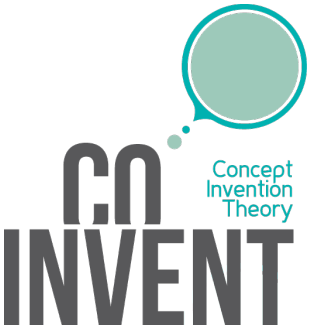




# D3.3

## Resolving Opposition and Contradiction for Concept Invention


| | |
|---|---|
| Authors | Oliver Kutz, Fabian Neuhaus |
| Reviewers | Tarek R. Besold, Marco Schorlemmer |



| | |
|---|---|
| Grant agreement no. | 611553 |
| Project acronym | COINVENT - Concept Invention Theory |
| Date | April, 30, 2016 |
| Distribution | PU |

**Disclaimer**

The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

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.

# Abstract

This deliverable D3.3 of the COINVENT project addresses issues related to the problem of resolving undesired consequences and/or inconsistencies in the concept invention workflow. It discusses variations of the basic workflow for concept invention, methods for generalisation resp. weakening of theories, and methods for the discovery resp. description of base spaces. A major focus is on the use of (formalisations of) image schemas as base spaces, in order to both steer the selection of interesting blends as well as to help debug inconsistencies.

Keyword list: **Conceptual Blending**, **Reasoning**, **Distributed Ontology Language DOL**, **Image Schemas**, **ASP**, **Generalisation**, **Amalgams**

## Executive Summary

Conceptual blending is a mental process that serves a variety of cognitive purposes, including human creativity. In this line of thinking, human creativity is modeled as a process that takes different mental spaces as input and combines them into a new mental space, called a *blend*. According to this form of *combinational creativity*, a blend is constructed by taking the commonalities among the input mental spaces into account, to form a so-called *generic space*, and by projecting the non-common structure of the input spaces in a selective way to the novel blended space.

The theory of conceptual blending has been applied very successfully in cognitive science to explain the process of concept invention. In previous COINVENT deliverables D3.1 and D3.2, we have elaborated on a formalisation and implementation of conceptual blending borrowing techniques from logic, ontological engineering, and algebraic specification, and in particular employing the Distributed Ontology Language DOL.

This deliverable D3.3 of the COINVENT project addresses issues related to the problem of resolving undesired consequences and/or inconsistencies in the concept invention workflow. It discusses variations of the basic workflow for concept invention, methods for generalisation resp. weakening of theories, and methods for the discovery resp. description of base spaces. A major focus is on the use of (formalisations of) image schemas as base spaces, in order to both steer the selection of interesting blends as well as to help debug inconsistencies.

The deliverable is based on four papers accepted for publication. In "Fabricating Monsters is Hard: Towards the Automation of Conceptual Blending", by Fabian Neuhaus, Oliver Kutz, Mihai Codescu, and Till Mossakowski [4], the general concept invention workflow and its automation are addressed. So far, most approaches for concept invention in the literature have been reconstructive. That is to say, 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, it is shown how to create 'monsters' from a library of animals formalised as OWL ontologies.

The next two papers deal with the relationship between concept invention and image schemas. Image schemas are recognised as a fundamental ingredient in human cognition and creative thought. They have been studied extensively in areas such as cognitive linguistics. In cognitive science, image schemas are identified as fundamental patterns of cognition. They are seen as schematic prelinguistic conceptualisations of events and serve as conceptual building blocks for concepts. However, the very notion of an image schema is still ill-defined, with varying terminology and definitions throughout the literature. In the work presented in this deliverable, it is proposed and outlined in what way image schemas can play an important role in computational concept invention, particularly within the computational realisation of conceptual blending.

With the goal of exploring the potential role of image schemas in computational creative systems, in "Choosing the Right Path: Image Schema Theory as a Foundation for Concept Invention",

by Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus [2], the viability of the idea to formalise image schemas as graphs of interlinked theories is studied. In particular, a selection of image schemas related to the notion of 'path' is discussed, and it is shown how they can be mapped to a formalised family of micro theories reflecting the different aspects of path following. Finally, the potential of this approach in the area of concept invention is illustrated, namely by providing several examples showing in detail in what way formalised image schema families support the computational modelling of conceptual blending.

In "Image schemas in computational conceptual blending", by Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus [3], a library of formalised image schemas is proposed, and it is illustrated how they can guide the search for a base space in the concept invention work flow. Their schematic nature is captured by the idea of organising image schemas into families. Formally, they are represented as heterogeneous, interlinked theories.

The last paper in this deliverable, "Upward Refinement Operators for Conceptual Blending in the Description Logic $\mathcal{EL}^{++}$" by Roberto Confalonieri, Manfred Eppe, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, and Enric Plaza [1], focuses on a particular formal approach to resolve inconsistencies arising in conceptual blending. Since input spaces for interesting blends are often initially incompatible, a generalisation step is needed before they can be blended. In this paper, this idea is applied to blend input spaces specified in the description logic $\mathcal{EL}^{++}$ and it is proposed to use an upward refinement operator for generalising $\mathcal{EL}^{++}$ concepts. It is shown how the generalisation operator is translated to Answer Set Programming (ASP) in order to implement a search process that can find possible generalisations of input concepts. The generalisations obtained by the ASP process are used in a conceptual blending algorithm that generates and evaluates possible combinations of blends. Finally, it is illustrated how the approach can be employed in the domain of computer icons for the generation and evaluation of new icons.

# Bibliography


[1] Roberto Confalonieri, Manfred Eppe, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, and Enric Plaza. Upward refinement operators for conceptual blending in the description logic $\mathcal{EL}^{++}$. *Annals of Mathematics and Artificial Intelligence*, 2016. (Forthcoming).

[2] Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus. Choosing the Right Path: Image Schema Theory as a Foundation for Concept Invention. *Journal of Artificial General Intelligence*, 6:21–54, 2015.

[3] Maria M Hedblom, Oliver Kutz, and Fabian Neuhaus. Image schemas in computational conceptual blending. *Cognitive Systems Research*, 39:42–57, 2016.

[4] 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.

# Table of Contents

# Fabricating Monsters is Hard: Towards the Automation of Conceptual Blending

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

## 1 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 [3, 24]. 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 [4]. In [7], 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 [6].

In [9, 13, 15], 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

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

[2] `www.ontohub.org`

on conceptual blending, few attempts have been made at a (more or less) complete automation of the blending process, notable exceptions include [7], [19], [16], and [25, 26].

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.

## 2 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 1 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 [21].

### 2.1 (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 2). 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 organism.

The Schorlemmer model differs from the model proposed in [7] 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 1). 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. 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. [12]). 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.

### 2.2 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

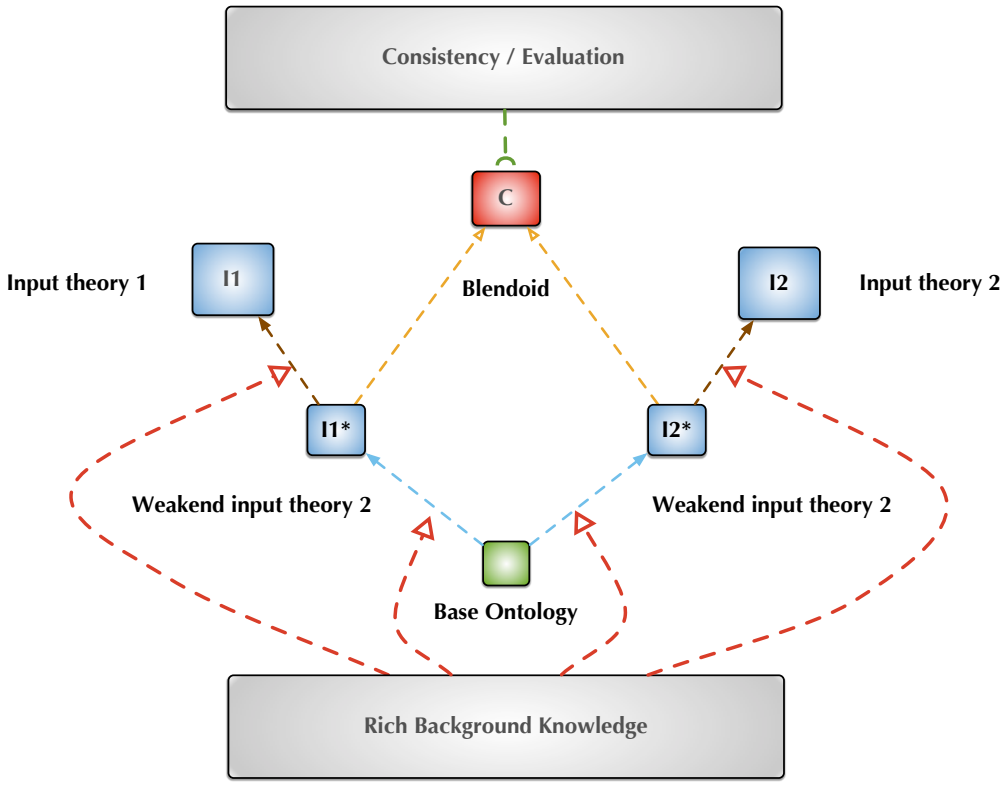



Fig. 1: The core Schorlemmer model for computational blending enriched with evaluation and background layers

```
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
```

Fig. 2: Input space example

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 1 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).

### 2.3 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]

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.

### 2.4 Background Knowledge and Requirements

To make the Schorlemmer model work in practice, the background knowledge and requirements have to play an essential role. Figure 1 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.

### 2.5 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.

[3] Technically, the blendoid is the co-limit of the underlying diagram. For the formal details see [1] and [13].

In order to automate the blending process as modelled in Figure 1, 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 3 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
```

Fig. 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.

[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.

*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 [5, 13, 14].

Figure 4 shows an example of a blendoid that is derived from the input spaces in Figure 2 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.

```
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
```

Fig. 4: Example Blendoid

*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 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 2 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 leg-less 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.

## 3 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 [17, 22]. 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 [18].

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 [2], and ontology debugging techniques, in particular for OWL [11]. 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. [8, 20]) from justifications for inconsistencies [10]. 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 [23], 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 5. The monster in this ontology has fins and a tail, it is a herbivore and lives in some grasslands.

The ontology in Figure 5 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

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

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

Fig. 5: Poor Blendoid

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 5 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.

# Bibliography


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

[2] C Alchourrón, P Gärdenfors, and D Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

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

[4] J. A. Goguen. An Introduction to Algebraic Semiotics, with Applications to User Interface Design. In *Computation for Metaphors, Analogy and Agents*, number 1562 in LNCS, pages 242–291. Springer, 1999.

[5] J. A. Goguen. Semiotic Morphisms, Representations and Blending for Interface Design. In *Proc. of the AMAST Workshop on Algebraic Methods in Language Processing*, pages 1–15. AMAST Press, 2003.

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

[7] Joseph Goguen and D. Fox Harrell. Style: A Computational and Conceptual Blending-Based Approach. In Shlomo Argamon and Shlomo Dubnov, editors, *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*, pages 147–170. Springer, Berlin, 2010.

[8] Michael Grüninger and Mark S Fox. The role of competency questions in enterprise engineering. In *Benchmarking—Theory and Practice*, pages 22–31. Springer, 1995.

[9] J. Hois, O. Kutz, T. Mossakowski, and J. Bateman. Towards Ontological Blending. In *Proc. of the The 14th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA-2010)*, Varna, Bulgaria, September 8th–10th, 2010. Springer.

[10] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all Justifications of OWL DL Entailments. In *Proc. of ISWC/ASWC2007*, 4825, pages 267–280. Springer, 2007.

[11] A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics*, 3(4):268–293, 2005.

[12] B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic Modularity and Module Extraction in Description Logics. In *18th European Conf. on Artificial Intelligence (ECAI-08)*, 2008.

[13] O. Kutz, T. Mossakowski, J. Hois, M. Bhatt, and J. Bateman. Ontological Blending in DOL. In Tarek Besold, Kai-Uwe Kuehnberger, Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity, Concept Invention, and General Intelligence, Proc. of the 1st Int. Workshop C3GI@ECAI*, volume 01-2012, Montpellier, France, August 27 2012. Publications of the Institute of Cognitive Science, Osnabrück.

[14] O. Kutz, T. Mossakowski, and D. 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?'.

[15] O. Kutz, F. Neuhaus, T. Mossakowski, and M. Codescu. Blending in the Hub—Towards a collaborative concept invention platform. In *Proc. of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia, June 10–13 2014.

[16] Boyang Li, Alexander Zook, Nicholas Davis, and Mark O. Riedl. Goal-Driven Conceptual Blending: A Computational Approach for Creativity. In *Proc. of the 2012 International Conference on Computational Creativity*, Dublin, Ireland, 2012.

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

[18] Santiago Ontañón and Enric Plaza. Amalgams: A formal approach for combining multiple case solutions. In *Case-Based Reasoning. Research and Development*, pages 257–271. Springer, 2010.

[19] F. C. Pereira. *Creativity and Artificial Intelligence: A Conceptual Blending Approach*, volume 4 of *Applications of Cognitive Linguistics (ACL)*. Mouton de Gruyter, Berlin, December 2007.

[20] Yuan Ren, Artemis Parvizi, Chris Mellish, Jeff Z Pan, Kees van Deemter, and Robert Stevens. Towards competency question-driven ontology authoring. In *The Semantic Web: Trends and Challenges*, pages 752–767. Springer, 2014.

[21] Marco Schorlemmer, Alan Smaill, Kai-Uwe Kühnberger, Oliver Kutz, Simon Colton, Emilios Cambouropoulos, and Alison Pease. COINVENT: Towards a Computational Concept Invention Theory. In *Proc. of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia, June 10–13 2014.

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

[23] Kostyantyn Shchekotykhin, Gerhard Friedrich, Patrick Rodler, and Philipp Fleiss. Interactive Ontology Debugging using Direct Diagnosis. In *Proc. of the Third International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM-14), May 26, 2014, ESWC, Anissaras/Hersonissou, Greece*, 2014.

[24] Mark Turner. *The Origin of Ideas: Blending, Creativity, and the Human Spark*. Oxford University Press, 2014.

[25] Tony Veale. From Conceptual Mash-ups to "Bad-Ass" Blends: A Robust Computational Model of Conceptual Blending. In *Proc. of the 2012 International Conference on Computational Creativity*, Dublin, Ireland, 2012.

[26] Tony Veale, Diarmuid O Donoghue, and Mark T Keane. Computation and blending. *Cognitive Linguistics*, 11(3/4):253–282, 2000.

# Choosing the Right Path: Image Schema Theory as a Foundation for Concept Invention

Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus

## 1 Introduction

The cognitive processes underlying concept invention are still largely unexplored ground, although promising theories have been developed within the last few decades. Two of the most influential directions are the theory of grounded cognition [4] and the embodied mind theory [17, 42]. Both propose that human cognition is grounded in our bodily experience with the environment. In the case of grounded cognition, mental representations are thought to be derived from the embodied experience that is used to structure concepts, including even the most abstract ones. Embodied cognition takes this one step further by arguing against mental representations, and states that concepts are (or may be identified with) the neural activation of embodied experiences.

Following the reasoning of how embodied experiences shape our cognition, a more specific theory of concept formation and language understanding was introduced under the name of image schemas. It is a theory that focuses on basic spatial cognition. The theory was originally developed in the late 80s by [41] and [29], and was quickly taken up by other researchers in the area (e.g. [45]).

According to [29] *"an image schema is a recurring dynamic pattern of our perceptual interaction and motor programs that gives coherence and structure to our experience."*

The 'image schema' is thought to be the abstracted spatial pattern from repeated sensorimotor experience. These mental structures offer a foundation and a way to ground other cognitive phenomena, such as language capacity, understanding, and reasoning. They offer a connection between the bodily experienced relationships of physical objects in time and space with the internal conceptual world of an agent. In language, they can be seen as the conceptual building blocks for metaphoric and abstract thought. Some of the most commonly mentioned examples of image schemas are: CONTAINMENT, SUPPORT and MOVEMENT_ALONG_PATH[1].

Another important approach towards an understanding of concept invention is the theory of *conceptual blending*, introduced by [14], which developed further the idea of *bisociation* introduced by the psychologist [32]. Conceptual blending proposes that novel concepts arise from a selective combination of previously known information (see Section 6 for a more thorough introduction). It is our belief that a formal approach to image schemas in connection with the framework of conceptual blending will greatly support the development of computational systems for concept invention. A formal representation of these conceptual building blocks might in particular provide a novel approach to the symbol grounding problem.

However, much of the image schema research is inconsistent regarding terminology and definitions of image schemas, making formal approaches challenging. The borders between different

[1] This image schema is also referred as SOURCE_PATH_GOAL schema. For reasons that will become more obvious in Section 5, we use MOVEMENT_ALONG_PATH as the most generic term for this image schema.

image schemas are often vague and overlapping. It is also unclear where to draw the line regarding what spatial relationships should be called image-schematic (cf. image schema concepts such as ABOVE, with directional concepts such as left and right), and to what degree. While previous research in cognitive linguistics (e.g. [25]) and developmental psychology (e.g. [46]) provides some first steps towards a more unified terminology, the identification of these abstract patterns has been established to be difficult.

From a formal perspective, previous research on image schemas (e.g. [36, 69, 78]) has provided a valuable portfolio of approaches that can be build on further. However, more attention still needs to be paid to building a more unified terminology integrating the formal and cognitive-linguistic approaches found in the literature, whilst allowing a more systematic formalisation strategy [24].

Our principle claim in this paper is that the 'Gestalt' idea of image schemas should be analysed as family-resemblance. Within the theory of mind, Gestalt psychology aims to explain the human ability to ascribe meaning to seemingly chaotic perceptions. The most important aspect here is that the mind has the capacity to 'form a whole' which can differ from the collection of parts in important aspects, including emergent properties that may result from a new 'self-organisation' of those parts. For image schemas, the notion of Gestalt is related to how the semantic content of a complex expression may extend, exceed, or simply change the semantic content of its parts [33]. The formal analysis of this 'family-resemblance'[2] should provide a set (i.e. a *family*) of interlinked theories (in the weakest case, a set of theories ordered by logical entailment, giving rise to a lattice), each of which covering a particular conceptual-cognitive scenario within the schema.

To illustrate our approach, we will use the image-schematic structure found in language to suggest how image schemas can be represented as lattices of theories. This illustrates how simple image schemas can be made more elaborate within their respective 'family'. To further outline our formal approach, we will use the image schema of MOVEMENT_ALONG_PATH, analyse its connections to natural language, and sketch-out a lattice axiomatised in first-order logic which makes explicit the different branching points of micro theories involved in the family.

The remainder of the paper is structured as follows: In Section 2, a more detailed account of image schemas is presented by discussing their internal structure and their role in language. In Section 3, the PATH-following is discussed in more detail. In Section 4, formal approaches to image schemas are discussed and we introduce our idea of how to gather image schemas in families of theories by using PATH-following as a proof of concept. This is done by representing PATH-following as a DOL-graph and as a FOL-axiomatisation. In Section 5, we investigate the role image schemas can play in concept invention within the framework of conceptual blending theory. Finally, in Sections 6 and 7, a discussion and a short conclusion are provided.

## 2 Image schemas

### 2.1 Image schemas and embodiment

The theory of image schemas stems from the theory of embodied/grounded cognition. It is a theory that emphasises the role of bodily experiences as a source for cognitive capacities. It has become

[2] Similar ideas can be found in e.g. [8], in relation to prototype theory [64] and of course in the work of [81].

increasingly supported by findings in cognitive linguistics and neuroscience (e.g. [15, 17, 44, 70, 80]).

The theory offers an interesting view of cognition for approaches to artificial intelligence as it provides a more direct route to computational cognition than traditional, more hard-coded approaches. I.e., artificial agents are encouraged to learn, and artificial cognitive structures are populated by their 'experiences', similar to the learning process observed in human children (see [9] for an introduction to embodied artificial intelligence and [76] for an overview of such cognitive architectures).

Building on grounded cognition, image schemas are thought to be the mental representations extracted from bodily experiences, and more specifically, experiences that can be described using basic spatial relationships. Image schemas are therefore mental abstractions of learnt spatial relationships (e.g. CONTAINMENT or SUPPORT).

In the early stages of cognitive development, these image schemas are formed and 'fine-tuned' as the experiences with a particular spatial relationship are increased and extended to different situations [47, 63]. Due to this fine-tuning, it appears prominent that image schemas consist of different 'parts' [46]. These 'parts' can either be removed or added while still capturing the same basic image schema, generating what can be described as an image schema 'family'. [48] refer to these 'parts' as *spatial primitives*; the fundamental spatial building blocks. As this view is essential to our approach for formalising image schemas we will regularly return to the notion of spatial primitives in later sections.

The purpose of image schemas lies in their role as abstracted spatial relationships. It is believed that they contain vital information for the conceptual understanding of concepts and their surroundings. For example, to properly understand what a 'cup' is, an infant needs to understand that the cup in essence is a CONTAINER. Respectively, the notion of a 'table' needs to be connected to a SUPPORT image schema since otherwise the child will not understand that objects remain on tables after being placed on them. This way, image schemas map affordances of objects (in the sense of Affordance Theory [20]) and can be used to explain increasingly more complicated concepts. This is done through information transfer and can be observed in natural language, for example in conceptual metaphors.

### 2.2 Information transfer and conceptual metaphor

[27] suggested analogy to be at the core of cognition. This makes image schemas a valuable asset as the cognitive benefit of image schemas lies in their generalised nature. For example, if the image schema of SUPPORT has been learnt through perceptual exposure of 'plates on tables', an infant can infer that table-like objects such as 'desks' also have the SUPPORT image schema and can SUPPORT objects such as 'books' as well. As the environment becomes increasingly complex for the infant, this information transfer becomes a fundamental part of cognition.

Information transfer in language is often done through conceptual metaphors. Information is moved from one known source domain to an unknown target domain. Conceptual metaphors can be further specialised to image-schematic metaphors. These are the metaphors that do not transfer general conceptual knowledge from one domain to another but the skeletal structure of the image schema [34], and are the metaphors that this paper primarily covers. Here the source domain in the

analogical transfer is stripped down to the image schema skeleton which is mapped to the target domain and there fleshed out with local domain information.

As language develops and the individual is exposed to increasingly more abstract concepts than those found in early infancy, image schemas can be used to ground the novel concepts in already comprehended concepts. Embodied experiences and image schemas are often used in natural language to explain abstract concepts. For example, in a social hierarchy people can be either 'above' or 'below' us, expressions learned from embodied experiences of the image schema ABOVE, which itself derives from experiencing the human body's vertical axis. In natural language, metaphors such as 'falling from grace' or 'the rise to power' use the same image schema to represent status and success.

One of the most well-known examples of an attempt at grounding abstract concepts in image schemas is the work of [43]. In *Where Mathematics Comes From*, they defend the view that image schemas lay the foundation for abstract concepts in mathematics. They explain how the notions of addition and subtraction can be traced from back and forward MOVEMENT_ALONG_PATH, and extend the reasoning to more abstract constructions such as complex numbers. While being influential work, it has also received heavy criticism, in particular targeted at vague terminology and methodology (e.g. [22, 65]) and mathematical errors (e.g. [77]).

The image schema CONTAINMENT is commonly described as the sum of the interrelationships of an inside, an outside and a boundary [41]. An abstract example of the image schema CONTAINMENT is the conceptual metaphor *"to be in love"*. Obviously, there is no spatial region for the emotional state of love in the same sense as there is for a physical container such as a cup. Yet, we use the spatial language to talk about the phenomenon of love: e.g. we 'fall in love' or 'fall out of love'.

There is a clear connection between CONTAINMENT and prepositions such as 'in', 'into' and 'out of'. [5] investigated the CONTAINMENT relationship by searching text corpora for words similar to containment, e.g. 'surrounding' and 'enclosing'. The authors' method distinguished eight different kinds of CONTAINMENT.

Prepositions in combination with verbs often do appear to be the key words that help identify image schemas in language [28]. Below we will discuss natural language and conceptual metaphors of the PATH-following image schema family.

### 2.3 Image schemas and their structure

In this section we, look more closely at how image schemas are thought to be structured. The first pertinent distinction is that image schemas can be both static and dynamic. For example, CONTAINMENT can either describe the situation in which the cup already contains coffee, or alternatively the situation in which the coffee is poured from a source: a kettle, to a goal: a cup, defined as an IN and OUT schema. The static image schemas can in turn be differentiated under three different categories: *orientational* (e.g. ABOVE), *topological* (e.g. CONTACT), and *force-dynamic* (e.g. SUPPORT) [43].

A more dynamic way to understand the IN and OUT schema is to view them as combinations of the two image schemas CONTAINMENT and MOVEMENT_ALONG_PATH, building on the idea that image schemas can be combined with one another to generate more specific and

complex image schemas [36, 48, 57, 78]. Another example is how MOVEMENT_ALONG_PATH easily can be connected with the image schema LINK resulting in the higher level image schema LINKED_PATH: The image schema concept that encompasses linked behaviour on two, or more, joint paths. This "Gestalt" grouping of image schemas means that there must be a distinction between the most perceptually primitive image schemas and the more complex image schemas.

In language, this corresponds to the observation that combinations of image schemas are suitable to describe more complex concepts. To illustrate this, [36] suggested that 'transportation' can be understood as a combination of the image schemas of SUPPORT and MOVEMENT_ALONG_PATH, and [46] suggested that 'marriage' can be viewed as a LINKED_PATH.

One proposal to hierarchically structure the range from simple to more complex and dynamic image schemas is the approach presented in [48], which builds on empirical data from studies on cognitive development. In their work, the umbrella term 'image schemas' is divided into three different levels: *spatial primitives*[3] (the conceptual building blocks build from spatial information), *image schemas*[4] (simple spatial stories), and *conceptual integrations* (image schemas combined with a non-spatial element such as force or emotion).[5]

The complex and intermixed structure of image schemas just outlined implies that it is not possible to consider an image schema as a stand-alone individual theory. Instead, we endorse the notion of grouping image schemas based on their general characteristics. We believe that the change from one spatial schema to another can be accomplished by providing or detailing more spatial information, such as that found in the spatial primitives. In Section 5, we will demonstrate this idea by using PATH-following as a proof of concept. In the next section PATH-following will be introduced in more detail.

## 3 The image schema family PATH-following

### 3.1 Introducing PATH-following

In this section, we will explain how image schemas, like MOVEMENT_ALONG_PATH, are members of image schema families. For this purpose we introduce the PATH-following image schema family and illustrate how the family is organised hierarchically from general to more specific versions of PATH-following by the addition of spatial information and primitives.

MOVEMENT_ALONG_PATH is one of the first image schemas to be acquired in early infancy as children are immediately exposed to movement from a range of objects. This, in combination with the neurological priority to process moving objects over static objects, suggests that the image schema is either innate or learnt at a very early stage in cognitive development [63]. However, in order to understand how the PATH-following family is fine-tuned and in 'more completion'

[3] The notion of spatial primitives is not novel. Research on such semantic building blocks can be found in the linguistic literature. E.g. the work on spatial semantics by [75] and the more general work on semantic primes that covers more than the spatial and temporal aspects found in spatial primitives [79].

[4] When referring to this concept we will use the term *spatial schemas* to avoid ambiguity between the wider notion of 'image schema' and its narrower sense introduced by [48].

[5] For the purposes of this paper, only spatial primitives and spatial schemas will be further discussed. In principle, our approach is general enough to allow for heterogeneity, also on the logical level. Therefore one may also include conceptual integrations involving non-spatial elements in our image schema families, cf. the discussion in Sections 5 and 4.3.

internally structured, experiments with children have provided some insights on distinguishing how the different spatial schemas may develop.

Firstly, already at an early age children pay more attention to moving objects than resting objects. Trivial as it may seem, it requires children to detect the spatial primitive OBJECT (or THING) and the spatial schema MOVEMENT_OF_OBJECT.[6] Secondly, children tend to remember the PATH of the movement of the object. The PATH is a spatial primitive, which is different from the movement and the moving object.[7].

In addition to these two basic spatial primitives and as the child becomes more and more familiar with the image schema, more spatial information is added to fine-tune PATH-following. This means that in more advanced stages, the image schema encompasses beyond MOVEMENT-_OF_OBJECT and the spatial PATH itself, also the spatial primitive END_PATH, and later also a START_PATH [48]. Already at five months infants can distinguish PATH-following that has an END_PATH (the image schema PATH_GOAL) from the initial PATH, while the START_PATH is less interesting until the end of the first year of life. This is further supported by linguistic analyses in which an END_PATH is initially more interesting than a START_PATH [28].

Table 1 summarises the spatial primitives that may be involved in image schemas of the PATH-following family.[8]

Table 1: Spatial primitives of the PATH-following family

| **Spatial primitive** | **Description** |
|---|---|
| OBJECT | an object |
| PATH | the path the object moves along |
| START_PATH | the initial location |
| END_PATH | the final location |

A more specified example of the PATH-following family is presented by [43]. In accordance with other linguistic literature on image schemas they are focussed on the SOURCE_PATH_GOAL schema, see Figure 1. Here, the object, called trajector, moves from a source to a goal. END_PATH and START_PATH are not identical to the SOURCE and GOAL found in the SOURCE_PATH_GOAL schema. In SOURCE_PATH_GOAL, a direction and a purpose are implied in the image schema, which changes the conceptual nature of the movement. [43] make the distinction of 'elements', or roles, that to some extent correspond to the spatial primitives discussed above, but additional distinctions are added. The elements[9] are listed in Table 2. Most importantly, they make the clear distinction between end location and goal, as they distinguish between 'path', the actual trajectory of a movement, and 'route', the expected movement.

[6] OBJECT is understood here in a very wide sense that includes not only solid material objects but entities like waves on a pond or shadows. [48] also discuss MOVE as a spatial primitive of its own. We consider MOVEMENT_OF-_OBJECT to be a spatial schema, since movement necessarily involves a temporal dimension and, further, it always involves at least one spatial primitive, since any movement, necessarily, involves at least one OBJECT that moves.

[7] This spatial primitive is not to be confused with the image schema family PATH-following.

[8] Table 1 is based on [48], but includes some changes. In particular, as mentioned above, we do not consider MOVE to be a spatial primitive.

[9] We altered the terms to better match the terminology in [48], but no change in content was made.

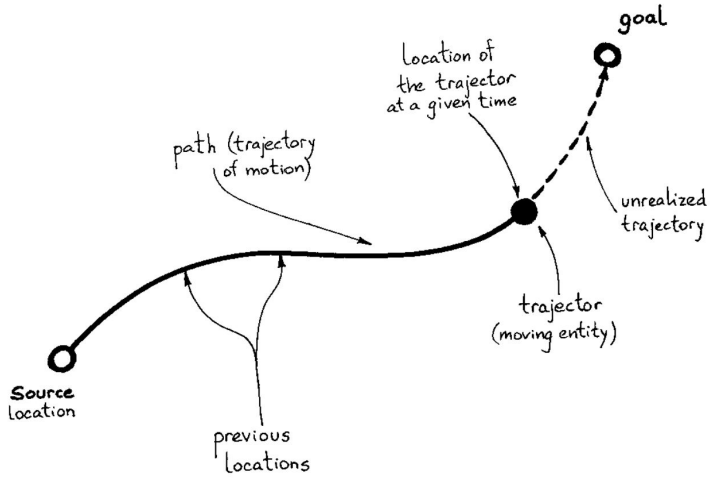



Fig. 1: The SOURCE_PATH_GOAL schema as illustrated by [43].

Table 2: Elements of PATH according to [43]

| Element | Description |
|---|---|
| trajector | The object |
| source | The initial location |
| goal | The intended end location |
| route | A pre-realised route from source to goal |
| path | The trajectory of motion |
| position | The position of the trajector at a given time |
| direction | The direction of the trajector at a given time |
| end location | End location, may not correspond to the goal location |

The distinction, made by [43], between the *expected* movement and the *actual* movement is primarily interesting for a description of how new image schemas relate to actual events and how new image schemas are learned. Consider, for example, a situation where a child observes the movement of a billiard ball and is surprised that the ball stops because it is blocked by another billiard ball. In this case, a given instance of the MOVEMENT_ALONG_PATH spatial schema formed the expectations of the child, which were disappointed by the actual physical movement, because the expected END_PATH (the goal) does not correspond to the actual END_PATH (end location). Given a repeated exposure to similar events, the child may develop the new spatial schema BLOCKAGE. After learning BLOCKAGE, the child will no longer be surprised by blocked movement since the expected END_PATH (the goal) will correspond to the actual END_PATH (end location). While the terminological distinction between *expected trajectory* and *actual trajectory* is useful, these do not necessarily need to constitute two different spatial primitives. Indeed, spatial primitives are parts of image schemas and, thus, always parts of conceptualisations, and not parts of actual events.

While the notions of path-following of [48] and [43] coincide widely, there are differences in terminology and definitions. In this paper we follow primarily the former.

### 3.2 Concepts that involve PATH-following

As briefly demonstrated with CONTAINMENT and ABOVE, image schemas can be used as a source for grounding abstract concepts in already comprehended concrete concepts. In this section we consider examples for concepts, which involve members of the PATH-following family.

The most straightforward examples of concepts that involve PATH-following are concepts that are about the spatial relationship of movement between different points. Prepositions such as *from, to, across* and *through* all indicate a kind of PATH-following[10]. This also includes key verbs that describe movement, e.g. *coming* and *going*. Another example, here for the spatial schema SOURCE-_PATH_GOAL, is *Going from Berlin to Prague*. Note that in many cases we do not provide information about START_PATH and END_PATH of a movement; e.g. *leaving Berlin* and *travelling to Berlin* are examples for the spatial schemas SOURCE_PATH and PATH_GOAL, respectively. *Meandering* is an example for a concept that realises MOVEMENT_ALONG_PATH, which involves a PATH but no START_PATH or END_PATH. In contrast, no discernable PATH is involved in *roaming the city*, which is an example for MOVEMENT_OF_OBJECT, the most general member of the PATH-following family.

Looking at abstract concepts and conceptual metaphors, PATH-following is found in many expressions. The concept of *"going for a joy ride"* realises the spatial schema SOURCE_PATH, since it has a START_PATH and a PATH but no END_PATH. Similarly, the expression *"running for president"* describes the process of trying to get elected as president metaphorically as a PATH_GOAL. In this metaphor the PATH consists of the various stages of the process (e.g. announcing a candidacy and being nominated by a party) with the inauguration as END_PATH.

Another metaphor *"life is a journey"*, studied by [2], makes an analogical mapping between the passing of time in life, to the passing of spatial regions on a journey. As in the example mentioned above, where the concept of "being in love" acquired information from the CONTAINMENT schema, this metaphor gains information from the spatial primitives connected to the image schema SOURCE_PATH_GOAL. Here, the most important spatial primitives are START_PATH and END_PATH – in this metaphor they are mapped to the moments of birth and death, as well as the PATH itself, illustrating how "life goes on" in a successive motion without branching.

A different perspective on life and death is expressed in the metaphorical expression *"the circle of life"*. Implied is that life leads to death, but also that death gives rise to life, completing a cyclic movement – the image schema MOVEMENT_IN_LOOPS. This image schema can be considered as a version of PATH-following, in which START_PATH and END_PATH coincide at the same 'location'.

These examples illustrate a general pattern, namely that many conceptual metaphors involving PATHs are about processes, and different events during such processes are treated metaphorically as locations on a path. This leads to a conceptualisation of the abstract concept of time, which we will further investigate in the next section.

[10] Some prepositions include other image schemas at the same time. E.g. 'through' involves apart from PATH also some notion of CONTAINMENT.

Table 3: Summary of the mentioned expressions and their level in the PATH-following hierarchy

| | Expression | Level in hierarchy |
|---|---|---|
| Concrete: | Roaming the city | MOVEMENT_OF_OBJECT |
| | Meandering | MOVEMENT_ALONG_PATH |
| | Leaving Berlin | SOURCE_PATH |
| | Travelling to Berlin | PATH_GOAL |
| | Going from Prague to Berlin | SOURCE_PATH_GOAL |
| Abstract: | Going for a joy ride | SOURCE_PATH |
| | Running for president | PATH_GOAL |
| | Life is a journey | SOURCE_PATH_GOAL |
| | The circle of life | MOVEMENT_IN_LOOPS |

### 3.3 Time and processes as PATH

The conceptualisation of time has been investigated by [7]. Here we follow suit by looking at how members of the PATH-following image schema family are widely used as a conceptual metaphors for time. We consider several examples and discuss the role of PATH-following image schemas for the conceptualisation of processes in general.

One popular way to conceptualise time is as MOVEMENT_ALONG_PATH. Often, time is conceptualised as having a beginning, a START_PATH; this may be the Big Bang or the moment of creation in a religious context. Depending on the cosmological preferences, time may also be conceptualised to have an end, an END_PATH: the Big Rip or an apocalypse.

Other religious traditions embrace the notion of a 'Wheel of Time', that is time as a cyclic repetition of different aeons. The underlying image schema involves a MOVEMENT_IN_LOOPS. The same image schema is used in the conceptualisation of time within calendars: the seasons are a continuous cycle where any winter is followed by a new spring. Similarly, the hours of the day are represented on analogue clocks as 12 marks on a cycle, and the passing of time is visualised as MOVEMENT_IN_LOOPS of the handles of the clock.

The conceptualisation of time, in itself, is an interesting example for the usage of image schemas. However, the real significance is that these image schemas can be seen as providing the conceptual skeletal structure for our understanding of processes. Assume we want to understand a complex process, e.g. the demographic development of a country, the acceleration of a falling object, or the economic situation of a country. In these situations we often use two-dimensional coordinate systems where the vertical axis represents the property in question (e.g. population, speed, GDP, respectively) and the horizontal axis represents time. These coordinate systems are so useful and so widely applicable because we can conceptualise arbitrary processes as MOVEMENT-_ALONG_PATH, where the paths represent some important dimension or aspect of the process.

The importance of PATH-following image schemas for the conceptualisation of processes can be illustrated by considering *similes*. If we pick from Table 1 randomly a target domain $X$ from the first column and a source domain $Y$ from the second column, the resulting simile *$X$ is like $Y$* will be sensible. (Of course, depending on the choice of $X$ and $Y$ the simile may be more or less witty.) Note that the target domains have little or nothing in common. Thus, at least on first glance, one would not expect that one can compare them meaningfully to one and the same source domain.

The similes work because all of the concepts in the second column involve physical MOVEMENT-_ALONG_PATH, which have some pertinent characteristics. These characteristics may concern the

Table 4: PATH similes: <target> is like <source>.

| **Target Domain** | **Source Domain** |
|---|---|
| Watching the football game | the swinging of a pendulum |
| Their marriage | a marathon |
| The story | escaping a maze |
| This piece of music | a sail boat during a hurricane |
| Bob's career | a roller coaster ride |
| Her thoughts | a Prussian military parade |
| Democracy in Italy | stroll in the park |

shape of the path itself (e.g. the path of a roller coaster involves many ups and downs and tight curves, the path out of a maze involves many turns, the path of a pendulum is regular and between two points), the way the movement is performed (e.g. the movement of a sail boat during a storm is erratic and involuntary, a stroll in the park is done leisurely), and the effects the movement may have (e.g. running a marathon is exhausting, a Prussian military parade may be perceived as threatening). In each of the similes we use some of the pertinent characteristics from the source domain to describe the process from the target domain. For example, in the simile 'Bob's career is like a Prussian military parade' we conceptualise the career as a path along time (with career-related events like promotions as the sites on the path) and transfer characteristics from the movement of a Prussian military parade on this path. Thus, one way to read the simile is that Bob moves through the stages of his career in a exceptionally predictable fashion. The example illustrates how the similes work: first, we conceptualise the process in the target domain as MOVEMENT_ALONG_PATH, where the events of the process are ordered by time, and then we transfer some pertinent characteristics of the MOVEMENT_ALONG_PATH of the source domain to the target domain. This pattern is not just applicable to the concepts in Table 1. As we discussed above, any process can be conceptualised as MOVEMENT_ALONG_PATH, thus, any process could be added as target domain in Table 1. Further, any concept that involves interesting physical movement along some path could be added as source domain. Hence, the use of the image schema MOVEMENT_ALONG_PATH enables the mechanical generation of similes for processes.

Similes are a particular form of concept generation in which two domains are combined. This phenomenon is strongly connected to conceptual blending that we will discuss further in Section 6.

To summarise, in this section we have introduced the image schema MOVEMENT_ALONG_-PATH. We have seen that it is widely used in natural language and plays an important role in our understanding of time and processes. The examples show that the notion of PATH-following, at its core, is about movement along some trajectory. However, there are important differences both with respect to the spatial primitives that are involved and with respect to the shape of the PATHs. In the next section, we consider how images schemas can be represented in formal languages. One particular concern is to represent image schemas in a way that adequately captures the variety and flexibility of image schemas.

## 4 Formalising image schemas as graphs of theories

### 4.1 Previous work on formalising image schemas

Image schema research has had great impact in the cognitive sciences and in particular in cognitive linguistics. However, within computational cognitive systems, and artificial intelligence in general, it has not yet been explored to its full potential.

Looking at how image schemas can be computationally acquired, there are studies that attempt to model early cognitive development and learn from perceptual input. The connectionist model proposed by [62] learns to linguistically classify visual stimuli in accordance with the spatial terms of various natural languages. Similarly, the *Dev E-R* system by [1] is a computer model that simulates the first sensorimotor stages in cognitive development. Their system learns to distinguish and fine-tune visual clues such as nuances of colour, as well as different sizes of objects and directions of movement. Both approaches demonstrate how an artificial agent can develop cognitive abilities and language development from perceptual input.

Another study using perceptual input to simulate the development of image schemas was made by [56]. They fed video material of OBJECTs moving IN and OUT of boxes into an unsupervised statistical model in order to capture the dynamic aspects of the CONTAINMENT schema. From this, the system learned how to categorise different CONTAINMENT contexts and could in combination with a linguistic corpus generate simple CONTAINMENT-related language constructions.

These are examples of systems that learn image schemas and visual relationships from perceptual input. More commonly, work on formalising image schemas is done when the image schemas are already identified. Prominent work in this field is the work by [35, 36]. He argues that image schemas capture abstractions in order to model affordances. Working top-down rather than bottom-up as above, he uses WordNet to define noun words and connects them to spatial categorisations related to image schemas based on affordance-related aspects of meaning.

[78] build further on Kuhn's work by visualising and formalising the connections between different image schemas using bigraphs. By visually representing the topological and 'physical' image schemas relevant in built environments, they demonstrate how more complex dynamic image schemas such as BLOCKAGE could be generated using sequences of bigraph reaction rules on top of simpler static image schemas.

[69] present what they call the *Image Schema Language*, ISL. In their paper, they provide a set of diagrams that illustrate how combinations of image schemas can lead to more complex image schemas, and provide some real life examples.

[8] discuss how image schema transformations form networks that capture the relationships in polysemous words, in particular the preposition 'over' is investigated. This relates to our own approach of how to formalise and formally represent image schemas. Namely to use the hierarchical structure of image schemas demonstrated previously to represent image schemas as families of theories.

### 4.2 Image Schema Families as Graphs of Theories

In the previous sections, we argued for image schemas to be members of families, which are partially ordered by generality. In the following section, we will describe and visualise an approach

to represent the connections between image schemas, belonging to the same family. In order to discuss the problem of how more complex image schemas can be constructed through a combination of different image schemas (e.g. LINKED_PATH, MOVEMENT_IN_LOOPS), we will discuss the possible interconnection these families of theories allow. Formally, we can represent the idea as a graph[11] of theories in DOL, the *Distributed Ontology, Modeling and Specification Language* [51].

This choice is motivated primarily by two general features of DOL: (1) the heterogeneous approach, which allows for a variety of image schematic formalisations without being limited to a single logic, and (2) the focus on linking and modularity. Therefore, DOL provides a rich toolkit to further formally develop the idea of *image schema families* in a variety of directions.

In more detail, DOL aims at providing a unified metalanguage for handling the diversity of ontology, modelling, and specification languages, for which it uses the umbrella term 'OMS'. In particular, DOL includes syntactic constructs for:

1. "as-is" use of OMS formulated (as a logical theory) in a specific ontology, modelling or specification language,
2. defining new OMS by modifying and combining existing OMS (which are possibly written in different languages), and
3. mappings between OMS, resulting in networks of OMS.

DOL is equipped with an abstract model-theoretic semantics.[12] The theoretical underpinnings of the DOL language have been described in detail in [39] and [54], whilst a full description of the language can be found in [51] or (in a more condensed form) in [53].

Building on similar ideas to those underlying the first-order ontology repository COLORE[13] [23], we propose to capture image schemas as interrelated families of (heterogeneous) theories. Similar ideas for structuring common sense notions have also been applied to various notions of time [3, 73]. This general approach also covers the introduction of non-spatial elements such as 'force' as a basic ingredient of image schemas, as for instance argued for by [18] and constitute the core of [48]'s *conceptual integrations* mentioned above.

In Figure 11, some of the first basic stages of the image schema family PATH-following are presented. Ranging from Mandler's general definition presented above, of object movement in any trajectory, to more complex constructions.

The particular image schema family sketched is organised primarily via adding new spatial primitives to the participating image schemas and/or by refining an image schema's properties (extending the axiomatisation). In general, different sets of criteria may be used depending, for example, on the context of usage, thereby putting particular image schemas (say, REVOLVE_AROUND) into a variety of families. Apart from a selection of spatial primitives, other dimensions might be deemed relevant for defining a particular family, such as their role in the developmental process.

One way MOVEMENT_ALONG_PATH can be specialised is as the image schema of MOVEMENT_IN_LOOPS. Note that this change does not involve adding a new spatial primitive, but just an additional characteristic of the path. The resulting image schema can be further refined by

[11] These graphs are diagrams in the sense of category theory.
[12] The final DOL specification was submitted as a standard to the Object Management Group (OMG) in late 2015
[13] See `http://stl.mie.utoronto.ca/colore/`

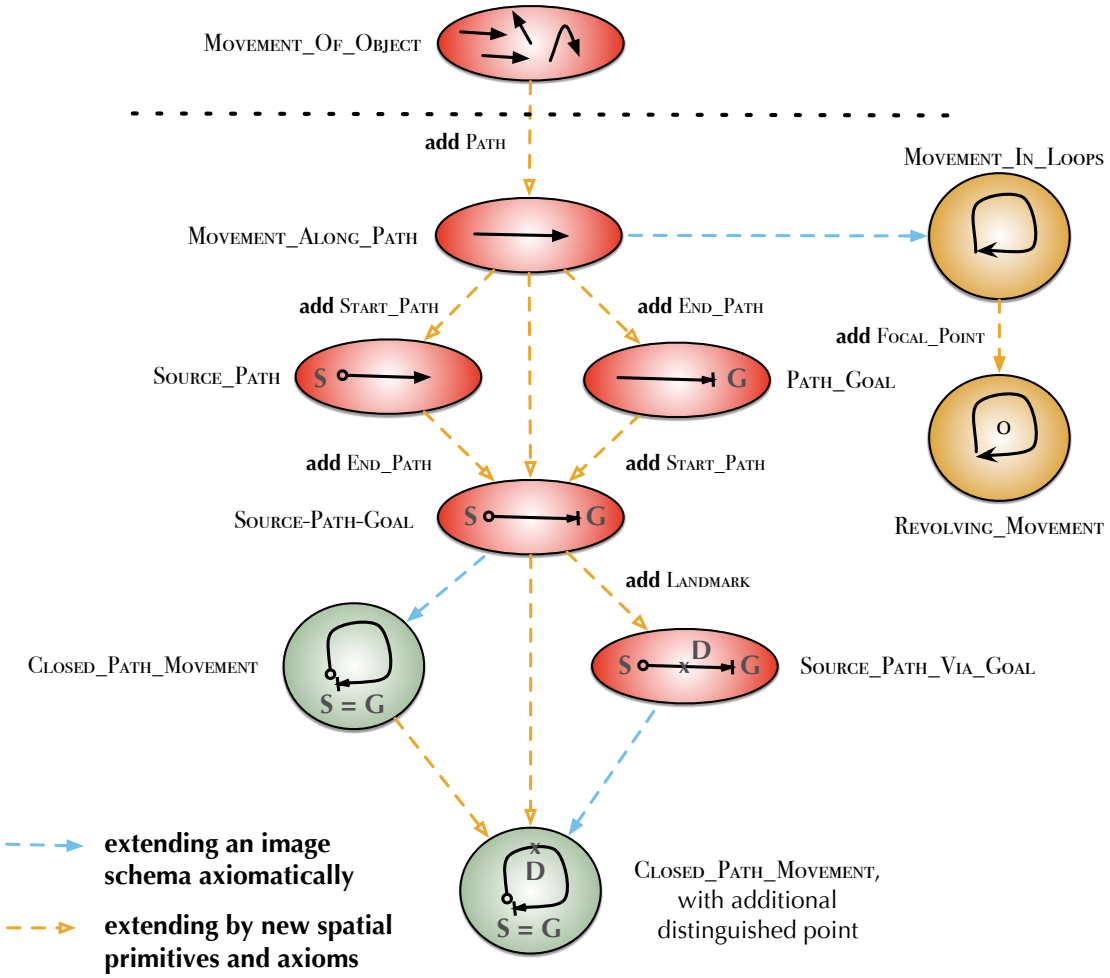



Fig. 2: A portion of the family of image schemas related to path following shown as DOL graph.

adding the spatial information of a *focal point*, which the path revolves around – this leads to the notion of *orbiting*, or, by continuously moving the orbiting path away from the focal point, to create the concept of *spirals*. Alternatively, we may change MOVEMENT_ALONG_PATH by adding distinguished points; e.g. the START_PATH, the target END_PATH, or both.

The MOVEMENT_IN_LOOPS image schema may be further specialised by identifying (the location of) the START_PATH and the END_PATH. In this case, the path is closed in the sense that any object which follows the path will end up at the location at where it started its movement. The difference between a closed path and a looping path is that the closed path has a start and an end (e.g. a race on a circular track), while the looping path has neither (like an orbit). It is possible to further refine the schema by adding more designated points (i.e. 'landmarks') or other related spatial primitives.

We will now show how the theories of image schemas and the various branching points in the graph can be characterised formally.

### 4.3 Axiomatisation of Path-Following

In this section, we present an axiomatisation of the image schemas represented in Figure 11. The focus of our axiomatisations is to capture the important differences of the branching points of the PATH-following family, not an exhaustive axiomatisation. For the sake of brevity, we will present only selected axioms in this section. A more complete axiomatisation is available at an Ontohub repository.[14]

Our axiomatisation approach is inspired by semantics in the neo-Davidsonian tradition [12, 59]. We consider image schemas as a type of event (in generality quite similar to the view defended in [10] to view image schemas as a kind of 'domain') and consider spatial primitives as thematic roles of these events. Thus, if a given image schema is enriched by adding a new spatial primitive, this is typically represented by adding a new entity (e.g. site) and a new relation (e.g. has_start_path) that determines the thematic role of the new entity in the event. As representation language we use ISO/IEC 24707 Common Logic. Common Logic is a standardised language for first-order logic knowledge representation, which supports some limited form of higher-order quantification and sequence variables [49].

For the axiomatisation of the image schemas in the PATH-following family we assume an image schema MOVEMENT_ALONG_PATH as the root of the family. MOVEMENT_ALONG_PATH is derived from a more general notion, namely MOVEMENT_OF_OBJECT. This is movement of some kind that involves only one spatial primitive, namely an OBJECT. This object plays the role of the *trajector* within the context of the MOVE. This can be formalised in Common Logic as follows:

