*$ contains the information that vipers have no legs. In this case, the resulting blend will be a legless monster with claws. Without the additional background knowledge that claws are part of legs, it is impossible for a consistency checker to detect the inconsistency.

One requirement for a good blendoid is that it needs to combine information from both input spaces. In other words, if the information in the blendoid is contained in one of the input ontologies, then the blendoid is not a good conceptual blend. Since our approach involves a weakening of the input ontologies, it may happen that one of the ontologies is weakened so extremely that it does not contribute anything significant to the blendoid.

Often, a blending process is done with certain requirements in mind. E.g., in our example we may look for monsters that have four appendages. These requirements can be stated in DOL as proof obligations that have to be proven from the blendoid together with the background knowledge.

Discussion and Future Work

Our prototype implementation works in the sense that the system creates monsters by blending two animals. It works, in spite of the fact that two essential components of the blending model are handled quite bluntly:

(i) the base space is fixed to the shared signatures of the weakened input spaces and, thus, trivial; and

(ii) the weakening of the input spaces does not utilise the results of the evaluation, but happens randomly.

We here focused on making the workflow work, while accepting that some modules within the workflow are trivialised. Future work includes refining such a workflow, and of course using more sophisticated methods for its various parts.

Firstly, regarding the generation of the base ontology, we are planning to use heuristic-driven theory projection (HDTP) as outlined in **??**. Taking $I1^*$ and $I2^*$ as input theories, HDTP applies anti-unification techniques to generalise the input ontologies to a more general theory. So far, HDTP has been implemented for a sorted version of

FOL. To make it work in our context, we are going to need to support OWL directly (the preferred approach) or reduce OWL anti-unification to FOL anti-unification (e.g. via using a logic translation first and then a theory projection).

Secondly, regarding weakening of theories, we particularly plan to use the idea of 'amalgams' as proposed in **?**.

Thirdly, regarding revision of inconsistent blendoids, a number of tools and approaches are available to be employed in this context, amongst them: non-monotonic reasoning, in particular belief revision [**?**], and ontology debugging techniques, in particular for OWL [**?**]. Indeed, ontology debugging techniques are readily available via the OWL API.[5]

Concerning the latter, a promising idea is to interactively generate competency questions (cf. **??**) from justifications for inconsistencies [**?**]. Here, a user can steer the generation of new blends by rejecting certain ways to fix an inconsistent blendoid. A similar debugging workflow has recently been proposed by **?**, although only for the debugging of single inconsistent ontologies. In the case of blending, such approaches need to be adapted to a revision procedure covering networks of ontologies, where several ontologies (i.e. input and base ontologies) as well as the mappings between them are subject to revision.

While our system works, it often does not produce very good monsters. Interestingly, the limitations of the results are not (or only indirectly) caused by the fixing of the base space or the randomised weakening of the input spaces. For example, consider the result of the blend of a shark and a horse in Figure **??**. The monster in this ontology has fins and a tail, it is a herbivore and lives in some grasslands.

```
Class: Monster
        SubClassOf: Herbivore
        SubClassOf: has_habitat some Grassland
        SubClassOf: has_part some Fin
        SubClassOf: has_part exactly 1 Tail
```

Figure 5: Poor Blendoid

The ontology in Figure **??** illustrates several typical weaknesses of the blendoids that are generated by our system. First, the ontology does not provide any information about what kind of a monster it is and what shape it has. Both input spaces contained this information (e.g,. a horse is a mammal with the shape of a quadruped). Since our system removes axioms until it finds a blendoid that is consistent with the background knowledge, often information is removed that is not causing any inconsistency, leading to very weak ontologies. In this case, the existence of fins is the only information that remains from the shark ontology.

This issue would be partially addressed by choosing a weakening strategy that utilises the results of the evaluation process to selectively remove axioms. However, the more general point is that humans expect a description of a monster to answer certain information – like "What does it look like?" And many blendoids that are produced by the system do not contain the expected information. This can be fixed by encoding additional requirements, which are used during the evaluation process. E.g., one could add the following proof obligation:

```
Monster SubClassOf:
    has_body_shape some BodyShape
```

This obligation ensures that any successful blendoid will contain the information about the shape of the monster, but leaves open which.

Another reason why the blendoid in Figure **??** may be considered to be not a very impressive specimen is that it is not particularly scary. It is a herbivore living on grasslands; for all we know it may be a cow with a fin on the back. Again, the issue is that when humans perform conceptual blending they are guided by implicit assumptions about the nature of the result they are expecting. If we are expecting monsters to be scary, then this leads to additional requirements. In particular, a monster is only scary if it is has the disposition to attack people, and it is only able to do that if it has anatomical features that enable such attacks; e.g., claws or fangs or venomous stings.

Our framework allowed us to encode this information in the background ontology and add the additional requirement that the monster is supposed to be scary. As a result, the background ontology became several times as long and significantly more complex than the animal ontologies themselves.

So our main conclusion is that the blending framework performs relatively well, in spite of the shortcomings of some of its components. However, to get blendoids that a human would consider as interesting, one needs to encode a lot of the background knowledge and implicit requirements, which humans take for granted when they perform blends. Without such additional information the system cannot evaluate the candidate blendoids properly.

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.

[5] See `http://owlapi.sourceforge.net`