```
(forall (m)
  (iff
    (MovementOfObject m)
    (exists (o)
      (and
        (Movement m)
        (Object o)
        (has_trajector m o)))))
```

[14] https://ontohub.org/repositories/imageschemafamily/

No additional information about what kind of object is moving and how it is moving is assumed.[15]

The schema MOVEMENT_ALONG_PATH is the result of adding a new spatial primitive to MOVEMENT_OF_OBJECT, which plays the role of a PATH.

```
(forall (m)
  (iff
    (MovementAlongPath m)
    (exists (p)
      (and
          (MovementOfObject m)
          (Path p)
          (has_path m p)))))
```

Under a PATH we understand a collection of two or more sites, which are connected by successor relationships. Each of these sites have (relative to the path) at most one successor site. The transitive closure of the successor relation defines a *before* relationship (relative to the path); and for any two different sites $x, y$ of a given path, either $x$ is before $y$ or $y$ is before $x$ (relative to the path).[16] This axiomatisation provides a representation of a quite abstract notion of MOVEMENT-_ALONG_PATH. It needs to be sufficiently abstract, since it serves as the root node for the PATH-following family. All other image schemas in the family are derived from this root by adding additional spatial primitives and/or additional axioms.

Given this notion of PATH, we can axiomatise the relationship between the PATH and the OBJECT, which characterises a MOVEMENT_ALONG_PATH. During the movement, the moving object needs to pass through all sites of the path in a temporal order, which matches the before-relationship between the sites:

```
(forall (p o m s1 s2)
  (if
    (and
      (MovementAlongPath m)
      (has_path m p)
      (has_trajector  m o)
      (before s1 s2 p))
    (exists (t1 t2)
      (and
        (Timepoint t1) (Timepoint t2)
        (during t1 m)  (during t2 m)
        (located_at o s1 t1) (located_at o s2 t2)
        (earlier t1 t2)))))
```

The image schema SOURCE_PATH is the result of adding the spatial primitive START_PATH to MOVEMENT_ALONG_PATH. We represent this with the *has_starts_path* relationship. The START_PATH of a PATH is a site on the path that is before any other site of the path:

[15] From an ontological perspective, MOVEMENT_OF_OBJECT can be seen as a kind of process (or occurrent). Thus, any adequate axiomatisation of MOVEMENT_OF_OBJECT needs to represent change over time in some form. To keep things simple, we here just quantify over time points. We assume that time points are ordered by an `earlier` relationship. Further, we use two other relationships to connect time points to processes: `(has_start m t)` means *The movement m starts at time point t* and `(during t m)` means *Time point t lies within the interval during which movement m happens*.

[16] The before-relationship is not a total order, since antisymmetry is not postulated.

```
(forall (m)
  (iff
    (SourcePathMovement m)
    (exists (s)
      (and
          (MovementAlongPath m)
          (has_start_path m s)))))
```

```
(forall (m s1 s2 p)
  (if
    (and
      (SourcePathMovement m)
      (Site s1)
      (Site s2)
      (not (= s1 s2))
      (has_path m p)
      (has_start_path m s1)
      (part_of s2 p))
    (before s1 s2 p)))
```

What distinguishes SOURCE_PATH from other movements is the following: at the start of a SOURCE_PATH movement the object that moves is located at the START_PATH:

```
(forall (m s t p o)
  (if
    (and
      (SourcePathMovement m)
      (has_start m t)
      (has_trajector m o)
      (has_start_path m s))
    (located_at o s t))))
```

Analogously, we can define PATH_GOAL as a MOVEMENT_ALONG_PATH with an END_PATH. A SOURCE_PATH_GOAL is a movement, which includes both landmarks of START_PATH and END_PATH. Thus, SOURCE_PATH_GOAL can be defined as the union (of the axioms) of SOURCE_PATH and PATH_GOAL.[17]

CLOSED_PATH_MOVEMENT is a special case of SOURCE_PATH_GOAL, where the location of the START_PATH and the END_PATH of the PATH coincide.

```
(forall (m s g)
  (if
    (and
         (has_start_path m s)
         (has_end_path m g))
    (iff
      (ClosedPathMovement m)
      (and
        (SourcePathGoalMovement m)
        (= (location_of s) (location_of g))))))
```

SOURCE_PATH_VIA_GOAL is a different way to refine SOURCE_PATH_GOAL. In this case an additional designated site is added, which lies between the START_PATH and the END_PATH of the PATH.

```
((forall (m)
  (iff
    (SourcePathViaGoalMovement m)
    (exists (s p)
      (and
        (SourcePathGoalMovement m)
        (has_path m p)
        (Site s)
        (part_of s p)
        (not (has_start_path m s))
        (not (has_end_path m s))))))
```

[17] In DOL, this is done by using the keyword 'and', which amounts to taking the model-theoretic intersection of the model classes of the theories of SOURCE_PATH and PATH_GOAL.

Both CLOSED_PATH_MOVEMENT and SOURCE_PATH_VIA_GOAL can be combined in the obvious way.

A completely different branch of the movement image schema family does not involve either START_PATH or END_PATH, but the PATH consists of a loop of sites. One way to represent this is by requiring that the before-relationship is reflexive (with respect to the path of the movement):

```
(forall (m)
  (iff
    (MovementInLoops m)
    (and
      (MovementAlongPath m)
      (forall (p s)
        (if
          (and
            (has_path m p)
            (Site s)
            (part_of s p))
          (before s s p))))))
```

The difference between MOVEMENT_IN_LOOPS and CLOSED_PATH_MOVEMENT is that in the latter case both START_PATH and END_PATH are present, they just spatially coincide. Hence, the movement is over when the object meets the target. In contrast, MOVEMENT_IN_LOOPS entails that the moving object is located at the same location more than once.

REVOLVING_MOVEMENT is a subtype of MOVEMENT_IN_LOOPS. To define it, we need to consider two additional factors: the shape of the path is elliptical, and there is a focal point, which the movement revolves around. The focal point itself is a site, but it is typically the location of an object. A detailed axiomatisation of this image schema is beyond the scope of this paper, we just provide an initial sketch:

```
(forall (m)
  (iff
    (RevolvingMovement m)
    (and
      (MovementInLoops m)
      (exists (p s)
        (and
          (has_path m p)
          (Eliptical (shape p))
          (Site s)
          (has_focal_point p s))))))
```

## 5 Image Schemas in Computational Conceptual Blending

In this section, we will illustrate how formalised families of image schemas, as just sketched above, can help in the computational modelling of concept invention, an area at the heart of AGI. More precisely, we will here focus on the highly influential framework of conceptual blending [13, 43, 71], and illustrate the foundational role that a formal theory of image schemas plays in its computational realisation.

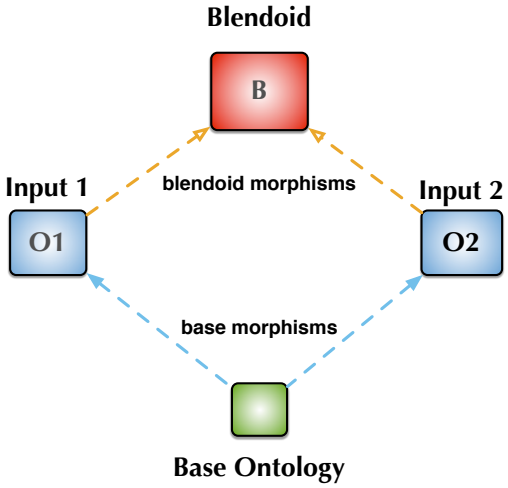
### 5.1 A crash course on conceptual blending

Introduced by [14], conceptual blending has been employed very successfully to understand the process of concept invention, studied particularly within cognitive psychology and linguistics.

The theory argues that at the heart of novel concept creation lies a combination process involving already existing knowledge and understood concepts. By merging two, or more, conceptual spaces, a blended conceptual space results. This blend contains information from both input spaces and has emergent properties due to its own unique composition. The classic example is the blend of a 'houseboat', containing merged information from the input spaces 'house' and 'boat'.[18]

One of the central aspects of blending is the the way in which 'common structure' between the input concepts is understood to steer the creation of the new concept. The 'merging' of the input spaces is moderated by this common structure, represented as the generic space, or as it is called in formal approaches, the base ontology (see Figure 3).[19] The common structure of the input spaces is understood to play a vital role in rendering the newly constructed concept meaningful, as it ensures that the blended space also contains the structure found in the generic space.

Fig. 3: 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.




However, despite this influential research, within computational creativity and AI in general, relatively little effort has been devoted to fully formalise these ideas and to make them amenable to computational techniques, but see [37, 67] for overviews.

Unlike other combination techniques, *blending* aims at creatively generating (new) concepts on the basis of input theories whose domains are thematically distinct but whose specifications share some structural similarity.

[37] describe in detail the basic formalisation of conceptual blending, as sketched by the late Joseph Goguen and discuss some of its variations [21]. Moreover, it is illustrated how the Distributed Ontology Language DOL can be used to declaratively specify blending diagrams of various shapes, and 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

[18] This and the related blend of 'boathouse' were fully formalised in [40].

[19] In the limit case, the shared structure might be trivial, and a concept such as 'red pencil' might be understood as a blend too, by simply imposing properties from one input space onto another.

repository with support for a large number of logical languages and formal linking constructs, see [40, 52]. The reasoning engine managing heterogeneous theories and computationally supporting the Ontohub repository is the *Heterogeneous Tool Set* Hets [55]. The graph for the structured theory illustrating the PATH-following family and automatically being generated by the Hets system from its formal specification is shown in Figure 4.

Figure 3 illustrates the basic, classical case of an ontological blending diagram. The lower part of the diagram shows the generic 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*.[20] Note that, in general, ontological blending can deal with more than one base and two input ontologies, and in particular, the sets of input and base nodes need not exhaust the nodes participating in a base diagram.

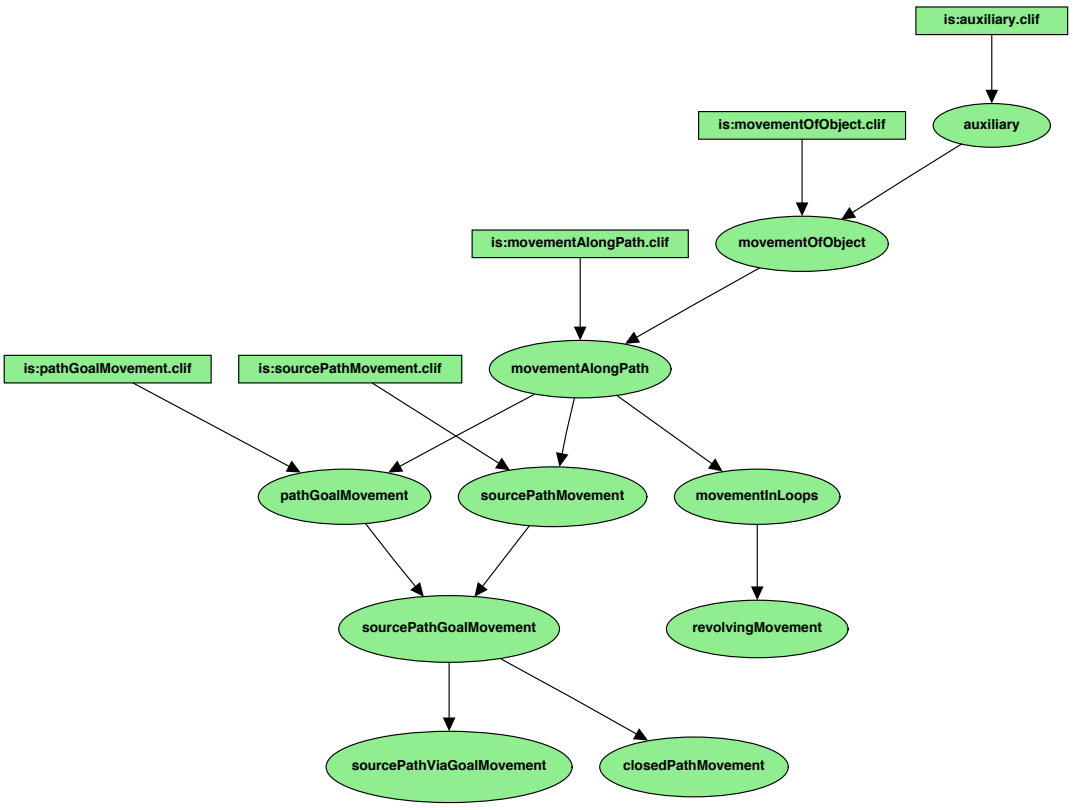



Fig. 4: The PATH-following family displayed as a structured DOL theory in Ontohub/Hets.

[20] A technically more precise definition of this notion is given in [38]. Note also that our usage of the term 'blendoid' does not coincide with the (non-primary) blendoids defined in [21].

### 5.2 Using image schemas in computational blending

One problem for conceptual blending, and related work on analogy engines (e.g. structure mapping [16, 19] and heuristic-driven theory projection (HDPT) [66]) is the generation of a 'sensible' blend. In a completely automatised system, there is currently no simple way to distinguish the blendoids that a human would consider meaningful from those that lack cognitive value. This problem grows exponentially in relation to the size of the input spaces. The larger the input spaces, the more combinations can be generated resulting in a multitude of possible blendoids, most of which will make little sense if evaluated by humans. In real life scenarios, the amount of information in the input spaces can be vast, complicating things for successful concept invention tremendously when looked at as a formal, combinatorial problem.

A proposal to explain the ease with which humans perform blending is given via the ideas of *packing and unpacking*, as well as *compression and expanding* of conceptual spaces, as outlined by [72]. These terms aim to capture how we mentally carry around ideas as compressed 'idea packages' that we can 'unpack' and utilise in different contexts on the fly. The process of packing and unpacking ideas is important for the contextualised usage of conceptual blends in various situations. Generally, the idea of *optimality principles* in blending theory is meant to account for an evaluation of the quality and appropriateness of the resulting blends [13]. However, there is currently no general formal proposal how such optimality principles could be implemented computationally, apart from some work on turning such principles into metrics for rather lightweight formal languages [60].

[26] suggested that instead of relying on purely syntactic approaches, image schemas in their role as conceptual building blocks could be used to guide the computational blending process. The principle idea here is that employing image schemas in the construction of generic spaces will not only result in a significant reduction of the number of generated blends, but will moreover filter out many of those blends human evaluators would deem meaningless. A related and complementary approach is [74], where the problem of constraining the search space was addressed by suggesting that blending is performed in a task-specific context. Here, selecting a task-specific context in a blending scenario means to simultaneously work forward from the input spaces and backward from the *desired* elements of the blend space.

In this line of thinking, one way to use image schemas in blending is to identify them as the prime ingredient for the construction of a generic space. When performing the search for common structure in the different input spaces, the search could be guided by mapping (parts of) the content of the input spaces to nodes in a library of formally represented image schemas. As image schemas hold semantic value in the form of spatial relationships, the blendoids would be based on the same content. In theory, this is similar to classic structure mapping that preserves relationships, but as image schemas model e.g. affordances [36], a blendoid will inherit such information as well.

Figure 5 shows the two basic ways of using image schemas within the conceptual blending workflow. In both cases, the image schematic content takes priority over other information the input concepts might contain. On the left, following the core model of blending described above, we first identify different spatial structure within the same image schema family in the input concepts, and then generalise to the most specific common version within the image schema family to identify a generic space, using our pre-determined graph of spatial schemas (i.e. we compute the least upper bound in the lattice). The second case, shown on the right, illustrates the situation where we first want to specialise or complete the (description of the) spatial schemas found in the input

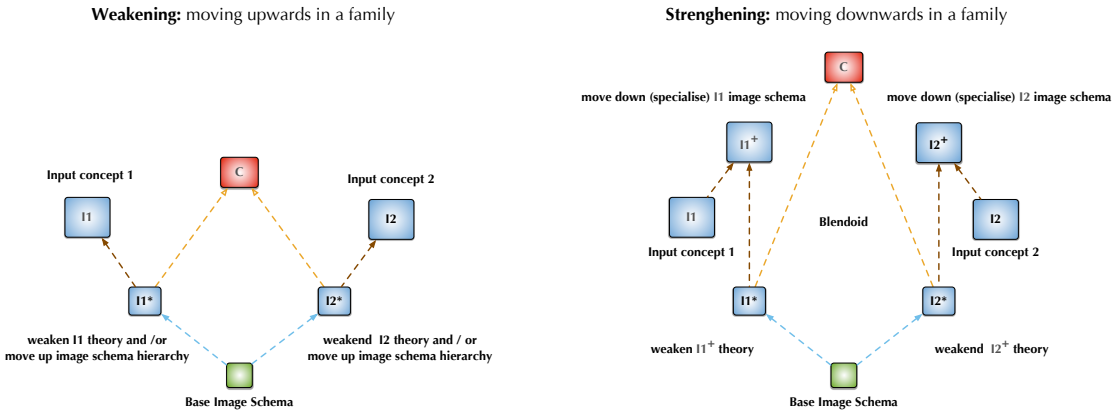



Fig. 5: Blending using common image schemas: strenghening vs. weakening.

concepts, before performing a generalisation step and to identify the generic space. This means moving down in the graph of the image schema family. Of course, also a mix of these two basic approaches is reasonable, i.e. where one input spatial schema is specialised within a family whilst the other is generalised in order to identify a generic space based on image-schematic content. Examples for both cases are described below in Sections 5.3 and 5.4.

### 5.3 The PATH-following family at work

To study how image schematic content can be used more concretely within conceptual blending, we will now look at a number of examples. In this section, we will illustrate how moving up and down within the image schema family of path-following opens up a space of blending possibilities, infused with the respective semantics of the (versions of) the image schema. In the next section, we will then discuss in more formal detail how these ideas work on a logic-based level.

As outlined in Section 3.3, processes in general can be easily combined with a variety of more specific PATH-following schemas. More specifically, we can explore the basic idea how to combine the input space of 'thinking process', which involves only an underspecified kind of 'movement of thoughts', with a second input space that carries a clearly defined path-following image schema. This leads intuitively to a number of more or less well known phrases that can be analysed as blends, including: 'train of thought', 'line of reasoning', 'derailment', 'flow of arguments', or 'stream of consciousness', amongst others. Indeed, a central point we want to make in this section is that these blends work well and appear natural because of the effectiveness of the following heuristics:[21] (i) given two input spaces $I_1$ and $I_2$, search for the strongest version $G$ of some image schema that is *common to both*, according to the organisation of a particular image schema family $\mathfrak{F}$; (ii) use $G$ as generic space; and (iii) use again $\mathfrak{F}$ to identify the stronger version of $G$, say $G'$,

[21] By 'heuristics' we mean a method that imposes rules on how to select a base (i.e. introduces a preference order on possible generic spaces) and, moreover, rules to decide which axioms to push into the blend. I.e., without any heuristics we are left to perform a randomised axiom selection, followed by an evaluation of the resulting blended concept.

inherent in one of the two inputs, and use the semantic content of $G'$ to steer the overall selection of axioms for the blended concept.

To illustrate this process informally, let us briefly consider the concepts of 'stream of consciousness', 'train of thought', and 'line of reasoning'[22].

On first inspection, the spatial schema of movement related to 'thinking' might be identified as MOVEMENT_OF_OBJECT, i.e. without necessarily identifying following a PATH at all. Indeed, in Figure 11, MOVEMENT_OF_OBJECT is marked as an 'entry point' to the path-following family. The stream of consciousness may be seen as an unguided flow of thoughts, in which topics merge into each other without any defined steps, but rather in a continuous manner. It lacks a clear START_PATH and has no guided movement towards a particular END_PATH. It resembles the more basic forms of PATH-following that, according to [48], is simply movement in any trajectory.

A 'train of thought'[23] can be conceptualised in various ways. It differs from a stream of consciousness by having a more clear direction, often with an intended END_PATH. It is possible to say that one "lost their train of thought", or that "it was hijacked" or how "it reversed its course". The 'train' may be understood as a chain-like spatial object (in which case 'losing the train' decodes to 'disconnecting the chain') or more plainly as a locomotive. In the Pixar film 'Inside Out' (2015), the 'Train of Thought' is an actual train that travels the mind of the fictional character Riley Anderson, and delivers daydreams, facts, opinions, and memories.

A 'line of reasoning' might be seen as a strengthening of this blend, where the path imposed is linear. Although a 'line', mathematically speaking, has no beginning or end, the way this expression is normally understood is as a discrete succession of arguments (following logical rules) leading to an insight (or truth). This blend might therefore be analysed to correspond to SOURCE_PATH_GOAL in [43], in which there is a clear direction and trajectory of the 'thought' (trajector).

In order to understand how blending can result in these concepts, and how image schemas are involved, let us have a closer look at the input spaces and their relationship to the PATH-following image schemas. Relevant input spaces include line (perhaps analysed as 'discrete interval'), stream/river, train/locomotive, and, as secondary input space, 'thinking process'.

'Thinking' as an input space is difficult to visualise. However, when 'thinking' is understood as a process it can be easily combined with various PATH-following notions (see Section 3.2 above). As thoughts (in the form of OBJECT) are moved around, the simplest form of thinking is MOVEMENT_OF_OBJECT. There is no START_PATH nor an END_PATH. Intuitively, it does not appear to have any particular PATH (in the sense of a spatial primitive).

A stream is characterised by a continuous flow along a PATH. Whilst a START_PATH and END_PATH can be part of a stream-like concept, like in the fleshed out concept of a river with a source and mouth, they do not constitute an essential part of the concept of stream.

For a train (understood as 'locomotive'), the concepts of a START_PATH and END_PATH has a much higher significance. The affordances found in trains are primarily those concerning going

[22] The examples presented here are chosen to illustrate the basic ideas how to employ families of image schemas in blending. It is not intended to capture fully the meaning of these terms as they are used in the psychological or linguistic literature, or indeed the subtle meaning they might carry in natural language.

[23] The expression 'train of thoughts' appears to have been first used by Thomas Hobbes in his Leviathan (1651): "*By 'consequence of thoughts' or 'TRAIN of thoughts' I mean the occurrence of thoughts, one at a time, in a sequence; we call this 'mental discourse', to distinguish it from discourse in words.*"

from one place to another. A train ride can also be seen as a discrete movement in the sense that for most train rides, there are more stops than the final destination. This results in a discrete form of the spatial schema SOURCE_PATH_GOAL.

When blending such forms of movement with the thinking process, what happens is that the unspecified form of movement found in 'thinking process' is specialised to the PATH-following characteristics found in the second input space. The result is the conceptual metaphors for the different modes of thinking listed above, where the generic space contains just MOVEMENT-_OF_OBJECT, and the blended concepts inherit the more complex PATH-following from 'train', 'stream', or 'line'.

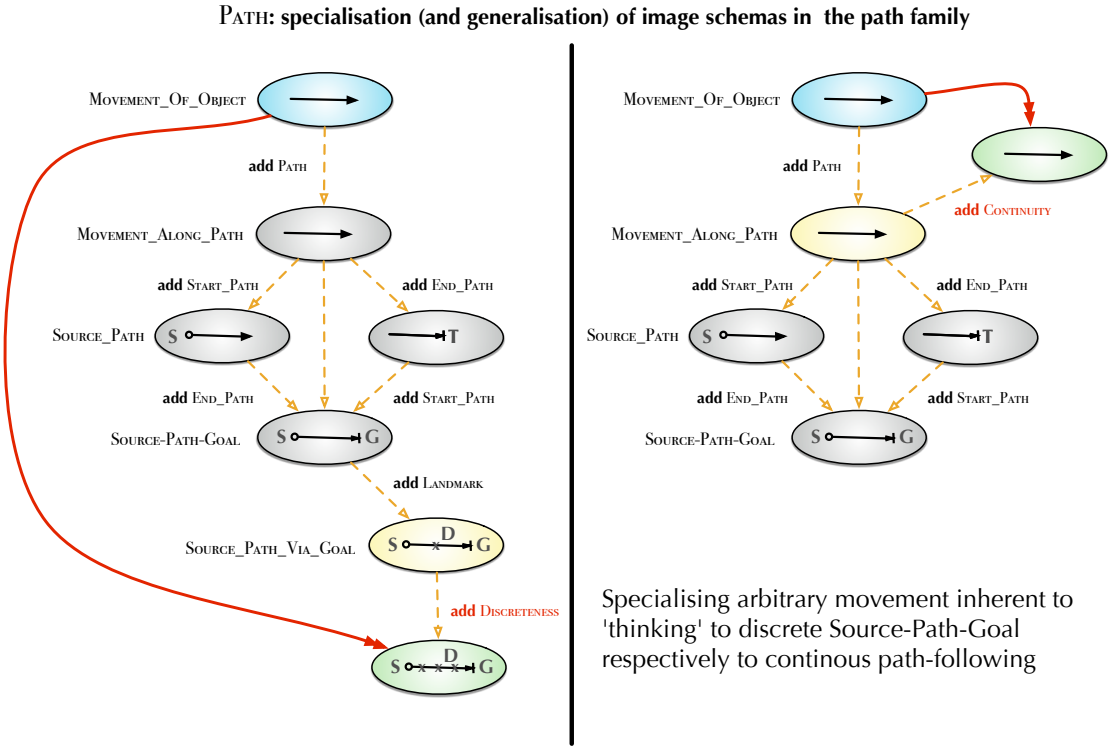



Fig. 6: How 'thinking' transforms into 'train of thought' respectively 'stream of consciousness'.

In more detail, Figure 6 shows two specialisations of the basic spatial schema of MOVEMENT-_OF_OBJECT. The first, shown on the left, specialises to a discrete version of the schema SOURCE-_PATH_GOAL with a designated element and discrete movement, supporting the 'train of thought' blend. The second, shown on the right, specialises to a continuous version of MOVEMENT_ALONG_-PATH, where an axiom for gapless movement is added to the MOVEMENT_ALONG_PATH spatial schema to support the 'flow of consciousness' blend. As a third possibility, in 'line of reasoning', we would impose additionally a linear (and perhaps discrete) path onto 'thinking'.

Fig. 7: Ontology story in DOL and Common Logic.

```
ontology story =
%% A story is defined as a telling of a plot.
(forall (x)
  (iff
     (Story x)
     (exists (y) (and (Plot y) (tells x y)))))

%% A plot consists of some events
(forall (x)
  (if  (Plot x)
       (exists (y) (and (Event y)(part_of y x)))))

%% Every event in a plot is causally connected to at least some other event in the plot.
(forall (x y)
  (if (and (Plot x) (Event y)(part_of y x))
      (exists (z) (and (Event z) (part_of z x) (or (causes y z x) (causes z y x))))))

%% Some stories  have a protagonist
(exists (x y) (and (Story x) (has_protagonist x y) (Character)))
end
```

### 5.4 Blending in depth

In Section 3.3, we discussed how PATH spatial schemas support similes, where some process (e.g. a story) is compared to some other concept (e.g. a roller coaster ride). In the same way as the elements in the PATH schema family may be used for creating similes, they can be used for conceptual blending. In this case, the PATH spatial schemas play the role of the generic space and may also be used to strengthen the input spaces. In this subsection, we discuss the process in detail and show some of the relevant axioms. The example illustrates the blending pattern from Figure 5b.

The input spaces for our blending process are *Story* and *Roller Coaster Ride*. They are formally represented in Figures 7 and 8 as axiomatisations in Common Logic with a DOL wrapper.[24] Both axiomatisations are quite weak. Stories are defined as a telling of a plot, and they may involve a protagonist. A plot consists of some causally connected events. A roller coaster ride is a scary amusement ride that follows either some steel or wooden track. The track is fast-paced and consists at least of a start, a thrill element, and an end. Note that the track is an instantiation of a PATH image schema, more specifically a SOURCE_PATH_GOAL.[25]

Because SOURCE_PATH_GOAL is embedded in the roller coaster concept, it is natural to use it as the base space in our blend. (We reuse its axiomatisation from Section 4.3.) However, SOURCE-_PATH_GOAL is not present in the space *Story*, thus we need to strengthen the concept by adding the image schematic content from SOURCE_PATH_GOAL to *Story*. This can be defined in DOL with the help of a signature map:

```
ontology linearStory = { sourcePathGoalMovement  with
        SourcePathGoalMovement |-> LinearStory ,
        Path |-> Plot  ,
        Site  |-> Event  ,
        has_trajector  |-> has_protagonist ,
```

[24] The first and the last line of each axiomatisation are DOL expressions. Their only purpose is to label these ontologies.
[25] To save space we omitted many of the axioms that realise the SOURCE_PATH_GOAL schema.

Fig. 8: Ontology rollerCoasterRide in DOL and Common Logic.

```
ontology rollerCoasterRide =
% A roller coaster ride is a kind of scary amusement ride.
(forall (x)
  (iff  (RollerCoasterRide x)
        (and (AmusementRide x) (Scary x))))

%  A roller coaster ride follows some track & involves some Person as participant.
(forall (x)
  (iff  (RollerCoasterRide x)
        (exists (y z)
          (and (RollerCoasterTrack y)(follows x y)(Person z)(has_participant x z)))))

% A roller coaster track is fast-paced. It starts with the start, has at
% least one thrill element as part & ends with the end.
(forall (x)
  (if (RollerCoasterTrack x)
    (exists (y1 y2 y3)
      (and  (fast_paced x) (starts_with x y1) (Start y1) (part_of y2 x)
            (ThrillElement y2) (ends_with x y3) (End y3)))))

% Roller coaster tracks are either made from wood or steel.
(forall (x)
  (if (RollerCoasterTrack x)
      (or (material_of x steel) (material_of x wood))))
[...]
end
```

```
        Object |-> Character ,
        has_path |-> has_plot  ,
        has_start_path |-> starts_with ,
        has_end_path  |-> ends_with  ,
        successor_of |-> causes }  and story
```

The resulting concept is a *Linear Story*, that is a story, where the protagonist participates in a linear succession of events with a clear start and ending.

While strengthening adds new information to an input space, weakening removes some information. This may be necessary, because the blend may otherwise be logically inconsistent. But even if logical consistency is not an issue, one of the input spaces may contain information that is not desirable or irrelevant for a specific blend. For example, the axiomatisation in Figure 8 provides information about the material of roller coasters, which may be removed completely. Further, the first axiom defines that roller coasters are scary amusement rides. This could be replaced by an axiom that keeps the information that roller coasters are scary, but omits the connection to amusement rides. All of these changes can be expressed in DOL with the help of filtering operations and extensions.

In this example it is not necessary to weaken the *Story* input space. On the other hand, since the SOURCE_PATH_GOAL image schema is already realised in the input space *Roller Coaster*, there is no strengthening necessary. Thus, in the case of this example $I1^+$ and $I1^*$ in Figure 5b are identical; so are $I2^+$ and $I2^*$.

The blended concept is the result of merging both of the weakened *Linear Story* $I1^*$ and weakened *Roller Coaster* $I2^*$. To achieve this is DOL we need to define the interpretations from the base

image schema SOURCE_PATH_GOAL to I1* and I2*. The following is the definition for weakened *Roller Coaster*.

```
interpretation base2rollerCoaster: base to rollerCoasterRideWeakened =
        Path |-> RollerCoasterTrack  ,
        Site  |-> RollerCoasterElement  ,
        SourcePathGoalMovement |-> RollerCoasterRide ,
        has_trajector |-> has_participant  ,
        Object |-> Person  ,
        has_path |-> follows  ,
        has_start_path |-> starts_with ,
        has_end_path  |-> ends_with
```

Analogously we define another interpretation `base2story`.[26]

By combining the two interpretations `base2story` and `base2rollerCoaster` we get a new concept: a *Thriller*. In DOL the blended concept can be defined as follows:[27]

```
ontology blend = combine base2rollerCoaster, base2plot with Story |-> Thriller
```

Figure 9 provides an overview over the whole blending diagram.

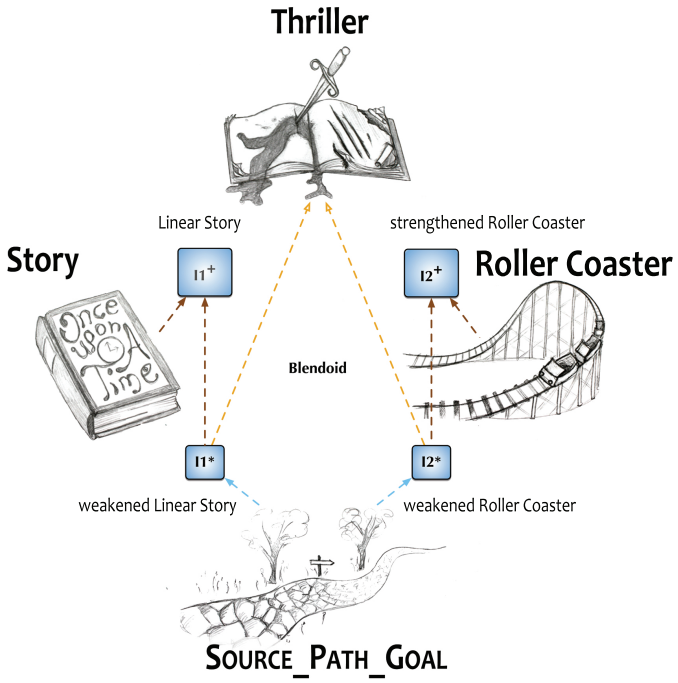



Fig. 9: Blending Thriller with the input spaces *Story* and *Roller Coaster*.

The newly defined concept, *Thriller*, inherits aspects from the input spaces *Story* and *Roller Coaster* as well as from the SOURCE_PATH_GOAL spatial schema. In particular, thrillers are scary, they have a fast-paced linear plot, which involves thrill elements, and a protagonist which participates in the events of the plot. (Figure 10 shows some of the axioms of the blended theory.)

[26] The interpretation from the base space to I1* reuses exactly the same signature map as in the strengthening process.

[27] The `with Story |-> Thriller` part of the definition just renames "Story" into "Thriller" to make the axiomatisation easier to read. The content of the concept is not affected.

Fig. 10: *Some axioms of the blended concept* Thriller

```
%% A thriller is scary.
(forall (x) (if  (Thriller x) (Scary x)))

%% A thriller has a plot and a character, who is its protagonist
(forall (x)
  (iff
    (Thriller x)
    (exists (y z)
      (and (Plot y) (has_plot x y)(Character z)(has_protagonist x z)))))

%% The plot (of a thriller) is fast-paced and involves a start, at least one thrill
%% element and an end.
(forall (x)
  (if (Plot x)
      (exists (y1 y2 y3)
        (and  (fast_paced x) (starts_with x y1) (Start y1) (part_of y2 x)
              (ThrillElement y2) (ends_with x y3) (End y3)))))


%%  If x and y are two different events in a plot p,
%%  then either x is before y (in p) or the other way round.
(forall (p x y)
  (if
    (and (Plot p) (Event y) (Event x) (part_of y p)(part_of x p) (not (= y x)))
    (or (before x y p) (before y x p))))
```

## 6 Discussion

One of the hardest problems yet to be solved in artificial general intelligence is the generation of concepts and their grounding in the environment, commonly known as the symbol grounding problem. The difficulty lies not only in establishing a relationship between objects in the real world and symbolic as well as mental representations, but also in the problem of defining 'meaning' itself.

This paper rests on the basic ideas of grounded and embodied cognition, in which physical experiences are thought of as the primary source that gives meaning to concepts. Indeed, some studies in linguistic neuroscience (e.g. [17, 70]) indicate that the cortical regions of the sensorimotor cortex are activated also in word comprehension tasks. If bodily experiences are a primary source in constituting the meaning of concepts, the symbol grounding problem can be meaningfully approached within this general framework.

We proposed in this paper an approach to computational concept invention in which image schemas, understood as embodied conceptual building blocks, were utilised in conceptual blending, the suggested cognitive machinery behind concept invention. To successfully investigate and evaluate the fruitfulness of this idea, a more comprehensive formalisation of image schemas is needed. Formalising image schemas has been a rather recent undertaking in artificial intelligence research as a means to aid computational concept invention and common-sense reasoning [21, 35, 37, 50, 78].

So far, the research on image schemas has illuminated the inherent complexity of the formalisation problem due to their abstract cognitive nature. At the same time, the incoherent scholarly terminology and corresponding key definitions make it challenging to find stable ground for fur-

ther research. In an attempt to bridge research from several research strands, the main part of this article was devoted to introducing the idea of how to formally structure image schemas as families of theories.

Our work differs from the approaches discussed in Section 4.1 by focusing on making explicit the structure of entire image schema families, using PATH as a proof of concept. While other approaches tend to look at the interconnections between particular image schemas, we have followed the psychological research of [48] to analyse formally the PATH-following image schema family concentrating on the involved spatial primitives. It is our belief that this will allow for a more fine-tuned and specialised use of image schemas in computational systems.

The most basic image schemas develop early and become more specialised with experience [63]. Currently there is no comprehensive and agreed upon list of these most basic image schemas, although the general consensus is that complex image schemas result from combining elements taken from various, more simple, image schemas [57].

It is therefore likely that there are a limited number of core image schemas that, from their most basic form, can be formally fleshed out into a structure similar to the illustrated PATH image schema graph. Finding the intersections and combinations of these basic families, thus, would mean identifying the source of the cognitive machinery behind the development of complex image schemas. These would be the spatial schemas that contain similar and overlapping spatial information and share certain spatial primitives. In our graph, MOVEMENT_IN_LOOPS could be considered to be in the intersection of the image schema families PATH-following and CYCLE. MOVEMENT_OF_OBJECT marks the entry point to the PATH-family, yet lacking itself the spatial primitive of 'path'. Here, the DOL language provides some of the tools to make such an interconnection of families formally feasible, and to gives a handle on a formal rendering of the notion of *construal* (image schema transformation) discussed by [10].

A second problem is the temporal nature of image schemas. Since image schemas are not only static but also capture change over time, any axiomatisation thereof needs to address the non-trivial problem of formally representing time. One motivation for the use of non-classical logics is the claim that these are cognitively and linguistically more adequate than classical logics involving variables and direct quantification over objects [6, 30].

Moreover, the cognitive adequateness of particular formalisms has been studied in detail (e.g. [31]). In this spirit, a large variety of temporal logics has been proposed to model various temporal aspects of natural language [61, 73]. Similarly, qualitative spatial logics have been designed to capture more adequately the way humans conceptualise and reason about space [11].

## 7 Conclusion

We have here presented an approach in which image schemas are treated as interconnected theories in a lattice (ordered by theory interpretation). This was motivated by image schematic structure found in language and the cognitive development of spatial primitives and image schemas. The main insights, we claim, support the hypotheses that the spatial primitives and their assumed properties distinguish not only the different usages in natural language and various cognitive stages, but can be systematically seen as and mapped to branching points in the lattice of image schema theories.

The benefits of this approach lie not only in the provided structuring of image schemas, but also in how formal systems may use them. By using image schemas in conceptual blending, it is our belief that computational concept invention has taken a step in the right direction. Image schemas provide a cognitively very plausible foundation for the idea of a generic space found in the theory of conceptual blending. In analogy engines, or (formal) approaches to conceptual blending [37, 72], the presented graph of image schemas can provide a method for theory weakening and strengthening based on the involved image schemas, employing basic ideas of amalgams [58]. This approach is therefore substantially different from the more syntactic-driven methods used by the Structure Mapping Engine (SME) [16, 19] or Heuristic-Driven Theory Projection (HDTP) [66, 68].

Future work will focus on extending the presented formalisation approach to other basic image schema families. This will include studying their interconnections, formal methods for their combination to construct complex schemas, as well as algorithmic approaches for detecting image schemas within given input concepts. Conversely, we hope that the systematic study of the formal interconnections between image schema families will have unifying value also for image schema research within the cognitive sciences, and provide some of the still missing systematicity to the field. Finally, we belief that the semantic and cognitive grounding of the idea of a generic space in the notion of image schema has great potential for computational realisations of conceptual blending.

**Acknowledgments.**

For this section of the deliverable, we thank the reviewers of the Journal of Artificial General Intelligence (JAGI) for constructive and valuable feedback. We would also like to thank John Bateman, Tarek R. Besold, Emilios Cambouropoulos, Tony Veale, and Mihailo Antović for valuable input and interesting discussions on topics related to this paper.

# Bibliography


[1] Wendy Aguilar and Rafael Pérez y Pérez. Dev E-R: A computational model of early cognitive development as a creative process. *Cognitive Systems Research*, 33:17–41, 2015.

[2] Kathleen Ahrens and Alicia L.T. Say. Mapping image schemas and traslating metaphors. In *Proceedings of Pacific Asia Conference on Language, Information and Computation*, pages 1–8, February 1999.

[3] J. Allen and P. Hayes. A common-sense theory of time. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 528–531, Los Angeles, CA, USA, 1985.

[4] Lawrence W. Barsalou. Grounded cognition. *Annual review of psychology*, 59:617–645, 2008.

[5] Brandon Bennett and Claudia Cialone. Corpus Guided Sense Cluster Analysis: a methodology for ontology development (with examples from the spatial domain). In Pawel Garbacz and Oliver Kutz, editors, *8th International Conference on Formal Ontology in Information Systems (FOIS)*, volume 267 of *Frontiers in Artificial Intelligence and Applications*, pages 213–226. IOS Press, 2014.

[6] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[7] Lera Boroditsky. Metaphoric structuring: Understanding time through spatial metaphors. *Cognition*, 75(1):1–28, 2000.

[8] C Brugman and George Lakoff. Cognitive Topology and Lexical Networks. In Michael Tannenhaus Stephen Small, Gary Cottrell, editor, *Lexical ambiguity resolution*, pages 477–508. 1988.

[9] Ron Chrisley. Embodied artificial intelligence. *Artificial Intelligence*, 149:131–â150, 2003.

[10] Timothy C. Clausner and William Croft. Domains and image schemas. *Cognitive Linguistics*, 10(1):1–31, 1999.

[11] A G Cohn and J Renz. Qualitative spatial representation and reasoning. In F. van Harmelen et al., editor, *Handbook of Knowledge Representation*, pages 551–596. Elsevier, Oxford, 2007.

[12] D. Davidson. The logical form of action sentences. In N. Rescher, editor, *The logic of decision and action*, pages 81–94. Pittsburgh, 1967.

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

[14] Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

[15] Jerome Feldman and Srinivas Narayanan. Embodied meaning in a neural theory of language. *Brain and Language*, 89(2):385–392, 2004.

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

[17] Vittorio Gallese and George Lakoff. The Brain's concepts: the role of the Sensory-motor system in conceptual knowledge. *Cognitive neuropsychology*, 22(3):455–79, May 2005.

[18] Peter Gärdenfors. *Embodiment in Cognition and Culture*, volume 71 of *Advances in Consciousness Research*, chapter Cognitive semantics and image schemas with embodied forces, pages 57–76. John Benjamins Publishing Company, 2007.

[19] Dedre Gentner. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[20] James J. Gibson. The theory of affordances, in perceiving, acting, and knowing. towards an ecological psychology. In Robert Shaw and John Bransford, editors, *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, pages 67–82. NJ: Lawrence Erlbaum, Hillsdale, 1977.

[21] Joseph A. Goguen and D. Fox Harrell. Style: A Computational and Conceptual Blending-Based Approach. In Shlomo Argamon and Shlomo Dubnov, editors, *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*, pages 147–170. Springer, Berlin, 2010.

[22] Gerald A. Goldin. Counting on the metaphorical. *Nature*, 413(6851):18–19, 09 2001.

[23] Michael Grüninger, Torsten Hahmann, Ali Hashemi, Darren Ong, and Atalay Ozgovde. Modular First-Order Ontologies Via Repositories. *Applied Ontology*, 7(2):169–209, 2012.

[24] Beate Hampe. Image schemas in cognitive linguistics: Introduction. In Beate Hampe and Joseph E Grady, editors, *From perception to meaning: Image schemas in cognitive linguistics*, pages 1–14. Walter de Gruyter, 2005.

[25] Beate Hampe and Joseph E. Grady. *From perception to meaning: Image schemas in cognitive linguistics*, volume 29 of *Cognitive Linguistics Research*. Walter de Gruyter, Berlin, 2005.

[26] Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus. On the cognitive and logical role of image schemas in computational conceptual blending. In *Proceedings of the 2nd International Workshop on Artificial Intelligence and Cognition (AIC-2014), Torino, Italy, November 26th–27th*, volume Volume 1315 of *CEUR-WS*, 2014.

[27] D. Hofstadter and E. Sander. *Surfaces and Essences*. Basic Books, 2013.

[28] Megan Johanson and Anna Papafragou. What does children's spatial language reveal about spatial concepts? evidence from the use of containment expressions. *Cognitive science*, 38(5):881–910, 6 2014.

[29] M. Johnson. *The Body in the Mind. The Bodily Basis of Meaning, Imagination, and Reasoning*. The University of Chicago Press, 1987.

[30] Hans Kamp. Instants, events and temporal discourse. In R. Bäuerle, C. Schwarze, and A. von Stechow, editors, *Semantics from Different Points of View*, pages 376–417. Springer, Berlin, 1979.

[31] Markus Knauff, Reinhold Rauh, and Jochen Renz. A cognitive assessment of topological spatial relations: Results from an empirical investigation. In Stephen C. Hirtle and Andrew U. Frank, editors, *Spatial Information Theory: A Theoretical Basis for GIS*, volume 1329 of *Lecture Notes in Computer Science*, pages 193–206. Springer, 1997.

[32] Arthur Koestler. *The Act of Creation*. Macmillan, 1964.

[33] K. Koffka. *Principles of gestalt psychology*. International library of psychology, philosophy, and scientific method. Harcourt, Brace and Company, 1935.

[34] Zoltán Kövecses. *Metaphor:A Practical Introduction*. Oxford University Press, USA, 2010.

[35] Werner Kuhn. Modeling the Semantics of Geographic Categories through Conceptual Integration. In *Proceedings of GIScience 2002*, pages 108–118. Springer, 2002.

[36] Werner Kuhn. An Image-Schematic Account of Spatial Categories. In Stephan Winter, Matt Duckham, Lars Kulik, and Ben Kuipers, editors, *Spatial Information Theory*, volume 4736 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2007.

[37] 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 Allain Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, Thinking Machines. Atlantis/Springer, 2014.

[38] Oliver Kutz, Till Mossakowski, Joana Hois, Mehul Bhatt, and John Bateman. Ontological Blending in DOL. In Tarek Besold, Kai-Uwe Kühnberger, Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity, Concept Invention, and General Intelligence, Proceedings of the 1st International Workshop C3GI@ECAI*, volume 01-2012, Montpellier, France, August 27 2012. Publications of the Institute of Cognitive Science, Osnabrück.

[39] 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?'.

[40] Oliver Kutz, Fabian Neuhaus, Till Mossakowski, and Mihai Codescu. Blending in the Hub—Towards a collaborative concept invention platform. In *Proceedings of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia, June 10–13 2014.

[41] G. Lakoff. *Women, Fire, and Dangerous Things. What Categories Reveal about the Mind*. The University of Chicago Press, 1987.

[42] George Lakoff and Mark Johnson. *Philosophy in the Flesh*. Basic Books, 1999.

[43] George Lakoff and Rafael Núñez. *Where Mathematics Comes from: How the Embodied Mind Brings Mathematics Into Being*. Basic Books, New York, 2000.

[44] Max M. Louwerse and Patrick Jeuniaux. The linguistic and embodied nature of conceptual processing. *Cognition*, 114(1):96–104, 2010.

[45] J M Mandler. How to build a baby: Ii. conceptual primitives. *Psychological review*, 99(4):587–604, 10 1992.

[46] Jean M. Mandler. *The Foundations of Mind : Origins of Conceptual Thought: Origins of Conceptual Though*. Oxford University Press, New York, 2004.

[47] Jean M. Mandler. On the birth and growth of concepts. *Philosophical Psychology*, 21(2):207–230, 4 2008.

[48] Jean M. Mandler and Cristóbal Pagán Cánovas. On defining image schemas. *Language and Cognition*, 0:1–23, 5 2014.

[49] Christopher Menzel. Knowledge representation, the World Wide Web, and the evolution of logic. *Synthese*, 182:269–295, 2011.

[50] Leora Morgenstern. Mid-Sized Axiomatizations of Commonsense Problems: A Case Study in Egg Cracking. *Studia Logica*, 67:333–384, 2001.

[51] 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.

[52] Till Mossakowski, Oliver Kutz, and Mihai Codescu. Ontohub: A semantic repository for heterogeneous ontologies. In *Proc. of the Theory Day in Computer Science (DACS-2014)*, University of Bucharest, September 15–16, 2014. Satellite workshop of ICTAC-2014.

[53] Till Mossakowski, Oliver Kutz, Mihai Codescu, and Christoph Lange. The Distributed Ontology, Modeling and Specification Language. In Chiara Del Vescovo, Torsten Hahmann, David Pearce, and Dirk Walther, editors, *Proceedings of the 7th International Workshop on Modular Ontologies (WoMO-13)*, volume 1081. CEUR-WS, 2013.

[54] Till Mossakowski, Christoph Lange, and Oliver Kutz. Three Semantics for the Core of the Distributed Ontology Language. In Michael Grüninger, editor, *7th International Conference on Formal Ontology in Information Systems (FOIS)*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2012.

[55] Till Mossakowski, C. Maeder, and Klaus Lüttich. The Heterogeneous Tool Set. In Orna Grumberg and Michael Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems. 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007. Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer, 2007.

[56] Sushobhan Nayak and Amitabha Mukerjee. Concretizing the image schema: How semantics guides the bootstrapping of syntax. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics, ICDL 2012*, 2012.

[57] Todd Oakley. Image schema. In Dirk Geeraerts and Hubert Cuyckens, editors, *The Oxford Handbook of Cognitive Linguistics*, pages 214–235. Oxford University Press, Oxford, 2010.

[58] Santiago Ontañón and Enric Plaza. Amalgams: A formal approach for combining multiple case solutions. In *Case-Based Reasoning. Research and Development*, pages 257–271. Springer, 2010.

[59] T. Parson. *Events in the Semantics of English: A Study in Subatomic Semantics.* MIT Press, 1990.

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

[61] Arthur N. Prior. *Past, Present and Future*. Oxford University Press, Oxford, 1967.

[62] Terry Regier. *The Human Semantic Potential: Spatial Language and Constrained Connectionism*. The MIT Press, 1996.

[63] Tim Rohrer. Image schemata in the brain. In Beate Hampe and Joseph E Grady, editors, *From perception to meaning: Image schemas in cognitive linguistics*, volume 29 of *Cognitive Linguistics Research*, pages 165–196. Walter de Gruyter, 2005.

[64] Eleanor H. Rosch. Natural categories. *Cognitive Psychology*, 4(3):328–350, 1973.

[65] M. Schiralli and N. Sinclair. A Constructive Response to 'Where Mathematics Comes From'. *Educational Studies in Mathematics*, 52:79–91, 2003.

[66] Martin Schmidt, Ulf Krumnack, Helmar Gust, and Kai-Uwe KÃ¼hnberger. Heuristic-Driven Theory Projection: An Overview. In H. Prade and G. Richard, editors, *Computational Approaches to Analogical Reasoning: Current Trends*, Computational Intelligence 548. Springer-Verlag, 2014.

[67] Marco Schorlemmer, Alan Smaill, Kai-Uwe Kühnberger, Oliver Kutz, Simon Colton, Emilios Cambouropoulos, and Alison Pease. COINVENT: Towards a Computational Concept Invention Theory. In *Proceedings of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia, June 10–13 2014.

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

[69] Robert St. Amant, Clayton T. Morrison, Yu-Han Chang, Paul R. Cohen, and Carole Beal. An image schema language. In *International Conference on Cognitive Modeling (ICCM)*, pages 292–297, 2006.

[70] Marco Tettamanti, Giovanni Buccino, Maria Cristina Saccuman, Vittorio Gallese, Massimo Danna, Paola Scifo, Ferruccio Fazio, Giacomo Rizzolatti, and Daniela Perani. Listening to action-related sentences activates fronto-parietal motor circuits. *Journal of Cognitive Neuroscience*, pages 273–281, 2005.

[71] M. Turner. The Way We Imagine. In Ilona Roth, editor, *Imaginative Minds - Proceedings of the British Academy*, pages 213–236. OUP, Oxford, 2007.

[72] Mark Turner. *The Origin of Ideas: Blending, Creativity, and the Human Spark*. Oxford University Press, 2014.

[73] J F A K Van Benthem. *The Logic of Time*. D. Reidel Publishing Company, Dordrecht, Holland, 1983.

[74] Tony Veale, Kurt Feyaerts, and Charles Forceville. E unis pluribum: Using mental agility to achieve creative duality in word, image and sound. In *Creativity and the Agile Mind: A Multi-Disciplinary Study of a Multi-Faceted Phenomenon (Applications of Cognitive Linguistics)*, pages 37–57. 2013.

[75] Tony Veale and Mark T. Keane. Conceptual Scaffolding: a Spatially Founded Meaning Representation for Metaphor Comprehension. *Computational Intelligence*, 8(3):494–519, 1992.

[76] David Vernon. *Artificial Cognitive Systems: A Primer*. MIT Press, 2014.

[77] Burton Voorhees. Embodied Mathematics: Comments on Lakoff and Núñez. *Journal of Consciousness Studies*, 11(9):83–88, 2004.

[78] L. Walton and M. Worboys. An algebraic approach to image schemas for geographic space. In *Proceedings of the 9th International Conference on Spatial Information Theory (COSIT)*, pages 357–370, France, 2009.

[79] Anna Wierzbicka. *Semantics : Primes and Universals: Primes and Universals*. Oxford University Press, UK, 1996.

[80] Nicole L Wilson and Raymond W Gibbs. Real and imagined body movement primes metaphor comprehension. *Cognitive science*, 31(4):721–731, 2007.

[81] Ludwig Wittgenstein. *Philosophical investigations*. Macmillan, New York, 1953. trans. G.E.M. Anscombe.

# Image schemas in computational conceptual blending

Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus

## 1 Introduction

Computational creativity has seen significant progress in the last decade. Using a variety of artificial intelligence techniques there are now a multitude of systems that paint, write poems and solve problem (see the recent overview [6]). In this field the notion of 'creativity' is typically understood as a cognitive process defined and evaluated based on *degree of novelty* and *usefulness* of the resulting artefact [8, 58]. While humans are creative on a daily basis, computer systems still struggle to consistently produce output that human evaluators would deem creative.

The cognitive mechanisms behind human concept generation and understanding, are still largely unknown. Cognitive psychology and developmental linguistics have yet to provide a holistic explanation of the human capacity to learn concepts and from these generate new ones. Naturally, this therefore becomes difficult to model computationally. However, there are promising approaches that describe aspects of it. This paper investigates two of these theories: *conceptual blending* and *image schemas*. Built on the cognitive mechanisms behind analogical thinking, the theories provide some of the fundamental parts to the puzzle of human concept formation.

*Conceptual Blending* is presented as the cognitive process behind creative thinking and generation of novelty in [62]. The idea is that novel concepts are created when already known (and potentially conflicting) conceptual spaces[1] are merged into a new conceptual space, which, due to the unique combination of information, exhibits emergent properties.

One critical step in blending is the identification of shared structure across the different input domains. While humans do this more or less automatically, this is one of the more complicated aspects of modelling conceptual blending formally. The main hypothesis of this paper is that image schema may play a vital role in identifying such shared structure.

While conceptual blending deals with already established concepts and knowledge, the theory of image schemas aims to explain some of the fundamental properties of concepts. Stemming from the embodied mind theory, image schemas are hypothesised to capture abstractions that model affordances related to spatio-temporal processes and relationships [35]. In the cognitive sciences, *image schemas* are identified as the fundamental patterns for the cognition of objects, which are perceived, conceptualised and manipulated in space and time [46]. Examples of image schemas, proposed in the literature, are CONTAINER, SUPPORT and SOURCE_PATH_GOAL.

In this paper, we argue that combining conceptual blending with image schemas may not only shed light on the phenomenon of concept generation and creative thinking in humans, but also provide a useful tool for computational concept invention in computational creativity [36, 60].

The paper is structured as follows: in Section 2, the theory of image schemas is introduced. This section also includes an illustration of the ubiquity of image schemas in existing applied

[1] These are also called mental spaces in [12] and are not to be confused with the 'conceptual spaces' in the sense of Gärdenfors [14].

ontologies, and a discussion of related (formal) work on image schemas. This is followed, in Section 6, by a brief introduction of conceptual blending and a discussion on how conceptual blending can be computationally modelled and implemented. Section 4 discusses how image schemas can provide heuristics in the computational blending process. As these heuristics are based on organising image schemas into families of closely related theories rather than seeing them as individual theories, this idea is discussed in more details in Section 5, including a discussion of formal and algorithmic aspects of the proposal. We conclude the paper with a short summary and outlook to future work.[2]

## 2 Image Schemas

This section presents the basic theory of image schemas. We begin, in Section 2.1, by introducing the central ideas with the help of a number of examples. We continue in Section 2.2 with an analysis of definitions of the notion of image schema found in the literature. We then, in Section 2.3, illustrate the prevalence of concepts closely related to image schemas in existing applied ontologies, before we conclude this introduction to image schemas with a discussion of related (formal) work in Section 2.4.

### 2.1 The basic idea illustrated by examples

Embodied theories of cognition [2] emphasise bodily experiences as the prime source for concept formation. Based on this cognitively supported view [13], the theory of image schemas suggests that our conceptual world is grounded in the perceptive spatial relationships between objects.

Founded on psychological research [44], the theory states that image schemas are formed as infants have repeated perceptual experiences, e.g. a plate being placed on a table. From this a generalisation emerges, an image schema, capturing the spatial relationships between the objects involved in an event. In the mentioned example, the image schema of SUPPORT is learnt. The understanding that plates can be placed on tables can be generalised and analogically transferred to other situations and objects. This means that infants who have learnt the SUPPORT schema through exposure to a plate on a table also grasp the notion of a book lying on a desk, as this represent the same spatial relationship. The more experience an infant has with a particular image schema, the more it becomes fine-tuned to accommodate different situations. Mandler [45] describes how children can be observed to mentally expand image schemas such as the SUPPORT schema by adding information, for example when understanding that a large part of an object needs to be on the supporting surface.

Another image schema example is the notion of CONTAINER, the notion that an object can be within a border (two-dimensional), or inside a container (three-dimensional). The image schema also includes the events of entering and exiting [3].

The CONTAINER schema is one of the most investigated image schemas [30] as it is one of the first to be developed [43], and since the relationships of enclosure and containment are essential

[2] This paper is a revised and extended version of [25].

[3] It can be argued that IN and OUT are by themselves image schemas, or spatial primitives. For now we include them under the umbrella schema of CONTAINER

for understanding our physical surroundings. It forms early as infants are immediately exposed to many situations in which objects are contained within one another, e.g. an embrace, lying in a crib, going into a house, eating food, etc.

One important aspect of image schemas is that they can be combined with one another. The image schema PATH can easily merge with the image schema LINK, leading to the more complex image-schematic concept LINKED_PATH. As PATH illustrates a movement through space, and LINK illustrates the causal relationship between two (or more) objects, a LINKED_PATH represents joint movement on two paths; e.g., a truck and trailer moving along a highway, or the joint movement of two separate magnets.

The 'cognitive benefit' of image schemas is to provide a means for information transfer. The conceptual abstraction that constitutes the image schema can be utilised to explain unknown relationships and affordances of objects. The core idea is that after an image schema has been formed, it can be generalised and the structure can be transferred through analogical reasoning to other domains with similar characteristics [43]. That is, an image schema structure may be used as a conceptual skeleton in an analogical transfer from the concrete spatial domain of the image schema to another domain. This target domain may involve quite abstract concepts. Traces of this can often be seen in how language is used to explain more abstract concepts. It can be argued that much of metaphorical language is based on sensory-motor experiences and, thus, involves image schemas.

For example, processes and time are often conceptualised as objects and spatial regions. Expressions such as 'we meet *on* Thursday', map information from a concrete situation such as 'a book on a table' to the abstract process and time period. Another example is our conceptualisation of relationships like love or marriage, which also are often based on spatial metaphors. For example, one way to view a marriage is as LINKED_PATH, where the PATH represents how two spouses move together trough time and the LINK between them is the bond they share. A sentence like *Their marriage chains them together* works only if one conceptualises the relationship as a LINKED_PATH, because it reinterprets the LINK as an element that constraints the movements of both lovers. Alternatively, marriage may also be conceptualised as CONTAINER. This is reflected by metaphors like 'marriage is a prison', 'marriage is a safe harbour', and 'open marriage'. Depending on whether one chooses CONTAINER or LINKED_PATH as a base for the conceptualisation of marriage, a different vocabulary and different metaphors are supported.

The examples illustrate how image schemas may be used to conceptualise an abstract domain. As mentioned above, the first image schemas are developed by infants at an early stage where abstract thought is not yet present. This illustrates how concrete reasoning involving physical objects can provide the basis for the conceptualisation of the world and the formation of more abstract concepts.

### 2.2 Defining "image schema"

The term "Image schema" is hard to define properly. Image schemas are studied in several disciplines and from various perspectives, including neuroscience [57], developmental psychology [43], cognitive linguistics [24] and formal approaches [1]. This broad range of research has lead to incoherence in the use of terminology. Also, the disputed relationship between socio-cultural as-

pects and the neurobiology of embodied cognition [23] complicates the literature on image schema research.

Oakley defines an image schema as *"... a condensed re-description of perceptual experience for the purpose of mapping spatial structure onto conceptual structure"* [53, p. 215]. Mark Johnson describes them as *"...a recurring, dynamic pattern of our perceptual interactions and motor programs that gives coherence and structure to our experience"* [31, p. xiv]. Kuhn [35] considers image schemas as the pre-linguistic structures of object relations in time and space.

One issue of these explanations of image schemas is that they do not provide individuation criteria. Hence, it is hard to evaluate whether a proposed image schema qualifies as such or not. The situation is complicated by the fact that image schema may change and become more specialised during the development of a child [46]. It is sometimes not obvious whether two conceptual structures are just variants of the same image schema or whether they are different image schemas.

One important attempt to structure the technical terminology of image schemas is made by Mandler and Pagán Cánovas [46]. In their paper they suggest to refine the umbrella term 'image schema' by distinguishing three different levels (p.17):

1. *Spatial primitives*. The first building blocks that allow us to understand what we perceive: PATH, CONTAINER, THING, CONTACT, etc.
2. *Image schemas*. Representations of simple spatial events using the primitives: PATH OF THING, THING INTO CONTAINER, etc.
3. *Schematic integrations*. The first conceptual representations to include non-spatial elements, by projecting feelings or non-spatial perceptions to blends structured by image schemas.

From our perspective, this terminology provides the benefit of clearly distinguishing between image schemas and their building blocks (the spatial primitives). An image schema always represents an event and, thus, has some temporal dimension. The spatial primitives are the components that are participating in the event. E.g., according to this terminology PATH is not an image schema but a spatial primitive. In contrast, MOVEMENT ON PATH is an image schema. Another benefit is that it provides a clear criterion for distinguishing two image schemas (or schematic integrations): if $x$ and $y$ involve different spatial primitives, then $x$ and $y$ are different.[4]

Mandler and Pagán Cánovas approach provides a useful way to explain how conceptualisations are refined: an image schema is a representation of some kind of spatial event involving a number of spatial primitives. Hence, an image schema may be enriched by adding spatial primitives, yielding a more complex image schema. E.g., by adding the spatial primitives CONTAINER and INTO to the image schema MOVEMENT ON PATH, we obtain the schema MOVEMENT ON PATH INTO CONTAINER. This new image schema is more specific and less universally applicable. However, it provides more specific information when it is utilised conceptualising analogous situations. It follows that image schemas can be ordered into a hierarchy ranging from general image schemas, which contain only few spatial primitives, to more specific image schemas, which contain more

[4] Note that this is a sufficient condition, but not a necessary one, since two different representations may involve the same spatial primitives arranged in different ways.

spatial primitives.[5] Hence, image schemas do not exists in isolation but can be organised (at least) with respect to their (shared) spatial primitives. This observation is discussed further in Section 5.

In the following we continue to use "image schema" as the umbrella term for the three levels of conceptualisations. To avoid any ambiguity, we will refer to image schemas in the sense of Mandler and Pagán Cánovas as *spatial schemas*.

### 2.3 The ubiquity of image schemas

Image schematic notions play a central role in many efforts aiming to capture common sense knowledge. In this section, we illustrate the ubiquity of notions closely related to image schemas in existing applied ontologies. We will focus on the notion of CONTAINER and discuss some prominent ontologies that incorporate them in various ways. Similar overviews could be generated for other prominent image-schematic notions such as 'PATH', 'LINK' and 'SUPPORT'.

Image schematic notions can be found early on in efforts such as building the Cyc knowledge-base, [42], or in the collections[6] of common sense modelling problems. Morgenstern's 'A Case Study in Egg Cracking' [47] contains extensive axiomatisations of variants of containment, and Cyc includes a variety of notions of containment and path-following at its most general levels of knowledge modelling and categorisation.

'Containment' is a crucial notion in areas such as geography and transportation [11], anatomy and bio-medicine in general [61], linguistics and cognition [3, 55], or indeed cooking [33].

The relevant notions of containment range from down-to-earth notions of containment such as 'holding milk in a cup' to semiotic and information-theoretic notions such as 'signs holding information' or '.tex files holding UTF8 characters' to fully abstract versions such as the 'class containing all twin primes'. In addition to the variety of concrete versions of containment, also the levels of formalisation differ dramatically, namely from extensive (at least) first-order based axiomatisations such as in Morgenstern [47], to more light-weight axiomatisations as found e.g. in GUM-Space[7] [3], to mere annotation of concepts or relations, as it is common practice in bio-medical terminologies and ontologies (see below).

We will now present a number of concrete containment notions as they can be found in prominent ontologies. Namely, we will give examples of containment notions from the areas or architecture, natural language, biomedicine, and cultural heritage. We will list them roughly in the order of concrete to abstract.

The Industry Foundation classes IFC[8] is an object-oriented data model to support data exchange in (among others) the areas of architecture and built environments, and can be seen as an application ontology. Notions of contact, containment, and composition are central to such engineering contexts (see e.g. [7, 28]). As described in [4]:

[5] In their list of spatial primitives, Mandler and Pagán Cánovas include MOVE, ANIMATED MOVE, and BLOCKED MOVE. This seems to suggests that the spatial primitives are ordered into a subtype hierarchy, since both animated movement and blocked movement are a kind of movement.

[6] See `http://www-formal.stanford.edu/leora/commonsense/`

[7] See `http://ontologydesignpatterns.org/wiki/Ontology:GUM-Space`

[8] IFC is registered by ISO and is an official International Standard ISO 16739:2013.

> The compositional aspect of the connectivity model supports three possible relationships: aggregation, containment and nesting. [...] Containment implies a stronger form of composition, where the components cannot be considered independently: where the definition of the whole element depends on the definition of its parts and the parts depend on the existence of the whole element. [...]

When investigating formal reasoning approaches for such notions of containment, often qualitative spatial calculi such as the Region Connection Calculus (RCC8) are employed [54], as discussed in [7].

Moving on from engineering to natural language, according to WordNet, a 'container' is described as a noun:

> container (any object that can be used to hold things)

In addition to this rather general version of containment (which however seems to exclude abstract containers), WordNet lists the most typical instantiation as:

> (especially a large metal boxlike object of standardised dimensions that can be loaded from one form of transport to another)

Moreover, the Synset for 'container' reports about 100 direct hyponyms, reflecting the prevalence of container-like terms in natural language, and includes terms such as 'basket', 'spoon', and 'time capsule'.

The linguistic ontology GUM (the 'Generalised Upper Model') [3] specifies detailed semantics for linguistic spatial expressions. GUM-Space (the space-related module of GUM) specifies 'containment' as a specific kind of 'FunctionalSpatialModality' exhibiting 'Control', namely:

> The reified functional relation holding between two spatial objects $x$ and $y$, such that $x$ functionally contains $y$; $x$ need not spatially contain $y$. An example of an expression falling into this category is: "The apple is *in* the bowl". Here, the apple does not necessarily need to be spatially contained in the bowl (no topological containment).

The GUM ontology was axiomatised both in first-order logic and description logic variants.[9]

Quite in contrast to this, though containment is an important notion in biomedicine and can be found in a large number of ontologies hosted on Bioportal[10], here the meaning of 'containment' is typically specified via annotations. E.g., according to the Ontology of Biomedical Investigations:[11]

> A device that can be used to restrict the location of material entities over time.

Notice that unlike the variants of containment notions mentioned above, we here find an explicit conceptualisation of the *temporal aspects* of containment. A more detailed formal treatment of various relations relevant in biomedical ontologies can be found in [61]. There, 'containment' is characterised in first-order logic with the help of mereological notions. The informal characterisation is:

[9] See `http://www.ontospace.uni-bremen.de/ontology/gum.html`
[10] See `http://bioportal.bioontology.org`
[11] See `http://bioportal.bioontology.org/ontologies/OBI`

> Containment is location not involving parthood, and arises only where some immaterial continuant is involved. [...] Containment obtains in each case between material and immaterial continuants, for instance: *lung contained_in thoracic cavity*; *bladder contained_in pelvic cavity*. Hence containment is not a transitive relation. [61]

We close this section with an example of an abstract, non-physical notion of containment, namely the idea of a document as a container of 'information'. The CIDOC [10] Conceptual Reference Model (CRM) provides an ontology for concepts and information in the domain of cultural heritage and museum documentation.[12]

We here find the abstract notion of a document as a container of 'information', e.g. the CIDOC notion of an 'Information carrier':

> This class comprises all instances of E22 Man-Made Object that are explicitly designed to act as persistent physical carriers for instances of E73 Information Object.

Similarly, motivated by the need to cover concepts related to UML to model aspects of information systems, the Cyc knowledge base was enriched by adding notions of 'abstract containment' [56].

### 2.4 Related work on formalising image schemas

Image schemas are a well studied field in research on cognitive linguistics and developmental psychology. Recently, more computationally-oriented research has shown an increased interest in image schemas as a route to approach new (partial) solutions to the symbol grounding problem and to aid computational concept invention [18, 34, 36, 47].

Lakoff and Núñez [41] used image schemas extensively in their reconstruction of abstract mathematical concepts using blending and image schemas. Working from the perspective that all of mathematics can be deduced from the body's interactions with its environment, they give a detailed account on how image schemas provide some of the conceptual principles that provide a grounding of abstract concepts.

While Lakoff and Núñez's effort is not a formalisation of image schemas, their attempt to ground mathematics in embodied cognition has been further developed and formalised. Guhe et al. [22] account for the ideas in [41] by formalising in first-order logic some basic mathematical constructs such as the *measuring stick*, *motion along a path*, and *object construction*. Using the analogy engine Heuristic Driven Theory Projection, HDTP, they illustrate how generalisations such as image schemas could be used to transfer information in a computational system. Their system uses anti-unification to find the common structure in both source and target domain. This common structure is used to transfer information to the target domain from the source.

St. Amant et al. [1] introduced the Image Schema Language, ISL, in which they discuss how image schemas can be represented and simulated computationally. They argue that their representation provides a structured image schema description of a situation. They use three different scenarios to discuss simulations of image schemas: a Chess game, military tactics, and a robot simulation. The Chess game example is particularly interesting as it accounts for the two-dimensional,

[12] It is registered as international standard (ISO 21127:2014) for the controlled exchange of cultural heritage information.

spatial relationships between the pieces on the Chess board, constrained by the rules of the game. Using a combination of the image schemas CONTAINER, LINK and PATH, they illustrate how the board configurations can be viewed from higher conceptual perspectives (rather than simply as spatial configurations).

Kuhn [34] presented another approach where he used Wordnet to extract meaning from words, and employed the programming language Haskell to generate testable models. In [35], he extended his previous image schema research by presenting a method to account for spatial categorisation and developing an algebraic theory formalising image schemas. Here he argues that the image schemas capture the abstractions essential to model affordances. For example, a cup is a cup because it can contain liquid, or an object is a vehicle when it affords transportation. With Kuhn's reasoning a vehicle can be described with a combination of the image schemas SUPPORT (alternatively CONTAINER) and PATH.

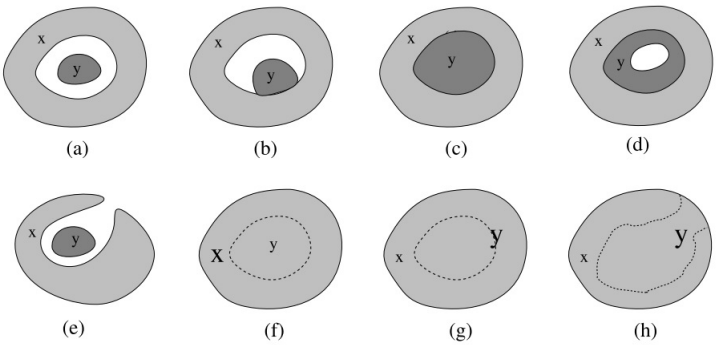



Fig. 1: The eight variations of CONTAINER as discussed in Bennett and Cialone [5].

Acquired from natural language, Bennett and Cialone [5] formally represented several different kinds of CONTAINER schemas. They distinguish eight different spatial CONTAINER relationships and their mappings to natural language constructs, illustrated in Figure 1. Their work also demonstrates the non-trivial nature of formalising image schemas, and that there are many closely related variants of any given image schema.

## 3 Conceptual Blending

### 3.1 A short introduction to conceptual blending

The theory of *Conceptual Blending* was introduced during the 1990's as the cognitive machinery for novel concept generation [12]. The theory aims to explain the process behind creative thinking. It has strong support from research in cognitive psychology and linguistics [20, 32, 65] as well as in more computational areas [18, 64].

According to conceptual blending theory, generation of novel concepts occurs via the combination of already existing ideas and knowledge. It is suggested that such novel concepts are selective and 'compressed' combinations, or blends, of previously formed concepts, building on

the notion that all novel generation builds from already existing knowledge. This cognitive process is thought to happen as two, or more, input domains, or information sources, are combined into a new domain, the blended domain, see Figure 2. The blend inherits some of the attributes and relationships from the source domains and at the same time the unique mix allows the blends to have emergent properties that are unique to each particular blend.

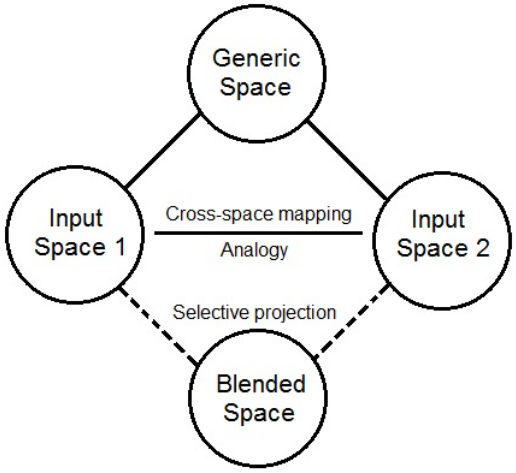



Fig. 2: The blending process as described by Fauconnier and Turner [12].

Conceptual blending can be compared to the cognitive mechanisms behind analogical reasoning. In analogical reasoning information flows from a source domain to a target domain by using cognitive structure-mapping mechanisms. Conceptual blending is comparable insofar it employs a search for 'similar structure' in the two input domains, information then gathered in the generic space, the base space[13]. The abstracted structure found in the base ontology is later used to structure the blend as well.

Many monsters are examples for conceptual blends. For example, a griffin is a fictive creature with the body and the tail of a lion and with the head and the wings of an eagle. The blend of the two creatures does not just involve the physical attributes of the animals, but also the characteristics associated with them. The lion provides attributes such as strength and power, and the eagle precision and capacity for flight. Hence, the blended creature has the skills to master both land and sky.

The griffin exemplifies *one particular* blend of the two input spaces *lion* and *eagle*. There are other possibilities to blend a monster based on these two concepts. For example, one could consider an 'inverted griffin', which has the head of the lion and the body of the eagle but no wings. A third possible monster is a creature which has the shape and strength of a lion but cannot use its strength because of its fragile bird-like bone structure. The last example shows that not all blends are equally successful. In order for the blend to be considered creative, the blend needs to be "useful" [8]. Given the task of blending a monster, a successful blend is required to produce a dangerous creature – a lion with brittle bones does not meet this requirement as well as a griffin.

[13] Introduced by Fauconnier and Turner as *generic space*, the notion carries the name *base space* or *base ontology* in formal approaches.

The blended space preserves the information from the generic space. However, usually only some selected features of the input spaces are retained. In the griffin example, the generic space contains the head, the body, and two limbs of a vertebrate. In the blend, the head in the generic space is mapped to the head of the lion and the head of the eagle, respectively. The same holds for the body. In contrast, the two limbs are mapped to the forelimbs of the lion and the hindlimbs (legs) of the eagle. For this reason, the griffin has six limbs, namely two wings of the eagle, two hindlegs from the lion and two forelegs, which are inherited from both input spaces. Since the shape and features of lion legs and eagle legs are mutually exclusive (e.g., one has hair and the other has feathers), the forelegs of the griffin cannot inherit all properties from both input spaces. Thus, griffins forelegs are usually conceptualised as exemplifying either only the features of one animal or as inheriting a consistent subset of features from both input spaces.

For humans conceptual blending is effortless. We are able to create new blends spontaneously and have no difficulty to understand new conceptual blends when we encounter them. This include the selection of suitable input spaces, the identification of a relevant generic space, the identification of irrelevant features of the input spaces, the performance of the blend, and the evaluation of the usefulness of the blend. In contrast, for an automated system each of these steps provides a significant challenge. In the next section we discuss a formal, logic-based model for conceptual blending.

### 3.2 Formalising conceptual blending

We formalise conceptual blending following an approach based on Goguen's [16] work on *algebraic semiotics* in which certain structural aspects of semiotic systems are logically formalised in terms of algebraic theories, sign systems, and their mappings. In [18] 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 languages from the area of algebraic specification, e.g. OBJ3 [19].

In [29, 37, 39], an approach to computational conceptual blending was presented, which is in the tradition of Goguen's proposal. In these earlier papers, it was suggested to represent the input spaces as ontologies (e.g., in the OWL Web Ontology Language[14]). The structure that is shared across the input spaces, i.e. the generic space, is also represented as an ontology, which is linked by mappings to the input spaces. As proposed by Goguen, the 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 [17, 37, 38].

The inputs for a blending process (input concepts, generic space, mappings) can be formally specified in a *blending diagram* in the Distributed Ontology, Model, and Specification Language (DOL).[15]

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

[15] Regarding blending diagrams as displayed in Figure 3, notice the following discrepancy in terminology and in the way the basic blending process is visualised. In the cognitive science literature following [12], conceptual blending is

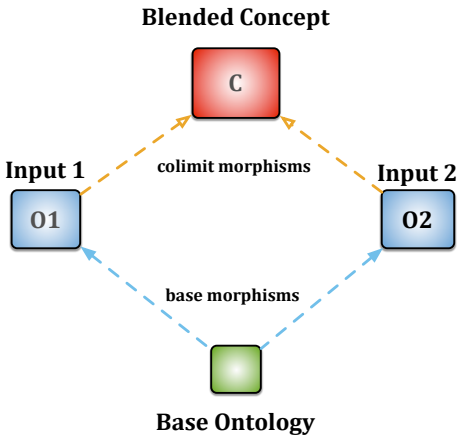



Fig. 3: The blending process as described by Goguen [18].

DOL is a metalanguage that allows the specification of (1) new ontologies based on existing ontologies, (2) relations between ontologies, and (3) networks of ontologies, including networks that specify blending diagrams. These diagrams encode the relationships between the base ontology and the (two or more) input spaces. The blending diagrams can be executed by the *Heterogeneous Tool Set* HETS, a proof management system. HETS is integrated into Ontohub,[16] 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, as seen in Neuhaus et al. [50]. An extensive introduction to the features and the formal semantics of DOL can be found in [48].

As illustrated with the example in the previous section, a critical step in the blending process is the identification of the common structure of the generic space and its mapping to the input spaces. The structural similarity between conceptual blending and analogical thinking suggests to investigate and apply approaches to analogical reasoning as tools for computational conceptual blending.

One important theory in analogical research is the Structure Mapping Theory [15]. It claims that analogical reasoning is characterised by the relationships between objects rather than their attributes. Following this idea is the analogy engine Heuristic Driven Theory Projection, HDTP [59]. HDTP computes a 'least general generalisation' $B$ of two input spaces $O1$ and $O2$. This is done by anti-unification to find common structure in both input spaces $O1$ and $O2$. HDTP's algorithm for anti-unification is, analogously to unification, a purely syntactical approach that is based on finding matching substitutions.[17]

visualised as shown in Figure 2, with a *generic space* at the top identifying commonalities. In the technically oriented literature following [18], the formalisation of this process is represented as a diagram as shown in Fig 3. This kind of diagram is on the one hand an upside-down version of the first illustration, following traditions of category theory to put the 'simpler' objects at the bottom of a diagram. On the other hand, it replaces the term 'generic space' with 'base space', partly because of a clash with mathematical terminology. In our work on formalisation of blending, we will make no technical difference between 'generic space' and 'base space' and treat them as synonymous.

[16] `www.ontohub.org`

[17] There are several other methods for finding generalisations. One example is the Analogical Thesaurus [63] which uses WordNet to identify common categories for the source and target spaces.

While this is an interesting approach, it has a major disadvantage. Typically, for any two input spaces there exists a large number of potential generalisations. Thus, the search space for potential base spaces and potential conceptual blends is vast. HDTP implements heuristics to identify interesting anti-unifiers; e.g., it prefers anti-unifiers that contain rich theories over anti-unifiers that contain weak theories. However, since anti-unification is a purely syntactical approach, there is no way to distinguish cognitively relevant from irrelevant information. As a result, an increase of the size of the two input ontologies leads to an explosion of possibilities for anti-unifications.

## 4 Blending with Image Schemas

Instead of relying on a purely syntactical approach to blending, the semantic content found in image schemas can be employed to help guiding the blending process. The basic idea here is that in order to identify common structure sufficient for defining a useful generic space for two (or more) given input spaces, we search for shared image-schematic information rather than *arbitrary* structure. As discussed above, a vast space of blends opens up if we work with more unconstrained resp. syntax-based shared structure in the generic space. Given the powerful role that image schemas generally seem to play in human conceptual (pre-linguistic) development, the working hypothesis is that the semantic content and cognitive relevance given by identifying shared image schemas will provide valuable information for constructing and selecting the more substantial or interesting possible blends.

This section therefore serves a twofold purpose. First, we will demonstrate in Section 4.1 that image schemas may enable similes based on a shared containment structure, and show how this extends to supporting the conceptual blending process. Second, in Sections 4.2 and 4.3, we will give formalised versions of blends where image schemas play a crucial role, showing that the gulf between the cognitive relevance of image schemas and formal, logic-based concept blending can be bridged.

### 4.1 Blending with Image Schemas in Natural Language

In this section we show that image schemas can provide the base structure for the blending of a wide variety of concepts.

Consider the concepts *Space Ship*, *North Korea*, *Spacetime*, *Marriage* and *Bank account*. Note that these concepts differ significantly. However, all of them can be construed as various kinds of *containers*. This is obvious in the case of space ships, which may contain passengers and cargo. Geopolitical entities like North Korea instantiate the CONTAINER schema, since they have boundaries and people may be inside and outside of countries. *Spacetime* conceived as a container is a particularly interesting case since it implies the notion of inertial frames of reference, which is arguably inconsistent with the Theory of Relativity [9]. This does not prevent science fiction writers to construe spacetime as a container for planets, suns and other things; in many fictive stories it is possible to leave and return to the universe (e.g., by visiting a 'parallel universe'). While the first three examples are physical entities, *Marriage* is a social entity. Thus, in the literal physical sense marriage cannot be a container. Nevertheless, we use vocabulary that is associated with containers to describe marriage. E.g., one can *enter* and *leave* a marriage, some marriages are *open*, others are *closed*, and people may find happiness *in* their marriage. Similarly, a *bank account* may *contain*

Table 1: CONTAINER similes: < target> is like a <source>.

| **Target Domain** | **Source Domain** |
|---|---|
| This space ship | leaky pot |
| North Korea | prison |
| The universe | treasure chest |
| Their marriage | bottomless pit |
| My bank account | balloon |

funds, and if it is *empty* we can put some additional funds *into* the account and take them *out* again later. These linguistic examples provide some evidence that we conceptualise *Marriage* and *Bank account* as kinds of *containers*.

The claim that these five concepts are indeed instantiating CONTAINER is supported by the behaviour of these concepts in *similes*. The first column ('target domain') of Table 1 contains our examples. The second column ('source domain') contains various concepts of physical containers which highlight some possible features of containers: e.g., a container may leak, be hard to get out of, or have a flexible boundary. Let us consider the similes *X is like a Y* that are the result of randomly choosing an element *X* from the first row and combining it with a random element *Y* from the second column. For example, 'The universe is like a treasure chest', 'Their marriage is like a prison', 'My bank account is like a leaky pot'. Note that all of the resulting similes are meaningful. Some of them will intuitively have more appeal than others, which may only be meaningful within a particular context.[18]

The fact that Table 1 can be used to randomly produce similes is linguistically interesting, because the target concepts vary significantly. The concepts *space ship*, *marriage* and *North Korea* seem to have nothing in common. Therefore, the fact that they can all be compared meaningfully to the same concepts needs an explanation. The puzzle is solved if we assume all concepts in the first column share the underlying image schema CONTAINER. For this reason they can be blended with the container concepts from the second column. In each simile we project some feature of the container in the source domain (second column) via analogical transfer onto the container aspect of the target domain (first column). Thus, Table 1 provides evidence that image schemas can help us to identify or (construe) shared structure between concepts.

The shared structure between concepts can be utilised in conceptual blending. For example, we can conceptually blend the concepts *universe* and *balloon* to a *balloon-universe*, that is a universe that continuously increases its size and expands. This concept is already lexicalised as *expanding universe* in English. Blending *space ship* with *prison* could lead to various interesting concepts: e.g., to a space ship that is used as a prison – a kind of space age version of the British prison hulks of the 19th century.

It is also possible to attempt to blend two different concepts from the first column from Table 1. However, since these concepts contain more prominent aspects than CONTAINER, these blends may not involve the CONTAINER as shared structure. E.g., a in a blend of *Space Ship* and *North Korea* probably other aspects of the concept of North Korea would be more dominant. E. g., a *North Korean Space Ship* may be, trivially, a space ship built in North Korea or a space ship with a dictatorial captain and a malnourished crew. Only by providing some additional context one

[18] For example, 'This space ship is like a bottomless pit' may sound odd in isolation, but in the context of 'I have already 20.000 containers in storage, and there is still empty cargo space' the simile works.

can prime the CONTAINER aspect of North Korea; e.g., in 'People inside North Korea do not learn anything about the rest of the world, from their perspective they live in the space ship North Korea, which is surrounded by an empty void.'

Let us consider two different examples from our list. A blend of *marriage* and *bank account* may yield the concept of a *marriage account*. This new concept could be used in sentences like the following: 'Marcus and Susie have just spent a long and happy holiday together, this was a big investment into their marriage account, it is now full of love' or 'Jim needs to watch the way he treats Jill, their marriage account is draining quickly and is nearly empty. She is probably going to leave him'. In this blend the *marriage account* is a container which contains feelings between the spouses instead of money. The blend inherits the domain from *marriage* (with the major difference that the spouses themselves are no longer inside the container). The main contribution of *bank account* to the blend is the ability to 'invest' and 'check the balance' of the content in the *marriage account*.

How something is conceptualised depends on the context. For example, surgeons may conceptualise people as containers of organs, blood, and various other anatomical entities, but in most contexts we do not conceptualise humans in this way. By choosing the appropriate context an image schema may be pushed from the background into the conceptual forefront. For example, in most contexts a *mother* is probably not conceptualised as a kind of container. However, in the appropriate contexts it is possible to generate similes for *mother* reusing the source domains from Table 1; e.g., 'The mother is pregnant with twins, she looks like a balloon' or 'The mother is like a prison for the unborn child'.

The examples that we have discussed in this section show how the CONTAINER image schema can be utilised as generic space in conceptual blending. In the next sections we present the formalisation of the blending of two of our examples, namely *space ship* and *mother*.

### 4.2 The mother ship example

Our thesis is that image schemas provide a useful heuristics for conceptual blending, because shared image schemas are good candidates for the generic space in the blending process.

The concepts *space ship* and *mother* share the CONTAINER schema. As a first step towards the formalisation of the blending process, we need to represent CONTAINER in some formal language.

For the sake of illustrating the basic ideas, we choose here a simplified representation in OWL (see Figure 4). Containers are defined as material objects that have a cavity as a proper part. A container contains an object if and only if the object is located in the cavity that is part of the container.[19]

As mentioned in Section 4.1, many concepts contain a rich structure. We do not attempt here to provide a full axiomatisation of *mother* or *space ship*, but just focus on some salient points for the sake of illustrating the blending process.

As discussed in 4.1, *mother* realises the CONTAINER schema, since mothers have a uterine cavity, which at some point in time contained some child. Further, *space ship* realises the CONTAINER schema since space ships may be used to transport goods and passengers. Of course, in

[19] This is a simplified view on CONTAINER. E.g., a more accurate formalisation of the CONTAINER schema would need to cover notions like moving into or out of the container.

```
Class: Container
    EquivalentTo: MaterialObject and has_proper_part some Cavity

ObjectProperty: contains
    SubPropertyChain: has_proper_part o is_location_of
    DisjointWith: has_proper_part
    Domain: Container
    Range: MaterialObject
```

Fig. 4: A (partial) representation of CONTAINER in OWL

almost any other aspect mothers and space ships are completely different; in Fig. 5 we only represent that mothers are female humans with children and that space ships are capable of space travel.

```
Class: Mother
    EquivalentTo: Female and Human and parent_of some (Small and Human)
    SubClassOf: has_proper_part some UterineCavity

Class: SpaceShip
    EquivalentTo: Vehicle and has_capability some Spacefaring
    SubClassOf: has_proper_part some CargoSpace
```

Fig. 5: Mothers and space ships

During the blending of *mother* and *space ship* into *mother ship* the CONTAINER schema structure of both input spaces is preserved (see Fig. 7). The uterine cavity and the cargo space are both mapped to the docking space. The *mother ship* inherits some features from both input spaces, while others are dropped. Obviously, a mother ship is a space travelling vessel. But like a mother, it is a 'parent' to some smaller entities of the same type. These smaller vessels can be contained within the mother ship, they may leave its hull (a process analogous to a birth) and are supported and under the authority of the larger vessel.[20]

To summarise, in our example we try to blend the input spaces of "Mother" and "Space ship". Instead of trying to utilise a syntactic approach like anti-unification to search for a base space, we recognise that both input spaces have cavities and, thus, are containers. Using the base space CONTAINER in the blending process yields a blended concept of "Mother ship". Here, the precise mappings from the base space axiomatisation of CONTAINER to the two input spaces regulate the various properties of the blended concept. Figure 6 illustrates this blend by populating the generic blending schema shown in Figure 3.

[20] To represent dynamic aspects like birth and vessels leaving a docking bay adequately, one needs a more expressive language than OWL.

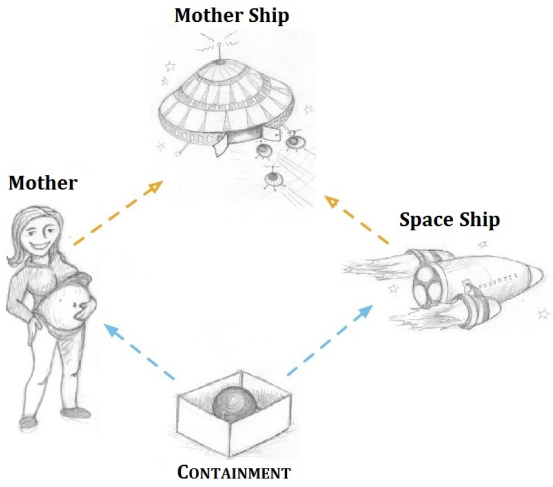



Fig. 6: The blending of mother ship

```
Class: MotherShip
    SubClassOf: Vehicle and has_capability some Spacefaring
    SubClassOf: has_proper_part DockingStation
    SubClassOf: parent_of some (Small and Vehicle)
```

Fig. 7: Mother ship

### 4.3 The satellite example

To further illustrate the role of image schemas in the construction of a newly blended concept, let us consider a second example. Assume we want to create a new concept by blending our mother ship with a moon. While this may not be astronomically completely correct, for the sake of this paper we consider a moon to be a celestial object that is part of some solar system, has a spheroidal shape, consists of rock, and orbits around a planet (see Figure 8). Of course, many people would associate additional information with the concept *moon*, but even if we consider only these aspects, there are different possibilities how we could blend the two concepts. E.g., a structure mapping approach would probably first try to identify the parthood relationship between the docking station and the mother ship on one hand with the parthood relationship between the moon and the solar system. This may lead to the concept of a Moon/DockingStation that is part of a SolarSystem/MotherShip – While not being wrong, it might not be a useful concept.

In contrast, if one utilises shared image schemas as heuristics for conceptual blending, it is quite natural to look at a very different place for blending opportunities. Since the mother ship is a kind of vehicle it has the capability to move stuff or people, which involves movement from some place to another along a path. A moon also moves along a path, namely it's orbit. This commonality we can utilise in the blending process. However, in this case the situation is not as straightforward as in Section 4.2, because the movements of the mother ship and the moon are quite different and do not instantiate the same image schema.

```
Class: Moon
    EquivalentTo: CelestialObject and participates_in some
        (OrbitalMovement and revolves_around some (Planet or DwarfPlanet))
    SubClassOf: consists_of Rock
    SubClassOf: part_of some SolarSystem
    SubClassOf: has_shape some Spheroid
```

Fig. 8: Moon

```
Class: Vehicle
    SubClassOf: has_capability some SourceToGoalMovementCapability
    SubClassOf: has_capability some TransportationCapability

Class: SourceToGoalMovement
    EquvialentTo: MovementProcess and
        (has_participant some MovingEntity) and
        (follows exactly 1 (Path and (has_source exactly 1 Location)
            and  (has_destination exactly 1 Location)))
    EquivalentTo: executes some SourceToGoalMovementCapability

Class: OrbitalMovement
    EquvialentTo: MovementProcess and
        (follows exactly 1 (LoopingPath and
                (revolves_around some owl:Thing)))

Class: LoopingPath
    EquvialentTo: Path and Looping
    SubClassOf: has_source only owl:Nothing
    SubClassOf: has_destination only owl:Nothing
```

Fig. 9: Movement

When the mother ship (or any vessel) executes its capability in some movement process, the vehicle starts at some location of origin and moves along a path until it reaches its goal, where it stops. Thus, the image schema is THING MOVES ON PATH FROM SOURCE TO GOAL. The orbital movement of the moon also follows some path. However, the orbital movement does not have a source or a goal; it is characterised by a focal point, which the orbiting object revolves around. Hence, the formal representation of both kinds of movement looks quite differently (see Figure 9 for a representation in OWL).

As discussed in Section 2.2, spatial schemas, can be enriched by adding additional spatial primitives; the spatial schemas instantiated by the movement of a vessel and of a moon, respectively, are different (and mutually exclusive) refinements of THING MOVES ON PATH. For the purpose of blending the important lesson is that image schemas do not exist in isolation, but they are members of *families of image schemas*. The members of these image schema families are variants of some root conceptualisation (e.g., movement) and can be partially ordered by their strength (see Section 5).

We can utilise this observation as a heuristic for conceptual blending: if two concepts involve two different image schemas, which are within the same image schema family, then a good candidate for the base space for blending both concepts is the least general member of the image schema family, which generalises the image schemas in the input spaces. In the case of our example, this is MOVEMENT ON A PATH. Further, the blended concept probably should include only one member of the image schema family. In our example, we can create a new concept that inherits the salient features of the mother ship, but replaces its ability to travel from one place to another by some orbital movement. The resulting theory describes a space station, which orbits around a planet or dwarf planet. Alternatively, we can think of a moon-like concept that is turned into a spacefaring vehicle. This is a kind of 'moon ship', that is a moon that has the capability to move from a location of origin along a path to a destination (see Figure 10).

This example illustrates how the use of image schemas can provide heuristics for (i) identifying suitable base spaces and (ii) selecting features during running interesting blends – even if the input domains do not share exact structure. However, it raises the question how we should represent image schema families formally. This is a question we will discuss in the next section.

```
Class: SpaceStation
    SubClassOf: has_capability some Spacefaring
    SubClassOf: has_proper_part DockingStation
    SubClassOf: parent_of some (Small and Vessel)
    SubClassOf: participates_in some
        (OrbitalMovement and revolves_around some (Planet or DwarfPlanet))

Class: MoonShip
    SubClassOf: Vehicle
    SubClassOf: consists_of Rock
    SubClassOf: has_shape some Spheroid
```

Fig. 10: Space station and moon ship

## 5 Image Schema Families as Graphs of Theories

In the previous section, we suggested that image schemas are members of families, which are partially ordered by generality. Formally, we can represent such a family as a graph of theories in DOL.[21] In this section, we discuss this approach in some more detail. On a technical level, our proposal for capturing image schemas as interrelated families of (heterogeneous) theories is quite similar to the ideas underlying the first-order ontology repository COLORE[22] [21].

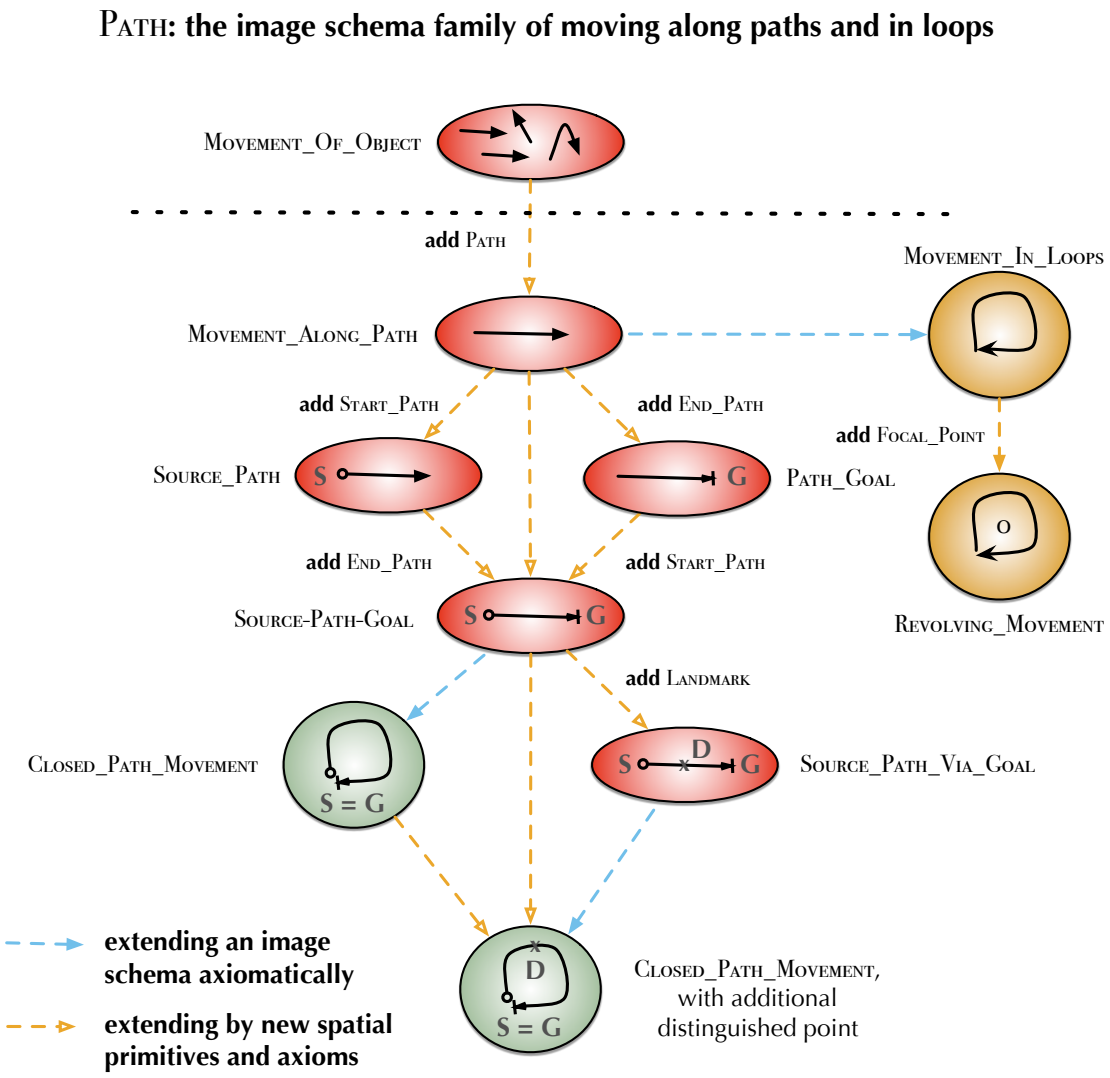



Fig. 11: A portion of the family of image schemas related to path following shown as DOL graph.

[21] These graphs are diagrams in the sense of category theory.
[22] See `http://stl.mie.utoronto.ca/colore/`

In our blend of *mother ship* with *moon* we considered two variants of MOVEMENT ON A PATH. Figure 11 shows a selection of some other members of the same image schema family.[23] One way MOVEMENT ON A PATH can be specialised is as MOVEMENT ON A LOOPING PATH. Note that this change does not involve adding a new spatial primitive, but just an additional characteristic of the path. The resulting image schema can be further refined by adding the notion of a *focal point*, which the path revolves around – this leads to the notion of orbiting. Alternatively, we may change MOVEMENT ON A PATH by adding distinguished points; e.g., the source, the target, or both.

The latter image schema may be further specialised by identifying the source and the target. In this case the path is closed in the sense that any object which follows the path will end up at the location at where it started its movement (the source). The difference between a closed path and a looping path is that the closed path has a start and an end (e.g., a race on a circular track), while the looping path has neither (like an orbit). It is possible to further refine the schema by adding more designated points or other related spatial primitives.

The particular image schema family sketched in Figure 11 is organised primarily via adding new spatial primitives to the participating image schemas and/or by refining an image schema's properties (extending the axiomatisation).[24] In general, different sets of criteria may be used depending, for example, on the context of usage, thereby putting particular image schemas (say, REVOLVE_AROUND) into a variety of families. Apart from a selection of spatial primitives, other dimensions might be deemed relevant for defining a particular family, such as their role in the developmental process.

In [5], eight closely related kinds of CONTAINERs were identified as being distinguishable within natural language corpora, illustrated in Figure 1 and discussed above. Hence, the selection criteria for grouping together these particular forms of containment are not simply driven by a selection of spatial primitives. Although [5] does not explicitly formalise the structural relationships between the different notions of containment, they are clearly present. Thus, their work provides an empirically well-motivated example of an image schema family.

To implement computationally the idea of using image schemas as generic spaces, two independent algorithmic problems have to be solved. Namely (1) the **Recognition Problem**: to identify an image-schematic theory within an input theory, and (2) the **Generalisation Problem**: to find the most specific image schema common to both inputs.

To address the recognition problem, suppose a lattice $\mathfrak{F}$ encoding an image schema family is fixed. We here assume for simplicity that elements of $\mathfrak{F}$ will be logical theories in a fixed formal logic, say first-order logic.[25] Given an input theory $O_1$ and $\mathfrak{F}$, solving the recognition problem means finding a member $f \in \mathfrak{F}$ that can be *interpreted* in $O_1$, i.e. such that we find a renaming $\sigma$ of the symbols in $f$ (called a signature morphism) and such that $O_1 \models \sigma(f)$ (also written $O_1 \models_\sigma$

[23] A disclaimer: in the following we will describe an approach to represent the connections between image schemas, belonging to the same family according to certain criteria. To illustrate some technical points, we will just postulate the existence of several image schemas and their connections. However, we here do not intend to make any claims regarding their empirical existence and/or their cognitive role in development.

[24] In [26, 27] we present a more complete description of the image schema family of 'path following' and the corresponding formal methodology.

[25] Note that none of the ideas presented here depend on a particular, fixed logic. Indeed, heterogeneous logical specification is central to formal blending approaches, see [36].

$f$).[26] Note that this is a more general statement than claiming the inclusion of the axioms of $f$ (modulo renaming) in $O_1$ (the trivial inclusion interpretation) since establishing the entailment of the sentences in $\sigma(f)$ from $O_1$ might indeed be involved.

Computational support for automatic theory-interpretation search in first-order logic is investigated in [51], and a prototypical system was developed and tested as an add-on to the *Heterogeneous Tool Set*, HETS [49]. Experiments carried out in [40, 52] showed that this works particularly well with more complex axiomatisations in first-order logic, rather than with simple taxonomies expressed in OWL, because in the latter case too little syntactic structure is available to control the combinatorial explosion of the search task. From the point of view of interpreting image schemas into non-trivial axiomatised concepts, we may see this as an encouraging fact, as image schemas are, despite their foundational nature, complex objects to axiomatise.

Once the recognition problem has been solved in principle, the given lattice structure of the image schema family $\mathfrak{F}$ gives us a very simple handle on the generalisation problem. Namely, given two input spaces $O_1$, $O_2$, and two image schemas $f_1$, $f_2$ from the same family $\mathfrak{F}$ (say, 'containment') such that $O_1 \models_{\sigma_1} f_1$ and $O_2 \models_{\sigma_2} f_2$, compute the most specific generalisation $G \in \mathfrak{F}$ of $f_1$ and $f_2$, i.e. their least upper bound in $\mathfrak{F}$. Since the signature of $G$ will be included in both signatures of $f_1$ and $f_2$, we obtain that $O_1 \models_{\sigma_1} G$ and $O_2 \models_{\sigma_2} G$. $G \in \mathfrak{F}$ is therefore an image schema common to both input spaces and can be used as generic space.

In order to implement this idea, a sufficiently comprehensive library of formalised image schema theories has to be made available for access by a blending engine. The first such library for the case of 'path following' is developed in [26].

## 6 Conclusion

In this paper we suggest that image schemas can provide useful heuristics for computational blending of concepts. They can serve as a driving force to identify or define the generic space and its mappings to the input spaces: because image schemas are building blocks of the concepts in the input spaces, we can generate generic spaces by identifying image schemas that are shared across both input spaces. Here, in particular, the idea of organising image schemas into lattice-like structures allows us to identify these shared structures even when the image schemas found in the input concepts do not precisely coincide, but are closely related within a common family.

Our hypothesis is that, compared to syntax-driven approaches (e.g., structure mapping), this approach allows us to identify more cognitively relevant generic spaces and, thus, for cognitively more interesting conceptual blends. Further, we conjecture that many interesting conceptual blends rely on generalisations of image schemas found in the input concepts and organised into well-motivated families.

To test this hypothesis, we intend to continue to develop and expand our image schema library, which formalises image schemas, their families, and interconnections between families, as DOL networks. This also includes more heterogeneous image schema families, where formal languages other than description logics are involved (CONTAINER of [5] is an example). While some formal-

[26] In more detail: a theory interpretation $\sigma$ is a signature morphism renaming the symbols of the image schema theory $f$ and induces a corresponding sentence translation map, also written $\sigma$, such that the translated sentences of $f$, written $\sigma(f)$, are logically entailed by $O_1$.

isation of image schemas can be found in the literature on conceptual blending and common sense reasoning [18, 34, 47], the area of systematically formalising and ontologically structuring image schemas is a largely unexplored ground.

The image schema library will allow us to use the tools for computational concept invention that are developed in the COINVENT Project[27], improve the tool's heuristics, and at the same time test our hypotheses.

We are planning to compare the quality of the blended concepts that are generated based on our proposed heuristics with other blended concepts by using human judges. This way we will evaluate whether our hypotheses are correct and image schemas indeed provide a useful tool for computational concept invention.

[27] http://coinvent-project.eu/

# Bibliography


[1] Robert St. Amant, Clayton T. Morrison, Yu-Han Chang, Wei Mu, Paul R. Cohen, and Carole Beal. An image schema language. In *Proc. of the 7th International Conference on Cognitive Modeling (ICCM)*, pages 292–297, Trieste, Italy, April 2006.

[2] Lawrence W. Barsalou. Grounded cognition. *Annual review of psychology*, 59:617–645, 2008.

[3] John Bateman, Joana Hois, Robert Ross, and Thora Tenbrink. A Linguistic Ontology of Space for Natural Language Processing. *Artificial Intelligence*, 174(14):1027–1071, 2010.

[4] Vladimir Bazjanac, James Forester, Philip Haves, Darko Sucic, and Peng Xu. HVAC component data modeling using industry foundation classes. Proc. of the 5th International Conference on System Simulation in Buildings, Liege, Belgium, 2002.

[5] Brandon Bennett and Claudia Cialone. Corpus Guided Sense Cluster Analysis: a methodology for ontology development (with examples from the spatial domain). In Pawel Garbacz and Oliver Kutz, editors, *Proc. of the 8th International Conference on Formal Ontology in Information Systems (FOIS)*, volume 267 of *Frontiers in Artificial Intelligence and Applications*, pages 213–226, Rio de Janeiro, Brazil, September 2014. IOS Press.

[6] Tarek R. Besold, Marco Schorlemmer, and Alan Smaill, editors. *Computational Creativity Research: Towards Creative Machines*, volume 7 of *Atlantis Thinking Machines*. Atlantis Press, 2015.

[7] Mehul Bhatt, Joana Hois, and Oliver Kutz. Ontological modelling of form and function in architectural design. *Applied Ontology*, 7(3):233–267, 2012.

[8] Margaret A. Boden. Computer models of creativity. *AI Magazine*, 30(3):23–34, 2009.

[9] Robert DiSalle. Space and time: Inertial frames. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2009 edition, 2009.

[10] Martin Doerr. The cidoc conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI magazine*, 24(3):75–92, 2003.

[11] Max J. Egenhofer and David M. Mark. Naive geography. In Andrew U. Frank and Werner Kuhn, editors, *Spatial Information Theory: a theoretical basis for GIS*, volume 988 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1995.

[12] Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

[13] Vittorio Gallese and George Lakoff. The Brain's concepts: the role of the Sensory-motor system in conceptual knowledge. *Cognitive neuropsychology*, 22(3):455–79, 2005.

[14] Peter Gärdenfors. *Conceptual Spaces - The Geometry of Thought*. Bradford Books. MIT Press, 2000.

[15] Dedre Gentner. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[16] 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.

[17] Joseph. A. Goguen. Semiotic Morphisms, Representations and Blending for Interface Design. In *Proc. of the AMAST Workshop on Algebraic Methods in Language Processing*, pages 1–15, Verona, Italy, August 2003. AMAST Press.

[18] Joseph A. Goguen and D. Fox Harrell. Style: A Computational and Conceptual Blending-Based Approach. In Shlomo Argamon and Shlomo Dubnov, editors, *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*, pages 147–170. Springer, Berlin, 2010.

[19] Joseph A. Goguen and Grant Malcolm. *Algebraic Semantics of Imperative Programs*. MIT, 1996.

[20] Joseph E. Grady. Cognitive mechanisms of conceptual integration. *Cognitive Linguistics*, 11(3-4):335–345, 2001.

[21] Michael Grüninger, Torsten Hahmann, Ali Hashemi, Darren Ong, and Atalay Ozgovde. Modular First-Order Ontologies Via Repositories. *Applied Ontology*, 7(2):169–209, 2012.

[22] Markus Guhe, Alison Pease, Alan Smaill, Maricarmen Martínez, Martin Schmidt, Helmar Gust, Kai-Uwe Kühnberger, and Ulf Krumnack. A computational account of conceptual blending in basic mathematics. *Cognitive Systems Research*, 12(3–4):249–265, 2011.

[23] Beate Hampe. Image schemas in cognitive linguistics: Introduction. In Beate Hampe and Joseph E Grady, editors, *From perception to meaning: Image schemas in cognitive linguistics*, volume 29, pages 1–14. Walter de Gruyter, 2005.

[24] Beate Hampe and Joseph E. Grady, editors. *From perception to meaning: Image schemas in cognitive linguistics*, volume 29 of *Cognitive Linguistics Research*. Walter de Gruyter, 2005.

[25] Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus. On the cognitive and logical role of image schemas in computational conceptual blending. In Antonio Lieto and Daniele Radicioni, editors, *Proc. of the 2nd International Workshop on Artificial Intelligence and Cognition (AIC-2014)*, volume Volume 1315 of *CEUR-WS*, Torino, Italy, November 2014.

[26] Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus. Choosing the Right Path: Image Schema Theory as a Foundation for Concept Invention. *Journal of Artificial General Intelligence*, 6(1):21–54, 2015.

[27] Maria M. Hedblom, Oliver Kutz, and Fabian Neuhaus. Image Schemas as Families of Theories. In Tarek R. Besold, Kai-Uwe Kühnberger, Marco Schorlemmer, and Alan Smaill, editors, *Proc. of the Workshop "Computational Creativity, Concept Invention, and General Intelligence" (C3GI-15)*, volume 02-2015 of *Publications of the Institute of Cognitive Science*, Osnabrück, 2015. Series Editor: Kühnberger, K.-U. and König, P. and Walter, S.

[28] Joana Hois, Mehul Bhatt, and Oliver Kutz. Modular Ontologies for Architectural Design. In *Proc. of FOMI-09*, volume 198 of *Frontiers in Artificial Intelligence and Applications*, Vicenza, Italy, 2009. IOS Press.

[29] Joana Hois, Oliver Kutz, Till Mossakowski, and John Bateman. Towards Ontological Blending. In *Proc. of the The 14th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA-2010)*, Varna, Bulgaria, September 2010.

[30] Megan Johanson and Anna Papafragou. What does children's spatial language reveal about spatial concepts? Evidence from the use of containment expressions. *Cognitive science*, 38(5):881–910, 2014.

[31] Mark Johnson. *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*. The University of Chicago Press, Chicago and London, 1987.

[32] Raymond W. Gibbs Jr. Making good psychology out of blending theory. *Cognitive Linguistics*, 11(3-4):347–358, 2001.

[33] Bernd Krieg-Brückner, Serge Autexier, Martin Rink, and Sidoine Ghomsi Nokam. Formal Modelling for Cooking Assistance. In Rocco De Nicola and Rolf Hennicker, editors, *Software, Services, and Systems*, volume 8950 of *Lecture Notes in Computer Science*, pages 355–376. Springer International Publishing, 2015.

[34] Werner Kuhn. Modeling the Semantics of Geographic Categories through Conceptual Integration. In *Proc. of GIScience 2002*, pages 108–118, Boulder, US, Sep 2002. Springer.

[35] Werner Kuhn. An Image-Schematic Account of Spatial Categories. In Stephan Winter, Matt Duckham, Lars Kulik, and Ben Kuipers, editors, *Spatial Information Theory*, volume 4736 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2007.

[36] 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.

[37] 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.

[38] 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?'.

[39] Oliver Kutz, Fabian Neuhaus, Till Mossakowski, and Mihai Codescu. Blending in the Hub—Towards a collaborative concept invention platform. In *Proc. of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia, June 2014.

[40] Oliver Kutz and Immanuel Normann. Context Discovery via Theory Interpretation. In *Proc. of the IJCAI Workshop on Automated Reasoning about Context and Ontology Evolution, ARCOE-09*, Pasadena, California, June 2009.

[41] George Lakoff and Rafael E. Núñez. *Where Mathematics Comes From*. Basic Books, 2000.

[42] Douglas B. Lenat, Mayank Prakash, and Mary Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI magazine*, 6(4):65–85, 1985.

[43] Jean M. Mandler. How to build a baby: II. Conceptual primitives. *Psychological review*, 99(4):587–604, 1992.

[44] Jean M. Mandler. *The Foundations of Mind: Origins of Conceptual Thought: Origins of Conceptual Though*. Oxford University Press, New York, 2004.

[45] Jean M. Mandler. On the Birth and Growth of Concepts. *Philosophical Psychology*, 21(2):207–230, 2008.

[46] Jean M. Mandler and Cristóbal Pagán Cánovas. On defining image schemas. *Language and Cognition*, 6(4):510–532, December 2014.

[47] Leora Morgenstern. Mid-Sized Axiomatizations of Commonsense Problems: A Case Study in Egg Cracking. *Studia Logica*, 67:333–384, 2001.

[48] 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.

[49] Till Mossakowski, Christian Maeder, and Klaus Lüttich. The Heterogeneous Tool Set. In Orna Grumberg and Michael Huth, editors, *TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer-Verlag Heidelberg, 2007.

[50] Fabian Neuhaus, Oliver Kutz, Mihai Codescu, and Till Mossakowski. Fabricating Monsters is Hard - Towards the Automation of Conceptual Blending. In *Proc. of Computational Cre-*

*ativity, Concept Invention, and General Intelligence (C3GI-14)*, volume 1-2014, pages 2–5, Prague, 2014. Publications of the Institute of Cognitive Science, Osnabrück.

[51] Immanuel Normann. *Automated Theory Interpretation*. PhD thesis, Department of Computer Science, Jacobs University, Bremen, 2008.

[52] Immanuel Normann and Oliver Kutz. Ontology Correspondence via Theory Interpretation. In *Workshop on Matching and Meaning (AISB-09)*, Edinburgh, UK, 2009.

[53] Todd Oakley. Image schema. In Dirk Geeraerts and Hubert Cuyckens, editors, *The Oxford Handbook of Cognitive Linguistics*, pages 214–235. Oxford University Press, 2007.

[54] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *Proc. of the 3rd International Conference on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. Morgan Kaufmann, Los Altos, 1992.

[55] Stephen K. Reed and Adam Pease. A framework for constructing cognition ontologies using WordNet, FrameNet, and SUMO. *Cognitive Systems Research*, 33:122–144, 2015.

[56] Stephen L. Reed, Douglas B. Lenat, et al. Mapping ontologies into cyc. In *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*, pages 1–6, 2002.

[57] Tim Rohrer. Image schemata in the brain. In Beate Hampe and Joseph E Grady, editors, *From perception to meaning: Image schemas in cognitive linguistics*, volume 29 of *Cognitive Linguistics Research*, pages 165–196. Walter de Gruyter, 2005.

[58] Mark A. Runco and Garrett J. Jaeger. The standard definition of creativity. *Creativity Research Journal*, 24(1):92–96, 2012.

[59] Martin Schmidt, Ulf Krumnack, Helmar Gust, and Kai-Uwe Kühnberger. *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*. Springer, Berlin Heidelberg, 2014.

[60] Marco Schorlemmer, Alan Smaill, Kai-Uwe Kühnberger, Oliver Kutz, Simon Colton, Emilios Cambouropoulos, and Alison Pease. COINVENT: Towards a Computational Concept Invention Theory. In *Proc. of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia, June 2014.

[61] Barry Smith, Werner Ceusters, Bert Klagges, Jacob Köhler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan L Rector, and Cornelius Rosse. Relations in biomedical ontologies. *Genome biology*, 6(5):R46, 2005.

[62] Mark Turner. *The Origin of Ideas: Blending, Creativity, and the Human Spark*. Oxford University Press, 2014.

[63] Tony Veale. The analogical thesaurus. In John Riedl and Randy Hill, editors, *Proc. of the 15th Innovative Applications of Artificial Intelligence Conference*, pages 137–142, Acapulco, Mexico, August 2003. AAAI Press.

[64] Tony Veale. From conceptual âmash-upsâ to âbad-assâ blends: A robust computational model of conceptual blending. In Mary Lou Maher, Kristian Hammond, Alison Pease, Rafael PÃ©rez y PÃ©rez, Dan Ventura, and Geraint Wiggins, editors, *Proc. of the 3rd International Conference on Computational Creativity*, pages 1–8, Dublin, Ireland, May 2012.

[65] Fan-Pei Gloria Yang, Kailyn Bradley, Madiha Huq, Dai-Lin Wu, and Daniel C. Krawczyk. Contextual effects on conceptual blending in metaphors: An event-related potential study. *Journal of Neurolinguistics*, 26:312–326, 2012.

# Upward Refinement Operators for Conceptual Blending in the Description Logic $\mathcal{EL}^{++}$

Roberto Confalonieri, Manfred Eppe, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, and Enric Plaza

## 1 Introduction

The upward refinement—or generalisation—of concepts plays a crucial role in creative cognitive processes for analogical reasoning and concept invention. In this work we focus on its role in *conceptual blending* [21], where one combines two input concepts to invent a new one. A problem in blending is that the combination of two concepts may generate an unsatisfiable one due to contradiction, or may not satisfy certain properties. However, by generalising input concepts, we can remove inconsistencies to find a novel and useful combination of the input concepts. For instance, a 'red French sedan' and a 'blue German minivan' can be blended to a 'red German sedan' by generalising the first concept to a 'red European sedan' and the second one to a 'coloured German car'. The least general generalisation of our input concepts—a 'coloured European car'—serves as an upper bound of the generalisation space to be explored, and, in a certain sense, plays the role of the so called *generic space* in conceptual blending, which encodes the shared structure of both concepts.

This paper addresses the formalisation and implementation of such a generalisation process in the context of the description logic $\mathcal{EL}^{++}$ [5, 7]. The choice of $\mathcal{EL}^{++}$ as the knowledge representation language for a computational interpretation of the cognitive theory of conceptual blending is motivated by several reasons. First, $\mathcal{EL}^{++}$ is the underpinning logic of the OWL 2 EL Profile[1], a recommendation of the W3C, and, therefore, a well-understood and commonly used knowledge representation formalism. Second, $\mathcal{EL}^{++}$ offers a good tradeoff between expressiveness and efficiency of reasoning and is considered to be sufficiently expressive to model large real-world ontologies, specially in the bio-medical domains [15, 41]. Finally, subsumption of concepts w.r.t. an $\mathcal{EL}^{++}$ TBox is computable in polynomial time [5], and therefore of special interest for a tractable real-world implementation of conceptual blending. Indeed, a nontrivial problem of conceptual blending is that there usually exists a considerable number of possible combinations for the blend creation that are inconsistent or otherwise not interesting (see e.g., [20]). These combinations need to be evaluated. Our $\mathcal{EL}^{++}$-based formalisation of conceptual blending suggests that these combinations, leading to the blends, can be evaluated against the entailment of some properties, modelled as ontology consequence requirements. The nice computational properties of $\mathcal{EL}^{++}$ facilitate this kind of evaluation since entailment in $\mathcal{EL}^{++}$ is not computationally hard.

The generalisation of $\mathcal{EL}^{++}$ concepts has been studied both in the Description Logic (DL) and in the Inductive Logic Programming (ILP) literature, although from different perspectives. Whilst approaches in DL focus on formalising the computation of a least general generalisation (LGG) (also known as least common subsumer) among different concepts as a non-standard reasoning task [2, 6, 44], approaches in ILP are concerned on learning DL descriptions from examples [33].

[1] `http://www.w3.org/TR/owl2-profiles/`, accessed 26/11/2015

In both cases, however, finding a LGG is a challenging task. Its computability depends on the type of DL adopted and on the assumptions made over the structure of concept definitions.

Our work relates to these approaches, but our main motivation for generalising DL concepts is intrinsically different. Although we do need to be aware of what properties are shared by the concepts in order to blend them, it is not necessary (though desirable) to find a *generic space* that is also a LGG. A minimally specific common subsumer w.r.t. the subconcepts that can be built using the axioms in a Tbox will suffice. With this objective in mind, we propose an upward refinement operator for generalising $\mathcal{EL}^{++}$ concepts which is inductively defined over the structure of concept descriptions. We discuss some of the properties typically used to characterise refinement operators; namely, local finiteness, properness and completeness [30].[2] Particularly, our operator is locally finite and proper, but it is not complete. As a consequence, it cannot generate all the possible generalisations of an $\mathcal{EL}^{++}$ concept. As we shall discuss, we sacrifice completeness for finiteness (since we do not need to compute a LGG, strictly speaking), but we need the applications of the operator to always terminate at each refinement step.

As far as the implementation of the operator is concerned, we state the problem of finding a generic space of $\mathcal{EL}^{++}$ concepts as a planning problem. This involves finding a sequence of generalisations with conditional effects to reach the generic space. This is natural, because modifying $\mathcal{EL}^{++}$ concepts underlies certain conditional rules. These rules are ultimately defined through the *upward cover set* which is generated within the generalisation operator definitions (see Definitions 6 and 7). It is well-known that planning problems are inherently non-monotonic because of the inertia assumption. That is, one assumes that world properties, in this case parts of $\mathcal{EL}^{++}$ concept descriptions, persist unless there is evidence that they changed. The 'unless there is evidence' condition implies the use of Negation as Failure (NaF). To this end, we adopt the nonmonotonic logic programming paradigm of Answer Set Programming (ASP) [24].[3]

To implement the the upward refinement operator and generic space search, we employ the incremental solving capabilities of *clingo* [22], an advanced ASP solver, to find a generic space among two $\mathcal{EL}^{++}$ input concepts. The ASP search is embedded in an amalgam-based process that models conceptual blending. We present a conceptual blending algorithm that uses the generalisations found by the ASP-based search process to create new blended concepts. New concepts are evaluated by means of ontology consequence requirements and a heuristics function. Throughout the paper, we use an example in the domain of computer icon design.

This paper is an extended and revised version of [14]. It now contains a formal definition and analysis of the refinement operator properties (Propositions 1-3 and Theorem 1), an extension of the operator definition to deal with infinite chain of generalisations, the complete implementation of the operator in ASP, and a blending algorithm.

The remainder of this paper is organised as follows: Section 2 provides the background knowledge to make this paper self-contained. Section 3 describes how conceptual blending can be used

[2] Briefly, a refinement operator is said to be locally finite when it generates a finite set of refinements at each step; proper, when its refinements are not equivalent to the original concept, and complete, when it produces all possible refinements of a given concept. These property are formally presented in Section 2.2.

[3] The planning problem could also have been encoded in SAT. There are many approaches to realize NaF and nonmonotonicity for SAT, with circumscription [35] probably being the most prominent method. In this sense, the computational complexity is equivalent with the one of ASP. However, since NaF is already an inherent part of ASP, we found the use of ASP more straightforward. This is also in-line with recent trends in Commonsense Reasoning about Action and Change, where ASP is commonly used to solve planning problems (see e.g., [17, 18, 31, 34]).

to design new computer icons modeled in $\mathcal{EL}^{++}$. Section 4 proposes the formalisation of a refinement operator for generalising $\mathcal{EL}^{++}$ concepts. In Section 5, the implementation of the operator and the ASP incremental encoding, which models the generic space search, are presented. Section 6 describes an algorithm for conceptual blending. Section 7 outlines several works that relate to ours from different perspectives. Finally, Section 8 concludes the paper and envisions some future work.

## 2 Background

In this section we introduce the basic notions that will be used throughout the paper. After presenting the $\mathcal{EL}^{++}$ description logic, we introduce refinement operators. Then, we provide the definition of *amalgams* that provides a computational chacterisation of conceptual blending. We conclude the background with an overview of Answer Set Programming (ASP) and the incremental solving capabilities of *clingo*.

### 2.1 The Description Logic $\mathcal{EL}^{++}$

In DLs, concept and role descriptions are defined inductively by means of concept and role constructors over a finite set $N_C$ of concept names, a finite set $N_R$ of role names, and (possibly) a finite set $N_I$ of individual names. As is common practice, we shall write $A$, $B$ for concept names, $C$, $D$ for concept descriptions, $r$, $s$ for role names, and $a$, $b$, for individual names.

The semantics of concept and role descriptions is defined in terms of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty domain and $\cdot^{\mathcal{I}}$ is an interpretation function assigning a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each concept name $A \in N_C$, a set $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name $r \in N_r$, and an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual name $a \in N_I$, which is extended to general concept and role descriptions. The upper part of Table 1 shows the constructors of the description logic $\mathcal{EL}^{++}$ that are relevant for this paper, together with their interpretation. For a complete presentation of $\mathcal{EL}^{++}$ we refer to [5, 7]. A knowledge base usually consists of a finite set $\mathcal{T}$ of terminological axioms, called TBox, which contains intensional knowledge defining the main notions relevant to the domain of discourse; and a finite set $\mathcal{A}$ of assertional axioms, called ABox, which contains extensional knowledge about individual objects of the domain. In this paper, we focus only on terminological axioms of the form $C \sqsubseteq D$, i.e. general concept inclusions (GCIs), and $r_1 \circ \cdots \circ r_n \sqsubseteq r$, i.e. role inclusions (RIs), as well as axioms specifying domain and range restrictions for roles. The lower part of Table 1 shows the form of these axioms, together with the condition for these to be satisfied by an interpretation $\mathcal{I}$. By $\mathcal{L}(\mathcal{T})$ we refer to the set of all $\mathcal{EL}^{++}$ concept descriptions we can form with the concept and role names occurring in $\mathcal{T}$.

RIs allow one to specify role hierarchies ($r \sqsubseteq s$) and role transitivity ($r \circ r \sqsubseteq r$). The bottom concept $\bot$, in combination with GCIs, allows one to express disjointness of concept descriptions, e.g., $C \sqcap D \sqsubseteq \bot$ tells that $C$ and $D$ are disjoint. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies all axioms in $\mathcal{T}$. The basic reasoning task in $\mathcal{EL}^{++}$ is subsumption. Given a TBox $\mathcal{T}$ and two concept descriptions $C$ and $D$, we say that $C$ is (strictly) subsumed by $D$ w.r.t. $\mathcal{T}$, denoted as $C \sqsubseteq_{\mathcal{T}} D$ ($C \sqsubset_{\mathcal{T}} D$), iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $C^{\mathcal{I}} \neq D^{\mathcal{I}}$) for every model $\mathcal{I}$ of $\mathcal{T}$. We write $C \equiv_{\mathcal{T}} D$ as an abbreviation for $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. Analogously, given two roles $r, s \in N_r$, we say that $r$ is (strictly) subsumed by $s$ w.r.t. $\mathcal{T}$, denoted as $r \sqsubseteq_{\mathcal{T}} s$ ($r \sqsubset_{\mathcal{T}} s$), iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ ($r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ and $r^{\mathcal{I}} \neq s^{\mathcal{I}}$) for every model $\mathcal{I}$ of $\mathcal{T}$.

| concept description | interpretation |
|---|---|
| $A$ | $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| $\top$ | $\Delta^{\mathcal{I}}$ |
| $\bot$ | $\emptyset$ |
| $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| $\exists r.C$ | $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ |

| axiom | satisfaction |
|---|---|
| $C \sqsubseteq D$ | $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ |
| $C \equiv D$ | $C^{\mathcal{I}} = D^{\mathcal{I}}$ |
| $r_1 \circ \cdots \circ r_n \sqsubseteq r$ | $r_1^{\mathcal{I}}\ ;\ \cdots\ ;\ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$ |
| $domain(r) \sqsubseteq C$ | $r^{\mathcal{I}} \subseteq C^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| $range(r) \sqsubseteq C$ | $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C^{\mathcal{I}}$ |

Table 1: Syntax and semantics of some $\mathcal{EL}^{++}$ contructors and axioms. (Note: ';' is the usual composition operator in relation algebra.)

### 2.2 Refinement Operators

Refinement operators are a well known notion in Inductive Logic Programming where they are used to structure a search process for learning concepts from examples. In this setting, two types of refinement operators exist: specialisation (or downward) refinement operators and generalisation (or upward) refinement operators. While the former constructs specialisations of hypotheses, the latter contructs generalisations.

Generally speaking, refinement operators are defined over quasi-ordered sets. A quasi-ordered set is a pair $\langle \mathcal{S}, \preceq \rangle$ where $\mathcal{S}$ is a set and $\preceq$ is a binary relation among elements of $\mathcal{S}$ that is reflexive ($a \preceq a$) and transitive (if $a \preceq b$ and $b \preceq c$ then $a \preceq c$). If $a \preceq b$, we say that $b$ is more general than $a$, and if also $b \preceq a$ we say that $a$ and $b$ are equivalent. A generalisation refinement operator is defined as follows.[4]

**Definition 1.** *A generalisation refinement operator $\gamma$ over a quasi-ordered set $\langle \mathcal{S}, \preceq \rangle$ is a set-valued function such that $\forall a \in \mathcal{S} : \gamma(a) \subseteq \{b \in \mathcal{S} \mid a \preceq b\}$.*

A refinement operator $\gamma$ can be classified according to some desirable properties [30]. We say that $\gamma$ is:

- *locally finite*, if the number of generalisations generated for any given element by the operator is finite, that is, $\forall a \in \mathcal{S} : \gamma(a)$ is finite;
- *proper*, if an element is not equivalent to any of its generalisations, i.e., $\forall a, b \in \mathcal{S}$, if $b \in \gamma(a)$, then $a$ and $b$ are not equivalent;
- *complete*, if there are no generalisations that are not generated by the operator, i.e., $\forall a, b \in \mathcal{S}$ it holds that if $a \preceq b$, then $b \in \gamma^*(a)$ (where $\gamma^*(a)$ denotes the set of all elements which can be reached from $a$ by means of $\gamma$ in zero or a finite number of steps).

[4] A deeper analysis of refinement operators can be found in [30].

When a refinement operator is locally finite, proper, and complete it is said to be *ideal*. An ideal specialisation refinement operator for $\mathcal{EL}$ has been explored in [32]. In this paper, we define a generalisation refinement operator for $\mathcal{EL}^{++}$ and study its properties.

### 2.3 Computational Concept Blending by Amalgams

The process of conceptual blending can be characterised in terms of *amalgams* [37], a notion that has its root in case-based reasoning and focuses on the problem of combining solutions coming from multiple cases in search-based approaches to reuse and that has also been used to model analogy [9]. According to this approach, input concepts are generalised until a generic space is found, and pairs of generalised versions of the input concepts are 'combined' to create blends.

Formally, the notion of amalgams can be defined in any representation language $\mathcal{L}$ for which a subsumption relation between formulas (or descriptions) of $\mathcal{L}$ can be defined, and therefore also in $\mathcal{L}(\mathcal{T})$ with the subsumption relation $\sqsubseteq_\mathcal{T}$ for a given $\mathcal{EL}^{++}$ TBox $\mathcal{T}$.

**Definition 2.** *Given two descriptions $C_1, C_2 \in \mathcal{L}(\mathcal{T})$:*

- *A* most general specialisation *(MGS) is a description $C_{mgs}$ such that $C_{mgs} \sqsubseteq_\mathcal{T} C_1$ and $C_{mgs} \sqsubseteq_\mathcal{T} C_2$ and for any other description $D$ such that $D \sqsubseteq_\mathcal{T} C_1$ and $D \sqsubseteq_\mathcal{T} C_2$, then $D \sqsubseteq_\mathcal{T} C_{mgs}$.*
- *A* least general generalisation *(LGG) is a description $C_{lgg}$ such that $C_1 \sqsubseteq_\mathcal{T} C_{lgg}$ and $C_2 \sqsubseteq_\mathcal{T} C_{lgg}$ and for any other description $D$ such that $C_1 \sqsubseteq_\mathcal{T} D$ and $C_2 \sqsubseteq_\mathcal{T} D$, then $C_{lgg} \sqsubseteq_\mathcal{T} D$.*

Intuitively, a MGS is a description that has all the information from both the original descriptions $C_1$ and $C_2$, while a LGG contains that which is common to them. Depending on the structure of $\mathcal{T}$, it is not always possible to find a least general generalisation. Thus, the definition of $C_{lgg}$ is relaxed as follows.

**Definition 3.** *Given two descriptions $C_1, C_2 \in \mathcal{L}(\mathcal{T})$, a* common generalisation *is a description $C_g$ such that $C_1 \sqsubseteq_\mathcal{T} C_g$ and $C_2 \sqsubseteq_\mathcal{T} C_g$.*

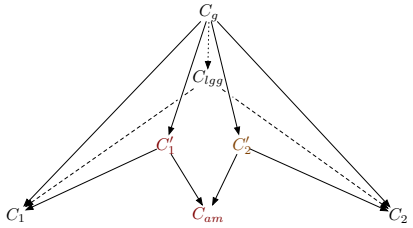
An *amalgam* of two descriptions is a new description that contains *parts from these original descriptions*. For instance, an amalgam of 'a red French sedan' and 'a blue German minivan' could be 'a red German sedan;' clearly, there are always multiple possibilities for amalgams, like 'a blue French minivan'. For the purposes of this paper we can define an amalgam of two descriptions as follows.

**Definition 4 (Amalgam).** *Let $\mathcal{T}$ be an $\mathcal{EL}^{++}$ TBox. A description $C_{am} \in \mathcal{L}(\mathcal{T})$ is an* amalgam *of two descriptions $C_1$ and $C_2$ (with common generalisation $C_g$) if there exist two descriptions $C_1'$ and $C_2'$ such that:*

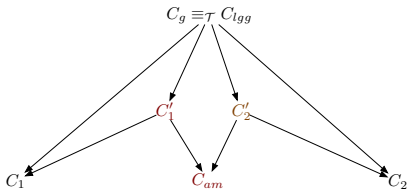
1. *$C_1 \sqsubseteq_\mathcal{T} C_1' \sqsubseteq_\mathcal{T} C_g$,*
2. *$C_2 \sqsubseteq_\mathcal{T} C_2' \sqsubseteq_\mathcal{T} C_g$, and*
3. *$C_{am}$ is a MGS of $C_1'$ and $C_2'$*

This definition is illustrated in Figure 1, where the common generalisation of the inputs is indicated as $C_g$, and the amalgam $C_{am}$ is the MGS of two concrete generalisations $C_1'$ and $C_2'$ of the inputs.

Notice that $C_g$ used to define the amalgam does not need to be a least general generalisation. Although having a least general generalisation is desirable, a common generalisation of the inputs will suffice.




(a) An amalgam with a common generalisation $C_g$ that is not a least general generalisation of $C_1$ and $C_2$.




(b) An amalgam with a common generalisation $C_g$ that is also a least general generalisation of $C_1$ and $C_2$.

Fig. 1: Two diagrams of an amalgam $C_{am}$ from descriptions $C_1$ and $C_2$ with generalisations $C'_1$ and $C'_2$. Arrows indicate the subsumption of the target by the source of the arrow.

In Section 4, we define an upward refinement operator that allows us to find generalisations of $\mathcal{EL}^{++}$ concept descriptions needed for computing the amalgams as described above. We may generalise concepts $C_1$ and $C_2$ beyond the LGG but we need to do this to guarantee termination, as we shall explain. We implement the operator and the search for generalisation in Answer Set Programming (ASP) [24]. To this end, we provide some basic notions about ASP in the next section.

### 2.4 Answer Set Programming

Answer Set Programming (ASP) is a declarative approach to solve NP-hard search problems (see e.g. [8, 24]). An ASP program is similar to a PROLOG program in that it is non-monotonic, takes logic programming style Horn clauses as input, and uses negation-as-failure (NaF). However, instead of using Kowalski [28]'s SLDNF resolution semantics as in PROLOG, it employs Gelfond and Lifschitz [25]'s Stable Model Semantics, which makes it truly declarative, i.e., the order in which ASP rules appear in a logic program does not matter. Furthermore, the Stable Model Semantics has the advantage that Answer Set Programs always terminate, while PROLOG programs do not. For example, given a program $p \leftarrow not\ q.$ and $q \leftarrow not\ p.$, asking whether $p$ holds results in an infinite loop for PROLOG, while ASP returns two stable models as solution, namely the sets $\{p\}$ and $\{q\}$.

An ASP program consists of a set of rules, facts and constraints. Its solutions are called *Stable Models* (SM). In this paper we only consider so-called *normal* rules [8], which are written as:

$$a_0 \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n \tag{2.1}$$

in which $a_1, ..., a_n$ are atoms and *not* is negation-as-failure. When $n = 0$ the rule $a_0 \leftarrow$ is known as a fact and the $\leftarrow$ is omitted. A constraint is a rule of the form $\leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n$. Constraints are rules that are used to discard some models of a logic program.

The models of an ASP program are defined according to the *stable model semantics*. The stable semantics is defined in terms of the so-called *Gelfond-Lifschitz reduction* [25]. Let $\mathcal{L}_P$ be the set of atoms in the language of a normal logic program $P$, then for any set $M \subseteq \mathcal{L}_P$, the Gelfond-Lifschitz reduction $P^M$ is the definite logic program obtained from $P$ by deleting:

**(i)** each rule that has a formula *not a* in its body with $a \in M$, and
**(ii)** all formulæ of the form *not a* in the bodies of the remaining rules.

$P^M$ does not contain *not* and $M$ is called a *stable model* of $P$ if and only if $M$ is the minimal model of $P^M$. A stable model $M$ of an ASP program $P$ contains those atoms that satisfy all the rules in the program and, consequently, represent a solution of the problem that represents.

ASP is interesting not only because can capture complex knowledge representation problems, but also because efficient ASP implementations exists. In particular, the *clingo* solver [22] offers a step-oriented, incremental approach that allows us to control and modify an ASP program at runtime, without the need of restarting the grounding the solving process from scratch. To this end, a program is partitioned into a base part, describing the static knowledge independent of a step parameter $t$, a cumulative part, capturing knowledge accumulating with increasing $t$, and a volatile part specific for each value of $t$. The grounding and integration of these subprograms into the solving process is completely modular and controllable from a scripting language such as Python.

The ASP implementation in this paper follows this methodology of specifying and solving a problem incrementally. For further details about incremental solving, we refer to [23] in which several examples can be found.

## 3 Conceptual Blending of Computer Icons

To exemplify our approach, we take the domain of computer icons into account. We consider computer icons as combinations of signs, such as *Document*, *MagnifyingGlass*, *HardDisk* and *Pen* that are described in terms of meanings [13]. Meanings convey *actions-in-the-world* or *object-types*.

Figure 2 shows the concept names defined in the *ComputerIcon* ontology and their relations. In what follows, concept names are capitalised (e.g., Sign) and role names are not (e.g., hasMeaning). We assume that a TBox $\mathcal{T}$ consists of two parts: one part that contains the background knowledge about the icon domain $\mathcal{T}_{bk}$, and another part that contains the domain knowledge about icon definitions $\mathcal{T}_{dk}$. $\mathcal{T}_{bk}$ contains the following axioms:

$\alpha_{\mathsf{bk}_1}$: Action $\sqsubseteq$ Meaning
$\alpha_{\mathsf{bk}_2}$: ObjectType $\sqsubseteq$ Meaning
$\alpha_{\mathsf{bk}_3}$: Search $\sqsubseteq$ Action
$\alpha_{\mathsf{bk}_4}$: Edit $\sqsubseteq$ Action
$\alpha_{\mathsf{bk}_5}$: HardDrive $\sqsubseteq$ ObjectType
$\alpha_{\mathsf{bk}_6}$: Doc $\sqsubseteq$ ObjectType

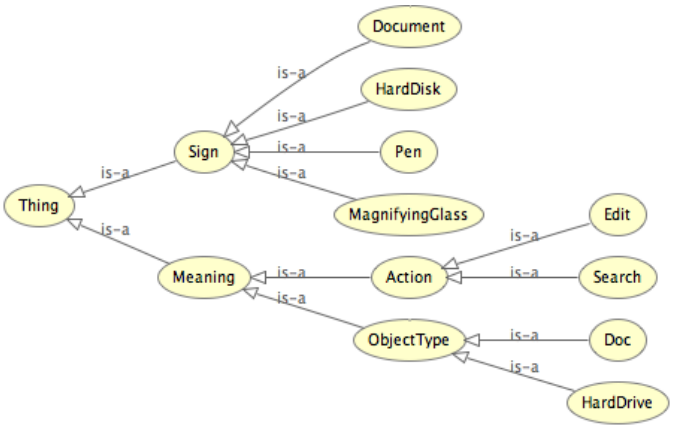



Fig. 2: The *ComputerIcon* ontology, showing the concept names and their relation.

$\alpha_{\mathsf{bk}_7}$: $\mathsf{Action} \sqcap \mathsf{ObjectType} \sqsubseteq \bot$
$\alpha_{\mathsf{bk}_8}$: $\mathsf{Search} \sqcap \mathsf{Edit} \sqsubseteq \bot$
... ...
$\alpha_{\mathsf{bk}_{14}}$: $\mathsf{HardDrive} \sqcap \mathsf{Doc} \sqsubseteq \bot$

Axioms $\alpha_{\mathsf{bk}_1}$-$\alpha_{\mathsf{bk}_6}$ capture the different meanings associated with signs; axioms $\alpha_{\mathsf{bk}_7}$-$\alpha_{\mathsf{bk}_{14}}$ model the disjointness among all Action and ObjectType concepts defined in the ontology. Signs are associated with a meaning. This is modeled by the hasMeaning role in the following axioms:

$\alpha_{\mathsf{bk}_{15}}$: $\mathsf{MagnifyingGlass} \equiv \mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Search}$
$\alpha_{\mathsf{bk}_{16}}$: $\mathsf{HardDisk} \equiv \mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{HardDrive}$
$\alpha_{\mathsf{bk}_{17}}$: $\mathsf{Pen} \equiv \mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Edit}$
$\alpha_{\mathsf{bk}_{18}}$: $\mathsf{Document} \equiv \mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Doc}$

$\alpha_{\mathsf{bk}_{19}}$: $\mathsf{MagnifyingGlass} \sqcap \mathsf{HardDisk} \sqsubseteq \bot$
... ...
$\alpha_{\mathsf{bk}_{25}}$: $\mathsf{Pen} \sqcap \mathsf{Document} \sqsubseteq \bot$

A sign is associated with a meaning. For instance, MagnifyingGlass is associated with Search to describe that it conveys the action of looking for something. Sign concepts are disjoint ($\alpha_{\mathsf{bk}_{19}}$-$\alpha_{\mathsf{bk}_{25}}$). Signs are related by spatial relationships such as isAboveIn, isAboveInLeft, isAboveInRight, isUpIn, isUpLeft, isUpRight, isDownIn, isDownLeft, and isDownRight. Spatial relationships are modelled as roles.

$\alpha_{\mathsf{bk}_{26}}$: $\mathsf{isAboveIn} \sqsubseteq \mathsf{isInSpatialRelation}$
$\alpha_{\mathsf{bk}_{27}}$: $\mathsf{isAboveLeft} \sqsubseteq \mathsf{isInSpatialRelation}$
$\alpha_{\mathsf{bk}_{28}}$: $\mathsf{isAboveRight} \sqsubseteq \mathsf{isInSpatialRelation}$
... ...
$\alpha_{\mathsf{bk}_{37}}$: $\mathsf{isDownRight} \sqsubseteq \mathsf{isInSpatialRelation}$

For the sake of simplicity, we assume that icons are modelled according to a canonical form. Axioms describing icon concepts are of the form $\mathsf{IconName} \equiv \mathsf{C} \sqcap \exists \mathsf{r}.\mathsf{D}$, where r is a spatial relation

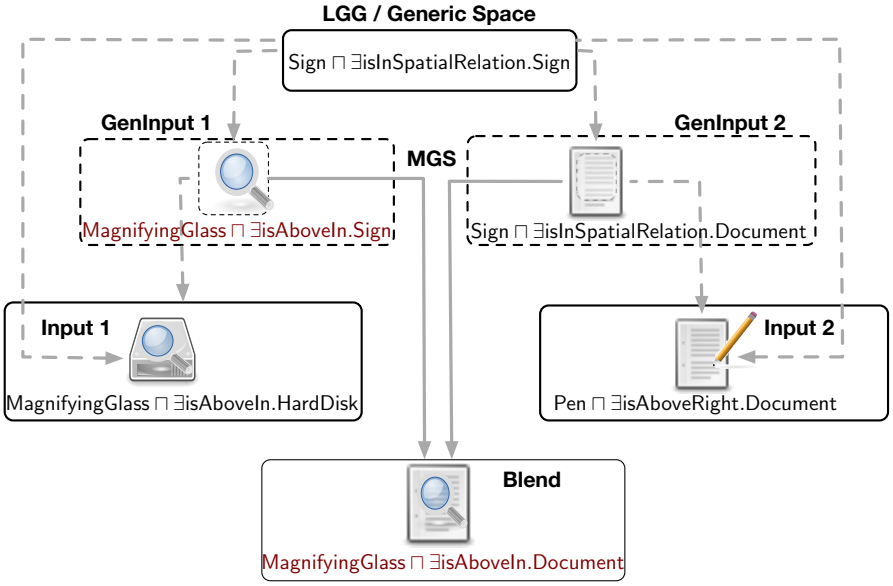



Fig. 3: Blending the SearchHardDisk and EditDocument icon concepts into a new concept representing a search-in-document icon. Sign's meanings are not represented.

and C, D are concepts that describe signs. Based on this canonical form and on the axioms above, we modeled some icons as domain knowledge of a TBox.

*Example 1.* SearchHardDisk is an icon that consists of two signs MagnifyingGlass and HardDisk, where the MagnifyingGlass sign is above in the middle of the HardDisk sign. Another icon is EditDocument, where the Pen sign is above on the right of the Document sign. Both icons are shown in Figure 3.

$$\alpha_{\mathsf{dk}_1}\colon \mathsf{SearchHardDisk} \equiv \mathsf{MagnifyingGlass} \sqcap \exists \mathsf{isAboveIn}.\mathsf{HardDisk}$$
$$\alpha_{\mathsf{dk}_2}\colon \mathsf{EditDocument} \equiv \mathsf{Pen} \sqcap \exists \mathsf{isAboveRight}.\mathsf{Document}$$

We consider the above knowledge as a library of icons. We assume that the library is managed and used by a computer icon design tool. The tool accepts a query as input and retrieves those icons that satisfy certain properties. For instance, a query asking for an icon with the meaning of searching in a hard-disk will retrieve the SearchHardDisk concept. In contrast, a query asking for an icon with the meaning of searching in a document does not return any result. In such a case, the tool tries to answer the query by running a conceptual blending algorithm.

Intuitively, the conceptual blending algorithm works as follows. Given two input concepts, the algorithm tries to create new concepts that can satisfy the query. New concepts are created by taking the commonalities and some of their specifics into account (Figure 3). For instance, both SearchHardDisk and EditDocument are icons that consist of two signs related by a spatial relation (the generic space). Then, if we keep the MagnifyingGlass concept from SearchHardDisk and the Document concept from EditDocument, and we generalise the HardDisk and Pen concepts and the role isAboveRight, we can blend the generalised input concepts of SearchHardDisk and EditDocument into a new concept representing an icon whose meaning is to search in a document.

MagnifyingGlass ⊓ ∃isAboveIn.Document

In this paper, we show how the above concept generation description can be computationally realised by two processes. An ASP-based implementation that generalises $\mathcal{EL}^{++}$ concept descriptions and finds a generic space; and a procedural implementation that generates and evaluates the blended concepts. First, we introduce a refinement operator for generalising an $\mathcal{EL}^{++}$ concept.

## 4 A Generalisation Refinement Operator for $\mathcal{EL}^{++}$

In any description logic the set of concept descriptions are ordered under the subsumption relation forming a quasi-ordered set. For $\mathcal{EL}^{++}$ in particular they form a bounded meet-semilattice with conjunction as meet operation, $\top$ as greatest element, and $\bot$ as least element.[5] In order to define a generalisation refinement operator for $\mathcal{EL}^{++}$, we need some auxiliary definitions.

**Definition 5.** *Let $\mathcal{T}$ be an $\mathcal{EL}^{++}$ TBox. The set of* subconcepts *of $\mathcal{T}$ is given as*

$$\mathsf{sub}(\mathcal{T}) = \{\top, \bot\} \cup \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(C) \cup \mathsf{sub}(D) \tag{4.1}$$

*where* sub *is inductively defined over the structure of concept descriptions as follows:*

$$\begin{aligned} \mathsf{sub}(A) &= \{A\} \\ \mathsf{sub}(\bot) &= \{\bot\} \\ \mathsf{sub}(\top) &= \{\top\} \\ \mathsf{sub}(C \sqcap D) &= \{C \sqcap D\} \cup \mathsf{sub}(C) \cup \mathsf{sub}(D) \\ \mathsf{sub}(\exists r.C) &= \{\exists r.C\} \cup \mathsf{sub}(C) \end{aligned}$$

Based on $\mathsf{sub}(\mathcal{T})$, we define the upward cover set of atomic concepts and roles. $\mathsf{sub}(\mathcal{T})$ guarantees the following upward cover set to be finite.[6]

**Definition 6.** *Let $\mathcal{T}$ be an $\mathcal{EL}^{++}$ TBox with concept names from $N_C$. The* upward cover set *of an atomic concept $A \in N_C \cup \{\top, \bot\}$ and of a role $r \in N_R$ with respect to $\mathcal{T}$ is given as:*

$$\begin{aligned} \mathit{UpCov}(A) := \{&C \in \mathsf{sub}(\mathcal{T}) \mid A \sqsubseteq_{\mathcal{T}} C \\ &\textit{and there is no } C' \in \mathsf{sub}(\mathcal{T}) \\ &\textit{such that } A \sqsubset_{\mathcal{T}} C' \sqsubset_{\mathcal{T}} C\} \end{aligned} \tag{4.2}$$

$$\begin{aligned} \mathit{UpCov}(r) := \{&s \in N_R \mid r \sqsubseteq_{\mathcal{T}} s \\ &\textit{and there is no } s' \in N_r \\ &\textit{such that } r \sqsubset_{\mathcal{T}} s' \sqsubset_{\mathcal{T}} s\} \end{aligned} \tag{4.3}$$

[5] A bounded meet-semilattice is a partially ordered set which has a meet (or greatest lower bound) for any nonempty finite subset.

[6] We assume that $\mathcal{T}$ is finite.

We can now define our generalisation refinement operator for $\mathcal{EL}^{++}$ as follows.

**Definition 7.** *Let $\mathcal{T}$ be an $\mathcal{EL}^{++}$ TBox. We define the* generalisation refinement operator $\gamma$ *inductively over the structure of concept descriptions as follows:*

$$\begin{aligned}
\gamma(A) &= \mathsf{UpCov}(A) \\
\gamma(\top) &= \mathsf{UpCov}(\top) = \emptyset \\
\gamma(\bot) &= \mathsf{UpCov}(\bot) \\
\gamma(C \sqcap D) &= \{C' \sqcap D \mid C' \in \gamma(C)\} \cup \{C \sqcap D' \mid D' \in \gamma(D)\} \cup \{C, D\} \\
\gamma(\exists r.C) &= \begin{cases} \gamma_r(\exists r.C) \cup \gamma_C(\exists r.C) & \text{whenever } \mathsf{UpCov}(r) \neq \emptyset \text{ or } \gamma(C) \neq \emptyset \\ \{\top\} & \text{otherwise.} \end{cases}
\end{aligned}$$

*where $\gamma_r$ and $\gamma_C$ are defined as:*

$$\begin{aligned}
\gamma_r(\exists r.C) &= \{\exists s.C \mid s \in \mathsf{UpCov}(r)\} \\
\gamma_C(\exists r.C) &= \{\exists r.C' \mid C' \in \gamma(C)\}
\end{aligned}$$

Given a generalisation refinement operator $\gamma$, $\mathcal{EL}^{++}$ concepts are related by refinement paths as described next.

**Definition 8.** *A finite sequence $C_1, \ldots, C_n$ of $\mathcal{EL}^{++}$ concepts is a* concept refinement path $C_1 \xrightarrow{\gamma} C_n$ *from $C_1$ to $C_n$ of the generalisation refinement operator $\gamma$ iff $C_{i+1} \in \gamma(C_i)$ for all $i : 1 \leq i < n$. $\gamma^*(C)$ denotes the set of all concepts that can be reached from $C$ by means of $\gamma$ in zero or a finite number of steps.*

**Proposition 1.** *The operator $\gamma$ is a generalisation refinement operator over the set of all $\mathcal{EL}^{++}$ concepts with the order $\sqsubseteq_{\mathcal{T}}$.*

*Proof.* We need to prove that for every $\mathcal{EL}^{++}$ concept $C$ and every $D \in \gamma(C)$, the subsumtion $C \sqsubseteq_{\mathcal{T}} D$ holds. We do this by induction on the structure of $C$. If $C$ is a concept name, $\top$, or $\bot$, the subsumption holds directly by definition. If $C$ is of the form $C_1 \sqcap C_2$, we can assume w.l.o.g. that $D$ is $C' \sqcap C_2$ for some $C' \in \gamma(C_1)$. By induction hypothesis, $C_1 \sqsubseteq_{\mathcal{T}} C'$ and hence $C_1 \sqcap D \sqsubseteq_{\mathcal{T}} C' \sqcap D$. Finally, if $C$ is of the form $\exists r.C_1$ we have three possible cases. If $\mathsf{UpCov}(r) \neq \emptyset$, and $D$ is $\exists s.C_1$ for $s \in \mathsf{UpCov}(r)$ then by definition $\exists r.C_1 \sqsubseteq_{\mathcal{T}} \exists s.C_1$. If $\mathsf{UpCov}(C) \neq \emptyset$, $C \neq \top$ and $D$ must be of the form $\exists r.C'$ with $C_1 \sqsubseteq_{\mathcal{T}} C'$, and hence the subsumption holds. In the last case, $D$ is equivalent to $\top$, and hence the subsumption follows trivially. □

We now analyse the properties of the generalisation refinement operator $\gamma$. Observe first that our definition of $\mathsf{UpCov}$ for basic concepts and roles only considers the set of subconcepts present in a TBox $\mathcal{T}$. This guarantees that $\gamma$ is locally finite, since at each generalisation step, the set of possible generalisations is finite.

**Proposition 2.** *The generalisation refinement operator $\gamma$ is locally finite.*

*Proof.* We prove that for every $\mathcal{EL}^{++}$ concept $C$, $\gamma(C)$ is finite by induction on the structure of $C$.

For $A \in N_C \cup \{\top, \bot\}$, we have that $\gamma(A) \subseteq \mathsf{sub}(\mathcal{T})$. Since $\mathsf{sub}(\mathcal{T})$ is finite, the result immediately holds. For $C \sqcap D$, we have that $|\gamma(C \sqcap D)| \leq |\gamma(C)| + |\gamma(D)|$. By induction hypothesis, the two sets on the right-hand side of this inequality are finite, and hence $\gamma(C \sqcap D)$ must be finite too. Finally, it holds that $|\gamma(\exists r.C)| \leq |\mathsf{UpCov}(r)| + |\gamma(C)|$. By the fact that $\mathsf{UpCov}(r) \subseteq N_R$, which is finite, and the induction hypothesis, the result follows. □

When generalising concept names and role names, we always ensure that the resulting concepts are more general (w.r.t. the TBox $\mathcal{T}$) than the original elements. Unfortunately, this does not guarantee that $\gamma$ is proper.

*Example 2.* Let $\mathcal{T} := \{A \sqsubseteq B\}$. Then, following Definition 7, we have that generalising the concept $A \sqcap B$ can yield $A \sqcap \top$. However, both these concepts are equivalent to $A$ w.r.t. $\mathcal{T}$. Therefore, $\gamma$ is not proper.

One possible way to avoid this situation, and, therefore, to guarantee the properness of $\gamma$, is to redefine it with an additional semantic test. More precisely, let $\gamma'$ be defined as:

$$\gamma'(C) := \gamma(C) \backslash \{D \in \gamma(C) \text{ such that } D \equiv_{\mathcal{T}} C\} \tag{4.4}$$

Essentially, $\gamma'$ discards those generalisations that are equivalent to the concept being generalised. It is easy to see that $\gamma'$ is still a finite generalisation refinement operator and it is proper.

**Proposition 3.** *The generalisation refinement operator $\gamma'$ is proper.*

*Proof.* This proposition trivially follows from Eq. 4.4.

The repetitive application of the generalisation refinement operator allows one to find a description that represents the properties that two or more $\mathcal{EL}^{++}$ concepts have in common. This description is a common generalisation of $\mathcal{EL}^{++}$ concepts, the so-called generic space that is used in conceptual blending.

**Definition 9.** *An $\mathcal{EL}^{++}$ concept description $G$ is a generic space of the $\mathcal{EL}^{++}$ concept descriptions $C_1, \ldots, C_n$ if and only if $G \in \gamma'^*(C_i)$ for all $i = 1, \ldots, n$.*

*Example 3.* Let us consider the $\mathcal{EL}^{++}$ concepts EditDocument and SearchHardDisk defined in Example 1. It can be checked that:

$\{(\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Action}) \sqcap \exists \mathsf{isInSpatialRelation}.(\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{ObjectType})\} \in \gamma'^*(\mathsf{EditDocument})$
$\{(\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Action}) \sqcap \exists \mathsf{isInSpatialRelation}.(\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{ObjectType})\} \in \gamma'^*(\mathsf{SearchHardDisk})$

$(\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Action}) \sqcap \exists \mathsf{isInSpatialRelation}.(\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{ObjectType})$ is a generic space (Definition 9) of EditDocument and SearchHardDisk.

Unfortunately, due to the fact that upward cover set we defined only takes subconcepts already present in the TBox into account, neither $\gamma$ nor its refinement $\gamma'$ are complete; that is, these operators may fail to compute some of the generalisations of a given $\mathcal{EL}^{++}$ concept.

*Example 4.* Let $\mathcal{T} := \{A \sqsubseteq B, A \sqsubseteq C\}$. Then, generalising the concept $A$ yields $\gamma(A) = \{B, C\}$. However, $B \sqcap C$ is also a possible generalisation of $A$ w.r.t. $\sqsubseteq_{\mathcal{T}}$.

More generally, as the following theorem shows, no generalisation refinement operator over $\mathcal{EL}^{++}$ concepts w.r.t. $\sqsubseteq_{\mathcal{T}}$ can be locally finite, proper, and complete.

**Theorem 1.** *There is no ideal generalisation refinement operator for $\mathcal{EL}^{++}$ concepts.*

*Proof.* Consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A, \exists r.A \sqsubseteq A\}$, and define the concepts $G_0 := \top, G_{i+1} := \exists r.G_i$ for all $i \geq 0$. Notice first that these concepts form an infinite chain of generalisations $G_0 \sqsupset_{\mathcal{T}} G_1 \sqsupset_{\mathcal{T}} G_2 \sqsupset_{\mathcal{T}} \cdots \sqsupset_{\mathcal{T}} A$. Moreover, every $\mathcal{EL}^{++}$ concept $C$ with $A \sqsubset_{\mathcal{T}} C$ is equivalent (w.r.t. $\mathcal{T}$) to one such $G_i$. Let now $\gamma$ be a locally finite and proper generalisation refinement operator. Then $\gamma(A)$ is a finite set of concepts which, w.l.o.g. we can assume to be of the form $\{G_i \mid i \in I\}$, where $I$ is a finite set of indices. In particular, $I$ contains a maximum index $n$. Then $G_{n+1}$ is strictly more specific than all elements of $\gamma(A)$ and cannot be derived by further applications of $\gamma$. Thus, $\gamma$ is not complete. □

Since the generalisation refinement operator is not complete, it cannot guarantee to find a generic space that is a *least* general generalisation. Although having a least general generalisation is desirable, finding a common description, which allows us creating new $\mathcal{EL}^{++}$ concepts from existing ones by conceptual blending, will suffice.

At this point, we should note, however, that the generalisation refinement operator may even fail to find a generic space of a set of $\mathcal{EL}^{++}$ concepts. Indeed, as the following example shows, $\gamma'$ can produce an infinite chain of generalisations.

*Example 5.* Let $\mathcal{T} := \{A \sqsubseteq \exists r.A, B \sqsubseteq \top\}$. Then, the generalisation of the concept description $B$ can yield $\top$. The generalisation of the concept description $A$ yields the concept defined as $\{\exists r.\exists r.\cdots\exists r.A\}$. A common (trivial) generalisation for $A$ and $B$ is $\top$ but it is not computed by $\gamma'$.

Not finding a common generalisation of a set of $\mathcal{EL}^{++}$ concepts is a not a new problem in the DL literature. Different solutions have been proposed [1, 2, 6, 44, 45]. Typically, some assumptions are made over the structure of the TBox or a fixed role depth of concepts is considered. In the following, we adopt the latter view, and we restrict the number of nested quantifiers in a concept description to a fixed constant $k$. To this end, we introduce the definition of role depth of a concept as follows.

**Definition 10.** *The role depth of an $\mathcal{EL}^{++}$ concept description $C$ is defined as the maximum number of nested (existential) quantifiers in $C$:*

$$\begin{aligned} \mathsf{roleDepth}(\top) = \mathsf{roleDepth}(A) &= 0, \\ \mathsf{roleDepth}(C \sqcap D) &= \max\{\mathsf{roleDepth}(C), \mathsf{roleDepth}(D)\}, \\ \mathsf{roleDepth}(\exists r.C) &= \mathsf{roleDepth}(\mathsf{C}) + 1 \end{aligned}$$

Based on the role depth of a concept we modify the definition of the generalisation refinement operator $\gamma'$ to take a fixed constant $k \in \mathbb{N}_{>0}$ of nested quantifiers into account. More precisely, let

$\gamma'_k$ be defined as $\gamma'$, except that for the case of generalising a concept $\exists r.C$ we set:

$$\gamma'_k(\exists r.C) := \begin{cases} \gamma_r(\exists r.C) \cup \gamma_C(\exists r.C) & \text{if } (\mathsf{UpCov}(r) \neq \emptyset \text{ or } \gamma(C) \neq \emptyset) \text{ and} \\ & \mathsf{roleDepth}(C) \leq k, \\ \{\top\} & \text{otherwise.} \end{cases}$$

The role depth prevents the generalisation refinement operator from generating infinite chains of generalisations. Consequently, it can ensure that a generic space between $\mathcal{EL}^{++}$ concepts can always be found.

**Definition 11.** *An $\mathcal{EL}^{++}$ concept description $G^k$ is a k-approximation of a generic space of the $\mathcal{EL}^{++}$ concept descriptions $C_1, \ldots, C_n$ if and only if $G^k \in \gamma'^*_k(C_i)$ for all $i = 1, \ldots, n$.*

**Proposition 4.** *There always exists a k-approximation $G^k$ for any $\mathcal{EL}^{++}$ concept descriptions $C_1, \ldots, C_n$.*

*Proof.* The proof of this proposition can be done by noticing that every concept can always be generalised to $\top$ in a finite number of applications of $\gamma'_k$. Therefore, $\top$ is always a generic space of any concept descriptions $C_1, \ldots, C_n$. □

The role depth not only avoids infinite chains of generalisations, but also provides a way to maintain the structure of the input concepts in conceptual blending. For instance, by choosing the value of $k$ as the maximum role depth of the input concepts to be blended, the operator yields generalisations with a similar role structure.

## 5 Implementing Upward Refinement in ASP

We consider an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$ that consists of a background knowledge $\mathcal{T}_{bk}$ and a domain knowledge $\mathcal{T}_{dk}$. A generic space between $\mathcal{EL}^{++}$ concepts in the domain knowledge is found by means of an ASP program that generalises $\mathcal{T}_{dk}$ in a step-wise transition process. Since finding a generic space of $n$ concepts can be reduced to the problem of finding a generic space between pairs of concepts [3], the ASP program we devise takes two $\mathcal{EL}^{++}$ concepts into account.

In what follows, we describe how an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$ is translated into an ASP representation needed for implementing the generic space search. Table 2 shows the main predicates used in the ASP implementation.[7]

### 5.1 Modeling $\mathcal{EL}^{++}$ concepts in ASP

For each concept name $A \in N_C$ in $\mathcal{T}_{bk}$, we state the fact:

$$concept(A) \tag{5.1}$$

[7] Disjointness axioms are not translated to ASP because they are not used in the generalisation process.

| **Predicates modeling $\mathcal{EL}^{++}$ concepts** | **Description** |
|---|---|
| $dConcept(\mathcal{C})$ | A reference to a domain knowledge concept $\mathcal{C}$ |
| $concept(A)$ | A concept $A$ |
| $subConcept(A,B)$ | A concept $B$ subsumes $A$ |
| $role(r)$ | A role $r$ |
| $subRole(r,s)$ | A role $r$ subsumes $s$ |
| $hasConjunct(\mathcal{C},ex,A,t)$ | A concept $A$ is an expression $ex$ in $\mathcal{C}$ at step $t$ |
| $hasRoleEx(\mathcal{C},roleEx,r,depth,A,t)$ | A concept $A$ fills the role $r$ in a role expression $roleEx$ with depth $depth$ in $\mathcal{C}$ at step $t$ |

| **Predicates modeling the refinement** | **Description** |
|---|---|
| $notEqual(\mathcal{C}_1,\mathcal{C}_2,t)$ | The domain concepts $\mathcal{C}_1$, $\mathcal{C}_2$ are not equivalent at step $t$ |
| $conjunctNotEq(\mathcal{C}_1,\mathcal{C}_2,A,t)$ | The concept $A$ is not equivalent in $\mathcal{C}_1$ and $\mathcal{C}_2$ at step $t$ |
| $hasRoleExNotEq(\mathcal{C}_1,\mathcal{C}_2,C,t)$ | A conjunct $C$ is not equivalent in the $\mathcal{C}_1$, $\mathcal{C}_2$ at step $t$ |
| $roleInExpressionNotEq(\mathcal{C}_1,\mathcal{C}_2,C,r,t)$ | A role $r$ in a conjunct $C$ is not equivalent in $\mathcal{C}_1$, $\mathcal{C}_2$ at step $t$ |
| $app(a,\mathcal{C},t)$ | A refinement step $a$ is applicable in $\mathcal{C}$ at step $t$ |
| $poss(a,\mathcal{C},t)$ | A refinement step $a$ is possible in $\mathcal{C}$ at step $t$ |
| $exec(a,\mathcal{C},t)$ | A refinement step $a$ is executed in $\mathcal{C}$ at step $t$ |

Table 2: Overview of the main predicates used to formalise the upward refinement process in ASP. The predicates in the top table are used to model $\mathcal{EL}^{++}$ concepts, whereas predicates in the table below are used to model the refinement operators.

For each role $r \in N_R$ in $\mathcal{T}_{bk}$ with $domain(r) \sqsubseteq C$ and $range(r) \sqsubseteq D$, we state the facts:

$$role(r) \tag{5.2a}$$
$$domain(r,C) \tag{5.2b}$$
$$range(r,D) \tag{5.2c}$$

For each inclusion axiom $A \sqsubseteq B \in \mathcal{T}_{bk}$ and $A$, $B$ are atomic concepts, we state the fact:

$$subConcept(A,B) \tag{5.3}$$

Similarly, for each role inclusion axiom $r \sqsubseteq s \in \mathcal{T}_{bk}$, we state the fact:

$$subRole(r,s) \tag{5.4}$$

For each inclusion axiom $A \sqsubseteq C \in \mathcal{T}_{bk}$ in which $A$ is an atomic concept and $C$ is a complex concept, we call $C$ the concept definition of $A$ and denote it as $\mathcal{C}$ within the following facts:

$$concept(\mathcal{C}) \tag{5.5a}$$
$$subConcept(A,\mathcal{C}) \tag{5.5b}$$

Then, $\mathcal{C}$ is translated to ASP facts by means of the following function:

$$\begin{aligned}
\mathsf{toASP}(\mathcal{C}, ex_{(k)}, C \sqcap D, depth) = \{&hasConjunct(\mathcal{C}, ex_{(k)}, subEx_{(k+1)}), \\
&hasConjunct(\mathcal{C}, ex_{(k)}, subEx_{(k+2)})\} \\
\cup\ \{&\mathsf{toASP}(\mathcal{C}, subEx_{(k+1)}, C, depth)\} \\
\cup\ \{&\mathsf{toASP}(\mathcal{C}, subEx_{(k+2)}, D, depth)\}
\end{aligned} \tag{5.6a}$$

$$\mathsf{toASP}(\mathcal{C}, ex_{(k)}, \top, depth) = \{hasConjunct(\mathcal{C}, ex_{(k)}, Thing)\} \tag{5.6b}$$

$$\mathsf{toASP}(\mathcal{C}, ex_{(k)}, B, depth) = \{hasConjunct(\mathcal{C}, ex_{(k)}, B)\} \tag{5.6c}$$

$$\begin{aligned}
\mathsf{toASP}(\mathcal{C}, ex_{(k)}, \exists r.B, depth) = \{&hasConjunct(\mathcal{C}, ex_{(k)}, roleEx_{(k)}), \\
&hasRoleEx(\mathcal{C}, roleEx_{(k)}, depth, r, B)\}
\end{aligned} \tag{5.6d}$$

$$\begin{aligned}
\mathsf{toASP}(\mathcal{C}, ex_{(k)}, \exists r.C, depth) = \{&hasConjunct(\mathcal{C}, ex_{(k)}, roleEx_{(k)}), \\
&hasRoleEx(\mathcal{C}, roleEx_{(k)}, depth, r, subEx_{(k+1)})\} \\
\cup\ \{&\mathsf{toASP}(\mathcal{C}, subEx_{(k+1)}, C, depth+1)\}
\end{aligned} \tag{5.6e}$$

toASP models a complex concept description as a set of *hasConjunct/3* and *hasRoleEx/5* predicates that are generated by recursively traversing its structure. $ex_{(k)}$ and $roleEx_{(k)}$ are atoms that are dynamically generated during the translation; $k$ is a counter that let the predicates be identifiable in a unique way, and *depth* is used to count the depth of a role $r$. A conjunction in a concept description is modeled by means of *hasConjunct/3* predicates. For instance, if $\mathcal{C} = C \sqcap D$, then the predicates $hasConjunct(\mathcal{C}, ex_{(1)}, subEx_{(2)})$ and $hasConjunct(\mathcal{C}, ex_{(1)}, subEx_{(3)})$ model the conjunction. (Eq. 5.6a). The translation of $C$ and $D$ is done through the recursive calls $\mathsf{toASP}(\mathcal{C}, subEx_{(2)}, C, 1)$ and $\mathsf{toASP}(\mathcal{C}, subEx_{(3)}, D, 1)$ respectively. Role expressions are modeled by means of *hasConjunct/3* and *hasRoleEx/5* predicates. For instance, if $D = \exists r.B$, then the role expression $\exists r.B$ is modeled by the predicates $hasConjunct(\mathcal{C}, subEx_{(3)}, roleEx_{(3s)})$ and $hasRoleEx(\mathcal{C}, roleEx_{(3)}, 1, r, B)$. The former predicate states that the expression $subEx_{(3)}$—referring to $D$—has a role expression $roleEx_{(3)}$. The latter predicate models that, in the the complex concept $\mathcal{C}$, the expression $roleEx_{(3)}$ has a concept $B$ filling the role $r$, and that the depth of $r$ is 1 (Eq. 5.6d). Cases 5.6b-5.6c-5.6e can be explained in a similar way.

While the background knowledge is static, the domain knowledge changes. To this end, we need to keep track of the generalisations applied to each domain concept. This is done by modeling a concept in the domain knowledge by means of the predicates *hasConjunct* and *hasRoleEx* with an extra atom, $t$, that is a step-counter representing the number of modifications made to the concept.

For each axiom $A \equiv C \in \mathcal{T}_{dk}$, in which $A$ is a concept in the domain knowledge and $C$ is its definition, we denote it by $\mathcal{C}$ and we add the following fact:

$$dConcept(\mathcal{C}) \tag{5.7}$$

Then, $\mathcal{C}$ is translated to ASP in the following way:

1. $\mathcal{C}$ is rewritten to $\mathcal{C}'$ by using all the axiom definitions in the background knowledge;
2. $\mathcal{C}'$ is translated to ASP by means of the function toASP with the only difference that the predicates *hasConjunct* and *hasRoleEx* have an extra atom $t$, equal to 0.

To exemplify the translation process, we provide the following example.

*Example 6.* Let us consider the TBox and the domain concept EditDocument in Section 3. The background knowledge is translated to the following ASP facts:

| | | |
|---|---|---|
| Sign $\sqsubseteq$ Thing | $concept(Sign).$<br>$concept(Thing).$<br>$subConcept(Document, Thing).$ | By Eq. 5.1<br><br>By Eq. 5.3 |
| Document $\sqsubseteq$ Sign | $concept(Document).$<br>$subConcept(Document, Sign).$ | By Eq. 5.1<br>By Eq. 5.3 |
| ... | ... | |
| $domain$(isAboveIn) $\sqsubseteq$ Sign<br>$range$(isAboveIn) $\sqsubseteq$ Sign | $role(isAboveIn).$<br>$domain(isAboveIn, Sign).$<br>$range(isAboveIn, Sign).$ | By Eq. 5.2a<br>By Eq. 5.2b<br>By Eq. 5.2c |
| ... | ... | |
| isAboveIn $\sqsubseteq$ isInSpatialRelation | $subRole(isAboveIn, isInSpatialRelation).$ | By Eq. 5.4 |
| ... | ... | |

The concept EditDocument is translated to the following ASP facts:

| | |
|---|---|
| Pen $\sqcap \exists$isRightIn.Document | |
| (Sign $\sqcap \exists$hasMeaning.Edit) $\sqcap \exists$isRightIn.(Sign $\sqcap \exists$hasMeaning.Doc) | |
| $dConcept(EditDocument).$ | By Eq. 5.7 |
| $hasConjunct(EditDocument, ex_{(1)}, subEx_{(2)}, 0).$ | By Eq. 5.6a |
| $hasConjunct(EditDocument, ex_{(1)}, subEx_{(3)}, 0).$ | By Eq. 5.6a |
| $hasConjunct(EditDocument, subEx_{(2)}, subEx_{(3)}, 0).$ | By Eq. 5.6a |
| $hasConjunct(EditDocument, subEx_{(2)}, subEx_{(4)}, 0).$ | By Eq. 5.6a |
| $hasConjunct(EditDocument, subEx_{(3)}, Sign, 0).$ | By Eq. 5.6c |
| $hasConjunct(EditDocument, subEx_{(4)}, roleEx_{(4)}, 0).$ | By Eq. 5.6d |
| $hasRoleEx(EditDocument, roleEx_{(4)}, 1, hasMeaning, Edit, 0).$ | By Eq. 5.6d |
| $hasConjunct(EditDocument, subEx_{(3)}, roleEx_{(3)}, 0).$ | By Eq. 5.6e |
| $hasRoleEx(EditDocument, roleEx_{(3)}, 1, isRightOn, subEx_{(6)}, 0).$ | By Eq. 5.6e |
| $hasConjunct(EditDocument, subEx_{(4)}, subEx_{(5)}, 0).$ | By Eq. 5.6a |
| $hasConjunct(EditDocument, subEx_{(4)}, subEx_{(6)}, 0).$ | By Eq. 5.6a |
| $hasConjunct(EditDocument, subEx_{(5)}, Sign, 0).$ | By Eq. 5.6c |
| $hasConjunct(EditDocument, subEx_{(6)}, roleEx_{(6)}, 0).$ | By Eq. 5.6d |
| $hasRoleEx(EditDocument, roleEx_{(6)}, 1, hasMeaning, Doc, 0).$ | By Eq. 5.6d |

Besides, we model the concept $\top$ as the fact $concept(Thing)$, and for each concept name $A \in N_C$, which is not already subsumed by other concept names, we add a fact $subConcept(A, Thing)$. We check for (in)equality of domain concepts $\mathcal{C}_1$ and $\mathcal{C}_2$ by a predicate $notEqual(\mathcal{C}_1, \mathcal{C}_2, t)$. The predicate is true whenever conjuncts, role expressions and roles are not equal in $\mathcal{C}_1$ and $\mathcal{C}_2$.

### 5.2 Formalising upward refinement in ASP

We consider each step of the refinement operator in Definition 7 as an operator type by itself. We consider five types of generalisation that can be applied to a concept in the domain knowledge at each step:

1. The generalisation of an atomic concept, and we denote it as $\gamma_A$;
2. The generalisation of a concept filling the range of a role up to a role depth $k$ ($\gamma_C$);
3. The generalisation of a role ($\gamma_r$);
4. The removal of a role, and we denote it as $\gamma_{r^-}$;

5. The removal of a concept, and we denote it as $\gamma_{C^-}$.

We treat each upward refinement operator type as an action. To this end, we model each operator type via a *precondition* rule, an *inertia* rule, and an *effect* rule. Preconditions are modelled with a predicate *app/3* that states when an operator type is *applicable*. Inertia is modelled with different non-inertial predicates that state when an element in a domain concept remains unchanged after the execution of a refinement operator type. Effect rules model how a refinement operator type changes a concept in the domain knowledge. We represent the execution of an upward refinement operator type with an atom $exec(\gamma_x, \mathcal{C}, t)$. This atom denotes that a generalisation operator type $\gamma_x \in \{\gamma_A, \gamma_C, \gamma_r, \gamma_{r^-}, \gamma_{C^-}\}$ is applied to $\mathcal{C}$ at step $t$.

**Upward refinement of atomic concepts.** A fact $app(genConcept(Ex, A, B), \mathcal{C}, t)$ denotes the applicability of the generalisation of a concept $A$ to a concept $B$ in a conjunct $Ex$ of $\mathcal{C}$ at step $t$ using $\gamma_A$:

$$\begin{aligned} app(genConcept(Ex, A, B), \mathcal{C}_1, t) \leftarrow & \\ & hasConjunct(\mathcal{C}_1, Ex, A, t), \\ & subConcept(A, B), \\ & \text{not } hasRoleEx(\mathcal{C}_1, A, \_, \_, \_, t), \\ & \text{not } hasConjunct(\mathcal{C}_1, A, \_, t), \\ & conjunctNotEq(\mathcal{C}_1, \mathcal{C}_2, A, t), \\ & \text{not } exec(genConcept(Ex, A, B), \mathcal{C}_2, t), dConcept(\mathcal{C}_2) \end{aligned} \tag{5.8}$$

There are several preconditions for generalising an atomic concept in a conjunct $Ex$. First, $Ex$ involves a concept $A$ that has a parent concept $B$ in the subsumption hierarchy defined by the axioms of the TBox (first two EDB predicates). Second, $Ex$ is neither a role expression nor a complex expression. Third, $A$ is not equivalent in $\mathcal{C}_1$ and $\mathcal{C}_2$ (*conjunctNotEq/4*). This latter atom is true when either $\mathcal{C}_1$ or $\mathcal{C}_2$ does not contain $A$. Another condition is that $A$ is not being generalised in $\mathcal{C}_2$, since we want to keep elements that are common in $\mathcal{C}_1$ and $\mathcal{C}_2$.

We also need a simple inertia rule for generalising a concept in a conjunct. This is as follows:

$$\begin{aligned} noninertialGenConcept(\mathcal{C}, Ex, A, t) \leftarrow exec(genConcept(Ex, A, \_), \mathcal{C}, t), & \\ hasConjunct(\mathcal{C}, Ex, A, t) & \end{aligned} \tag{5.9}$$

*noninertialGenConcept* atoms will cause a concept $A$ to remain in a conjunct $Ex$ in $\mathcal{C}$, as defined via rule (5.18a).

**Upward refinement of range concepts.** A fact $app(genConceptInRole(Ex, r, A, B), \mathcal{C}, t)$ denotes the applicability of the generalisation of a concept $A$ to a concept $B$ when $A$ fills the range of a role $r$ in a role expression $RoleEx$ of $\mathcal{C}$ at step $t$ using $\gamma_C$:

$$\begin{aligned} app(genConceptInRole(RoleEx, r, A, B), \mathcal{C}_1, t) \leftarrow & \\ & hasRoleEx(\mathcal{C}_1, RoleEx, Depth, r, A, t), \\ & app(genConcept(RoleEx, A, B), \mathcal{C}_1, t), \\ & hasRoleExNotEq(\mathcal{C}_1, \mathcal{C}_2, RoleEx, t), Depth \leq k, \\ & \text{not } exec(genConceptInRole(RoleEx, \_, \_, \_), \mathcal{C}_2, t), dConcept(\mathcal{C}_2) \end{aligned} \tag{5.10}$$

The preconditions for generalising a concept filling the role of a role expression *RoleEx* are similar to the case of the upward refinement of an atomic concept: *RoleEx* involves a concept $A$ that is generalisable, the role expression is not equivalent in $\mathcal{C}_1$ and $\mathcal{C}_2$ (*hasRoleExNotEq*/4), and the concept to be generalised must not be under generalisation in $\mathcal{C}_2$. Please note how the maximum role depth of a concept $k$ controls the applicability of this rule.

The inertia rule for generalising a concept that fills the range of a role in $\mathcal{C}$ is:

$$\begin{aligned} noninertialGenConceptInRole(\mathcal{C}, RoleEx, r, A, t) \leftarrow \\ exec(genConceptInRole(RoleEx, r, A, \_), \mathcal{C}, t), \\ hasRoleEx(\mathcal{C}, RoleEx, \_, r, A, t) \end{aligned} \tag{5.11}$$

*noninertialGenConceptInRole* atoms will cause a concept $A$ to remain in the range of a role as defined via rule (5.18b).

**Upward refinement of roles.** A fact $app(genRole(RoleEx, r, s), \mathcal{C}, t)$ denotes the applicability of the generalisation of a role $r$ to a role $s$ in a role expression *RoleEx* of $\mathcal{C}$ at step $t$ using $\gamma_r$:

$$\begin{aligned} app(genRole(RoleEx, r, s), \mathcal{C}_1, t) \leftarrow \\ hasConjunct(\mathcal{C}_1, Ex, RoleEx, t), \\ hasRoleEx(\mathcal{C}_1, RoleEx, \_, r, A, t), \\ subRole(r, s), \\ roleInExpressionNotEq(\mathcal{C}_1, \mathcal{C}_2, RoleEx, r, t), \\ \text{not } exec(genRole(RoleEx, r, \_), \mathcal{C}_2, t), dConcept(\mathcal{C}_2) \end{aligned} \tag{5.12}$$

The main precondition for generalising a role $r$ contained in a role expression *RoleEx* is that $r$ has a parent role $s$ in the subsumption hierarchy defined by the axioms of the TBox. Other preconditions are that the role expression *RoleEx* is not equivalent in $\mathcal{C}_1$ and $\mathcal{C}_2$ (*roleInExpressionNotEq*/4) and is not being generalised in $\mathcal{C}_2$.

The inertia rule for generalising a role in a role expression is:

$$\begin{aligned} noninertialGenRole(\mathcal{C}, RoleEx, r, t) \leftarrow exec(genRole(RoleEx, r, \_), \mathcal{C}, t), \\ hasRoleEx(\mathcal{C}, RoleEx, \_, r, A, t) \end{aligned} \tag{5.13}$$

*noninertialGenRole* atoms will cause a role $r$ to remain in a role expression *RoleEx* in $\mathcal{C}$, as defined via rule (5.18b).

**Removal of a role.** A fact $app(rmRole(RoleEx, r, A), \mathcal{C}, t)$ denotes the applicability of the removal of a role $r$ from a role expression *RoleEx* of $\mathcal{C}$ at step $t$ using $\gamma_{r^-}$:

$$\begin{aligned} app(rmRole(RoleEx, r, A), \mathcal{C}_1, t) \leftarrow \\ hasConjunct(\mathcal{C}_1, Ex, RoleEx, t), \\ hasRoleEx(\mathcal{C}_1, RoleEx, \_, r, A, t), \\ \text{not } app(genRole(RoleEx, r, s), \mathcal{C}_1, t), \\ \text{not } app(genConceptInRole(RoleEx, r, A, \_), \mathcal{C}_1, t), \\ hasRoleExNotEq(\mathcal{C}_1, \mathcal{C}_2, RoleEx, t), \\ \text{not } exec(rmRole(RoleEx, r, \_), \mathcal{C}_2, t), dConcept(\mathcal{C}_2) \end{aligned} \tag{5.14}$$

Essentially, a role $r$ is removable from a role expression $RoleEx$ when neither itself nor the concept filling its range are generalisable. This is captured by the negated-by-failure predicates $app/3$. Other preconditions are that the role expression $RoleEx$ is not equivalent in $\mathcal{C}_1$ and $\mathcal{C}_2$ ($hasRoleExNotEq/4$) and is not being removed from $\mathcal{C}_2$.

The inertia rule for removing a role in a role expression is:

$$\begin{aligned} noninertialRmRole(\mathcal{C},Ex,RoleEx,r,A,t) \leftarrow\ & exec(rmRole(RoleEx,r,A),\mathcal{C},t), \\ & hasConjunct(\mathcal{C},Ex,RoleEx,t), \\ & hasRoleEx(\mathcal{C},RoleEx,\_,r,A,t) \end{aligned} \tag{5.15}$$

*noninertialRmRole* atoms will cause a role $r$ to remain in a role expression in $\mathcal{C}$, as defined via rules (5.18a-5.18b).

**Removal of a concept.** A fact $app(rmConcept(C,A),\mathcal{C},t)$ denotes the applicability of the removal of a concept $A$ from a conjunct $Ex$ of $\mathcal{C}$ at step $t$ using $\gamma_{C^-}$:

$$\begin{aligned} app(rmConcept(Ex,A),\mathcal{C}_1,t) \leftarrow\ & \\ & hasConjunct(\mathcal{C}_1,Ex,A,t), \\ & \text{not } app(genConcept(Ex,A,\_),\mathcal{C}_1,t), \\ & conjunctNotEq(\mathcal{C}_1,\mathcal{C}_2,A,t), \\ & \text{not } exec(rmConcept(Ex,A),\mathcal{C}_2,t),dConcept(\mathcal{C}_2) \end{aligned} \tag{5.16}$$

Essentially, a concept $A$ is removable from a conjunct $Ex$ when is not generalisable. This is captured by the negated-by-failure predicates $app/3$. Other preconditions are that the conjunct from where the concept will be removed is not equivalent in $\mathcal{C}_1$ and $\mathcal{C}_2$ ($conjunctNotEq/4$) and $A$ is not being removed from $\mathcal{C}_2$.

The inertia rule for removing a concept is:

$$\begin{aligned} noninertialRmConcept(\mathcal{C},Ex,A,t) \leftarrow\ & exec(rmConcept(Ex,A),\mathcal{C},t), \\ & hasConjunct(\mathcal{C},Ex,A,t) \end{aligned} \tag{5.17}$$

*noninertialRmConcept* atoms will cause a concept $A$ to remain in a conjunct $Ex$ in $\mathcal{C}$, as defined via rule (5.18a).

**Inertia.** The following rules state which concepts remain unchanged when they are inertial.

$$\begin{aligned} hasConjunct(\mathcal{C},C,A,t+1) \leftarrow\ & hasConjunct(\mathcal{C},C,A,t), \\ & \text{not } noninertialGenConcept(\mathcal{C},C,A,t), \\ & \text{not } noninertialRmRole(\mathcal{C},C,A,\_,\_,t), \\ & \text{not } noninertialRmConcept(\mathcal{C},C,A,t) \end{aligned} \tag{5.18a}$$

$$\begin{aligned} hasRoleEx(\mathcal{C},C,r,Depth,A,t+1) \leftarrow\ & \\ & hasRoleEx(\mathcal{C},C,r,Depth,A,t), \\ & \text{not } noninertialGenConceptInRole(\mathcal{C},C,r,A,t), \\ & \text{not } noninertialGenRole(\mathcal{C},C,r,A,t), \\ & \text{not } noninertialRmRole(\mathcal{C},\_,C,r,A,t) \end{aligned} \tag{5.18b}$$

**Effects.** Effect rules model how the knowledge changes when a concepts is generalised. The rule below shows an example of the effects of the generalisation of an atomic concept. Other two effect rules model the changes in the case of the generalisation of a role and of a concept in the range of a role.

$$\begin{aligned} hasConjunct(\mathcal{C},C,B,t+1) \leftarrow & \\ & hasConjunct(\mathcal{C},C,A,t), \\ & exec(genConcept(C,A,B),\mathcal{C},t) \end{aligned} \tag{5.19}$$

Additional rules handle the case in which the generalisation adds facts that model concept definitions (Eq. 5.6a-5.6e). In such a case, the number of roles $Depth$ can be increased. To this end, the precondition $Depth \leq k$ in Eq. 5.10 prevents the applicability of further generalisations of a concept filling the range of a role when $Depth$ reaches $k$, the maximum number of nested roles allowed.

**Checking the equivalence between generalisations.** As seen in the previous section, the upward refinement operator $\gamma$ is proper when those generalisations, which are equivalent to the concept being generalised, are discarded (see Eq. 4.4). To this end, during the generic space search, we discard these generalisations. The *clingo* solver allows one to interleave the solving capabilities of ASP with a procedural language such as Python. This allowed us to check the equivalence between two generalisations in an external Python function and return the result to the ASP program. The rule below shows an example of how an external function *isGenEq* can be called from our ASP program.

$$\begin{aligned} poss(genConcept(Ex,A,B),\mathcal{C}_1,t) \leftarrow & \\ & app(genConcept(Ex,A,B),\mathcal{C}_1,t), \\ & EQ \neq 1, EQ = @isGenEq(`genConcept',\mathcal{C},Ex,\_,A,B,t) \end{aligned} \tag{5.20}$$

The *isGenEq* function internally does two things. First, it builds the concept description $\mathcal{C}$ based on the current generalisation. Since the incremental ASP solving process is controlled by a Python script, the Python function contains all the generalisations of a concept. Second, it checks whether the generalisation at step $t$ is equivalent to the generalisation at step $t-1$. This is done by means of the *jcel* reasoner [36].[8] We test the equivalence between the current and the previous generalisation by checking the corresponding subsumptions. If the two generalisations are equivalent, then the function returns 1. In this case, the applicability of a generalisation operation is disabled by preventing the instantiation of the corresponding $poss/3$ predicate.

### 5.3 Upward refinement search

We use ASP for finding a generic space and the generalised versions of the concepts in the domain knowledge of an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$, which can lead to a blend. This is done by successively generalising the concepts in the domain knowledge by means of the upward operator steps we described in the previous subsection.

[8] The *jcel* is a modular rule-based reasoner for description logics of the EL family implemented in Java. It uses a rule-based completion algorithm in which a set of completion rules are successively applied to saturate data structures that are used to model $\mathcal{EL}$ axioms. The algorithm is based on the CEL's algorithm [5] but is generalised with a change propagation approach. It implements reasoning tasks such as *classification*, *consistency*, *satisfability*, and *entailment*. The main advantage of the *jcel* reasoner is that these tasks are computable in polynomial time.'

Given a concept description $C$ in an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$, the repetitive application of the generalisation operator types is a *refinement path*.

**Definition 12.** *Let $\mathcal{C}$ be a domain concept in an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$, let $\{\gamma_x^1, \ldots, \gamma_x^n\}$ be the set of generalisation steps for $\mathcal{C}$, $0 = t_1 < \cdots < t_n = n$ be refinement steps and $\gamma_x \in \{\gamma_A, \gamma_C, \gamma_r, \gamma_{r^-}, \gamma_{C^-}\}$ . The set of atoms $S = \{exec(\gamma_x^1, \mathcal{C}, t_1), \cdots, exec(\gamma_x^n, \mathcal{C}, t_n)\}$ is a refinement path of $\mathcal{C}$. A refinement path of $\mathcal{C}$ leads to the generalised concept $\mathcal{C}^n = \gamma_x^n(\cdots \gamma_x^2(\gamma_x^1(\mathcal{C})))$. We write $\mathcal{C}^j$ ($1 \leq j \leq n$) to denote the concept $\mathcal{C}$ after $j$ generalisation steps.*

Refinement paths are generated by means of a choice rule, that allows one or zero refinement operators per $\mathcal{C}$ at each step $t$. The only generalisations that are executed are those whose preconditions are satisfied. Refinement paths lead from the domain concepts to a generic space. A generic space is reached, if the generalised domain concepts are equal. A constraint ensures that the generic space is reached in all stable models. The ASP program generates one stable model for each combination of generalisation paths that lead to the generic space.

We should note at this point that the ASP implementation is sound and complete w.r.t. the upward refinement operator $\gamma'$. So, given two $\mathcal{EL}^{++}$ concepts $C_1$ and $C_2$, each stable model of the logic program $P$ —encoding the generic space search and the two concepts— contains the refinement paths $S_1$ and $S_2$ through which $C_1$ and $C_2$ can be generalised to a concept $G$ that is a generic space according to Definition 9. Proving this result can be done by induction over the structure of toASP and $P$, similar to the proof in [17, Appendix B]. On the other hand, if two concepts have a generic space $G$ by applying $\gamma'$, this generic space is found by $P$, thus, the implementation is complete. However, the ASP implementation is neither sound nor complete w.r.t. the $\mathcal{EL}^{++}$ semantics ($\sqsubseteq_{\mathcal{T}}$) since the operator is not complete (Theorem 1).

*Example 7.* Let us consider the SearchHardDisk and EditDocument concepts in Example 1 representing icons in the domain knowledge of the *ComputerIcon* ontology. Their refinement paths are:

$$
\begin{aligned}
S_{SearchHardDisk} = \{&exec(genConceptInRole(roleEx_{(6)}, hasMeaning, HardDrive,\\
&\qquad ObjectType), SearchHardDisk, 0),\\
&exec(genRole(roleEx_{(3)}, isAboveIn, isInSpatialRelation, subEx_{(4)}),\\
&\qquad SearchHardDisk, 1),\\
&exec(genConceptInRole(roleEx_{(4)}, hasMeaning, Search, Action),\\
&\qquad SearchHardDisk, 2)\}\\
S_{EditDocument} = \{&exec(genConceptInRole(roleEx_{(4)}, hasMeaning, Edit, Action),\\
&\qquad EditDocument, 0),\\
&exec(genRole(roleEx_{(3)}, isAboveInRight, isInSpatialRelation, subEx_{(4)}),\\
&\qquad EditDocument, 1),\\
&exec(genConceptInRole(roleEx_{(6)}, hasMeaning, Doc, ObjectType),\\
&\qquad EditDocument, 2)\}
\end{aligned}
$$

The refinement paths are parsed in order to translate the ASP encoding back to $\mathcal{EL}^{++}$ and apply the corresponding generalisation operators. The application of the refinement paths to the input

concepts lead to the generalised concepts and to their a generic space. It is easy to check that this corresponds to the generic space in Example 3.

The output the generalisation search is then passed to a blending algorithm in order to create and evaluate $\mathcal{EL}^{++}$ blended concepts, as described next.

## 6 Blending $\mathcal{EL}^{++}$ concepts

Conceptual blending by Fauconnier and Turner [21] is a cognitive theory that explains human creativity. According to this theory, humans create through a mental process that takes different mental spaces as input and combines them into a new mental space, called a *blend*. A blend is constructed by taking the commonalities among the input mental spaces into account, to form a so-called *generic space*, and by projecting the non-common structure of the input spaces in a selective way to the novel blended space. Since this theory focuses on the cognitive aspects of human creation, it is not a computational framework. It needs to be re-interpreted in a computational way, when one wants to use it in computational creativity.

In working towards this objective, we have characterised mental spaces in terms of $\mathcal{EL}^{++}$ concept descriptions and we have devised a generalisation algorithm to find the generic space between $\mathcal{EL}^{++}$ concepts. In this section, we provide an algorithm to find $\mathcal{EL}^{++}$ blended concepts. From a cognitive point view, conceptual blending involves the following aspects:

1. blend *generation*: it takes the generic space of two input spaces into account and combines their non-common structure in a selective way to a novel blended space;
2. blend *completion*: it constructs the *emergent structure* of a blend —a structure that is not directly copied from the inputs— by taking some background knowledge into account;
3. blend *evaluation*: it assesses the quality of a blend by means of certain optimality principles.

Our algorithm for blending $\mathcal{EL}^{++}$ concepts (Algorithm 1) implements these aspects as three phases, and re-interprets them in order to provide a computational account for conceptual blending. The implementation of the conceptual blending algorithm is available at: `https://bitbucket.org/rconfalonieri/ontolp-implementation`.

The blend generation is implemented according to the definition of an amalgam (Definition 4). To this end, first, a generic space is found by means of the ASP-based generalisation process described in the previous section. The method *generalise* finds different refinement paths of two (domain) $\mathcal{EL}^{++}$ concepts that lead to a generic space (Line 1). Then, a blend is created by computing the most general specialisation (MGS) of a pair of generalised concepts (Line 4). The MGS of two $\mathcal{EL}^{++}$ concepts corresponds to their conjunction.

Due to this combinational way of generating the blends, some of them might have already been found using some previous refinement paths, and they are simply not considered. Some other blends, on the other hand, may be not interesting. For instance, they might not have certain desirable properties.

In the algorithm, blend evaluation consists of two parts: a logical check and a heuristic function (Line 5 and 7).[9] The logical check discards those blends that do not satisfy certain properties.

[9] Blend evaluation is an open research topic in conceptual blending and it can be accomplished in different ways. For instance, evaluation could be achieved through an argumentative dialogue, in which users engage in order to decide

**Algorithm 1** Conceptual blending of $\mathcal{EL}^{++}$ concepts

**Input:**
- An $\mathcal{EL}^{++}$ TBox $\mathcal{T}$
- Two domain concepts $C_1$ and $C_2$
- A consequence requirement $CR$
- A maximum role depth $k$

**Output:** A ranked list of blended concepts $\mathcal{B}$

{$\langle C'_1, C'_2\rangle$ denotes a set of generalisations for $C_1$ and $C_2$ that lead to a generic space.}

1: **for all** $\langle C'_1, C'_2\rangle \leftarrow generalise(C_1, C_2, k)$ **do**
2:  **for** $C'_1 \in C'_1$ **do**
3:  **for** $C'_2 \in C'_2$ **do**
4:  $C_{am} \leftarrow MGS(C'_1, C'_2)$
5:  **if** $C_{am} \notin \mathcal{B}$ and $\{\mathcal{T} \cup C_{am}\}$ entails $CR$ **then**
6:  $C'_{am} \leftarrow completion(C_{am})$
7:  $rankBlend(C'_{am}, compactness(C'_{am}), \mathcal{B})$
8:  **end if**
9:  **end for**
10:  **end for**
11: **end for**
12: **return** $\mathcal{B}$

Desirable properties are modeled as an ontology consequence requirement *CR* that is given as input to the algorithm. For instance, a consequence requirement can ask that a blend should contain certain concepts and roles. For our purposes, it can require that a blended concepts contains a sign with meaning *search* above a sign with meaning *document*, which can be modeled in $\mathcal{EL}^{++}$ as $\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Search} \sqcap \exists \mathsf{isAboveIn}.\mathsf{Sign} \sqcap \exists \mathsf{hasMeaning}.\mathsf{Doc}$, To verify whether a consequence requirement is satisfied or not, the algorithm makes use of the *jcel* reasoner. Consequence satisfaction is achieved by checking whether the ontology in the TBox $\mathcal{T}$ and the new blended concept entail the consequence requirement (Line 5).[10]

Then, those blends that satisfy the consequence requirement are *completed*. In conceptual blending, completion refers to the "*background knowledge that one brings into a blend*" [21]. Clearly, in a computational setting, there can be different interpretations of what background knowledge stands for. In our implementation, we interpreted it as structural properties that a blend should have. In blending computer icons, we expect new blended icon concepts to be defined by one spatial relation between signs in which each sign has only one meaning relation.[11] To this end, completion is an operation that transforms the structure of a blend by taking this background knowledge into account. In particular, completion consists of a set of transformation rules that aggregate roles and concepts by taking the axioms in the TBox into account.

To implement completion, we specified a simple rewriting system using Maude [12], a system that supports rewriting logic specification and programming. The transformation rules that we used

which blend to keep and which one to discard. We refer the interested reader to [13] where a discussion about the use of Lakatosian reasoning to evaluate conceptual blends is presented.

[10] Consequence satisfaction can be checked by means of the 'entailment' option of the *jcel* reasoner as: `java -jar jcel.jar entailment ontology.owl -conclusion=conclusion.owl`, where `ontology.owl` is the ontology extended with the definition of a new blended concept $\mathsf{Blend}$ and `conclusion.owl` contains an axiom of the form $\mathsf{Blend} \sqsubseteq \mathsf{CR}$.

[11] This constraint is not expressable in $\mathcal{EL}^{++}$. We re-interpreted it as the background knowledge used to complete blended concepts.

for completing a blend are:

$$A \sqcap B \text{ is transformed to } A \text{ if } A \sqsubseteq_{\mathcal{T}} B \tag{6.1a}$$

$$\exists r.C \sqcap \exists s.C \text{ is transformed to } \exists r.(C \sqcap D) \text{ if } r \sqsubseteq_{\mathcal{T}} s \tag{6.1b}$$

Besides, we make use of concept definitions—equivalence axioms in the TBox—to rewrite a blend into a shorter equivalent form. It is worthy to notice that whilst this last rewriting preserves concept equivalence—therefore, it can be considered a simple instance of DL rewriting [3]—the above rules do not. Indeed, the rule in Eq. 6.1b is not invariant w.r.t. $\mathcal{EL}^{++}$ semantics, since it transforms a concept into a more specific one. Blends are completed before a heuristic function is applied (Line 6).

To decide which blends are better than others, the algorithm ranks them by means of a heuristic function (Line 7). The *compactness* heuristic counts the number of concepts and roles used in the definition of a blend $B$:

$$compactness(B) = \frac{1}{conceptsNr(B) + rolesNr(B)} \tag{6.2}$$

The algorithm considers as best blends those that have a higher compactness value. This heuristic can be considered as a computational interpretation of some of the optimality principles proposed by Fauconnier and Turner [21]. The *integration principle*, for instance, states that "*a blend must constitute a tightly integrated scene that can be manipulated as a unit*". The compactness of a blend captures the idea behind this principle in the sense that minimises the number of concepts and roles that are used to define a blend.

*Example 8.* Let us consider $C_1$ = SearchHardDisk, $C_2$ = EditDocument, $G$ in Example 3 and the generalisation steps in $P_{SearchHardDisk}$ and in $P_{EditDocument}$ of Example 7. Given a consequence requirement expressing that a blended concept should contain a sign with meaning *search* above a sign with meaning *document*, modeled in $\mathcal{EL}^{++}$ as Sign $\sqcap \exists$hasMeaning.Search $\sqcap \exists$isAboveIn.Sign $\sqcap \exists$hasMeaning.Doc, and the maximum role depth $k = 2$, the algorithm returns the following ranked blends:[12]

| Blend | Compactness |
|---|---|
| $MGS(C_1^1, C_2^2)$ | 0,33 |
| $MGS(C_1, C_2^1)$ | 0,2 |
| $MGS(C_1, C_2^2)$ | 0,16 |
| $MGS(C_1^1, C_2^1)$ | 0,14 |
| $MGS(C_1, C_2)$ | 0,13 |
| $MGS(C_1^1, C_2)$ | 0,1 |

Each blend is obtained by combining different generalisations of the input concepts $C_1$ and $C_2$. The concepts $C_1^1$ and $C_2^2$ correspond to the generalised concepts 'GenConcept1' and 'GenConcept2' in Figure 3 respectively. They are obtained by applying the generalisations steps from Example 7.

[12] Recalling Definition 12, in the table, $C_i^j$ stands for '$j$ generalisations have been applied to the concept $i$'. When $j$ is omitted, $C_i$ denotes the input concept $i$ with no generalisations.

$C_1^1$ is obtained from $C_1$ in one generalisation step by generalising the concept HardDrive (filling the role hasMeaning) to ObjectType. $C_2^2$ is obtained from $C_2$ in two generalisation steps by generalising the concept Edit (filling the role hasMeaning) to Action and the role isAboveRightIn to isInSpatialRelation. $MGS(C_1^1, C_2^2)$ is completed and elaborated into

$$\mathsf{MagnifyingGlass} \sqcap \exists \mathsf{isAbove.Document}$$

and its compactness value is 0,33. $MGS(C_1^1, C_2^2)$ is the best blend found by the algorithm. Other valid but less ranked blends are obtained by other combinations of generalised concepts.

## 7 Related Work

Conceptual blending in $\mathcal{EL}^{++}$ as described in this paper is a special case of the amalgam-based concept blending model described in [10], and implemented for CASL theories in [19] in order to invent cadences and chord progressions. This model has also been used to study the role of blending in mathematical invention [11]. This concept blending model, as the one presented here, is based on the notion of amalgam defined over a space of generalisations [37]. The space of generalisations is defined by refinement operators, that can be specialisation operators or generalisation operators, notions developed by the Inductive Logic Programming (ILP) community for inductive learning. These notions can be specified in any language where refinement operators define a generalisation space like ILP [30], description logics [40], or order-sorted feature terms [38].

Several approaches for generalising ontology concepts in the $\mathcal{EL}$ family exist in the DL and ILP literature.

On the one hand, in DL approaches, the LGG is defined in terms of a non-standard reasoning task over a TBox [1, 2, 6, 44, 45]. Generally speaking, since the LGG w.r.t. general TBoxes in the $\mathcal{EL}$ family does usually not exist, these approaches propose several solutions for computing it. For instance, Baader [1, 2] devises the exact conditions for the existence of the LGG for cyclic $\mathcal{EL}$-TBoxes based on graph-theoretic generalisations. Baader et al [6] propose an algorithm for computing good LGGs w.r.t. a background terminology. Turhan and Zarrieß [44], Zarrieß and Turhan [45] specify the conditions for the existence of the LGG for general $\mathcal{EL}$- and $\mathcal{EL}^+$-TBoxes based on canonical models. As already commented in the introduction, our work relates to these approaches, but it is different in spirit, since we do not need to find the LGG between (two) $\mathcal{EL}^{++}$ concepts for the kind of application we are developing.

Our work also relates to the problem of concept unification in $\mathcal{EL}$ [4]. However, we do not focus on finding the substitutions needed to make two $\mathcal{EL}^{++}$ concepts equivalent, but rather than on generalising them by taking the axioms in the TBox into account.

An approach in DL that does use refinement operators is [40], where the language chosen for representing the generalisation space is that of DL Conjunctive Queries. Here LGG between two inputs, translated to conjunctive queries, can be determined by searching over the generalisation space using downward specialisation operators.

On the other hand, studying the LGG in terms of generalisation and specialisation refinement operators has been used for order-sorted feature terms and Horn clauses in ILP. Anti-unification (or LGG) in order-sorted feature terms was studied in [38], which was conducive to later develop the notion of amalgam [37]. The notion of refinement operator, that originated in ILP, has been more

studied in the space of Horn clauses [30], but LGG in particular has not been a topic intensively pursued in the context of inductive learning in ILP.

Finally, other approaches that combine ASP for reasoning over DL ontologies worthy to be mentioned are [16, 39, 42].

## 8 Conclusion and Future Works

In this paper, we defined an upward refinement operator for generalising $\mathcal{EL}^{++}$ concepts for conceptual blending. The operator works by recursively traversing their descriptions. We discussed the properties of the refinement operator. We showed that the operator is locally finite, proper, but it is not complete (Propositions 2-3 and Theorem 1). We claimed, however, that completeness is not an essential property for our needs, since being able to find a generic space between two $\mathcal{EL}^{++}$ concepts, although not a LGG, is already a sufficient condition for conceptual blending.

We presented an implementation of the refinement operator in ASP. We showed how to model the description of $\mathcal{EL}^{++}$ concepts in ASP and to implement a search process for generalising the domain knowledge of an $\mathcal{EL}^{++}$ TBox. The stable models of the ASP program contain the generalisation steps needed to be applied in order to generalise two $\mathcal{EL}^{++}$ concepts until a generic space is reached. We embedded the ASP-based search process in an amalgamation process to implement an algorithm for conceptual blending. The algorithm creates new $\mathcal{EL}^{++}$ concepts by combining pairs of generalised $\mathcal{EL}^{++}$ concepts. The blends are logically evaluated and ranked by means of ontological consequence requirements and a heuristical function respectively. We exemplified our approach in the domain of computer icon design.

We envision some directions of future research. We aim at employing a richer DL, such as $\mathcal{SROIQ}$ [27] (the DL underlying the Web Ontology Language OWL 2[13]), in our conceptual blending framework. This will allow us to capture more complex concept descriptions and consequence requirements. By doing this, however, we will have to sacrifice efficiency, since the reasoning tasks in this logic are computational more expensive than in $\mathcal{EL}^{++}$. A possible way to find a tradeoff between expressivity and efficiency is to employ a richer DL only in one of the phases of our blending algorithm, e.g., either in the generation or in the evaluation phase. For instance, $\mathcal{SROIQ}$ could be employed in the generation phase (to this end, we will need to extend the generalisation operator), while the blend evaluation could be realised through argumentation [13]. On the contrary, we can keep $\mathcal{EL}^{++}$ in the generation phase and use $\mathcal{SROIQ}$ in the evaluation. These options are perfectly justifiable from the conceptual blending point of view, since the blend generation and evaluation are separate processes that can use different languages and techniques. This is also what usually happens in data mining approaches to computational creativity [43].

Another extension of the framework that we wish to explore is the blending of entire ontologies rather than just single concepts. Blending ontologies has already been explored in an ontological blending framework [26, 29], where blends are computed as *colimits* of algebraic specifications. In this framework, the blending process is not characterised in terms of amalgams, nor are input concepts generalised syntactically. Rather, the generic space is assumed to be given and mapped to

[13] `http://www.w3.org/TR/owl2-overview/`, accessed 04/12/2015

the input ontologies via theory interpretations. Therefore, the results of this paper can be extended and directly applied to this framework as a computational solution to creating generic spaces.

We consider the work of this paper to be a fundamental step towards the challenging task of defining and implementing a computational framework for concept invention that uses DLs as its formal underpinning language.

# Bibliography


[1] Baader F (2003) Computing the Least Common Subsumer in the Description Logic $\mathcal{EL}$ w.r.t. Terminological Cycles with Descriptive Semantics. In: Ganter B, de Moor A, Lex W (eds) Conceptual Structures for Knowledge Creation and Communication, Lecture Notes in Computer Science, vol 2746, Springer Berlin Heidelberg, pp 117–130

[2] Baader F (2005) A Graph-Theoretic Generalization of the Least Common Subsumer and the Most Specific Concept in the Description Logic $\mathcal{EL}$. In: Hromkovič J, Nagl M, Westfechtel B (eds) Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol 3353, Springer Berlin Heidelberg, pp 177–188

[3] Baader F, Küsters R (2006) Non-standard Inferences in Description Logics: The Story So Far. In: Gabbay DM, Goncharov SS, Zakharyaschev M (eds) Mathematical Problems from Applied Logic I, International Mathematical Series, vol 4, Springer New York, pp 1–75

[4] Baader F, Morawska B (2009) Rewriting Techniques and Applications: 20th International Conference, RTA 2009 Brasília, Brazil, June 29 - July 1, 2009 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, chap Unification in the Description Logic $\mathcal{EL}$, pp 350–364

[5] Baader F, Brandt S, Lutz C (2005) Pushing the EL Envelope. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 364–369

[6] Baader F, Sertkaya B, Turhan AY (2007) Computing the least common subsumer w.r.t. a background terminology. Journal of Applied Logic 5(3):392 – 420

[7] Baader F, Brandt S, Lutz C (2008) Pushing the EL Envelope Further. In: Clark K, Patel-Schneider PF (eds) In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions

[8] Baral C (2003) Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press

[9] Besold TR, Plaza E (2015) Generalize and Blend: Concept Blending Based on Generalization, Analogy, and Amalgams. In: Proceedings of the 6th International Conference on Computational Creativity, ICCC15

[10] Bou F, Eppe M, Plaza E, Schorlemmer M (2014) D2.1: Reasoning with Amalgams. Tech. rep., COINVENT Project, available at `http://www.coinvent-project.eu/fileadmin/publications/D2.1.pdf`

[11] Bou F, Schorlemmer M, Corneli J, Gomez-Ramirez D, Maclean E, Smail A, Pease A (2015) The role of blending in mathematical invention. In: Proceedings of the 6th International Conference on Computational Creativity, ICCC15

[12] Clavel M, Durán F, Eker S, Lincoln P, Martí-Oliet N, Meseguer J, Talcott C (2003) The Maude 2.0 System. In: Nieuwenhuis R (ed) Rewriting Techniques and Applications (RTA 2003), Springer-Verlag, no. 2706 in Lecture Notes in Computer Science, pp 76–87

[13] Confalonieri R, Corneli J, Pease A, Plaza E, Schorlemmer M (2015) Using Argumentation to Evaluate Concept Blends in Combinatorial Creativity. In: Proceedings of the 6th International Conference on Computational Creativity, ICCC15

[14] Confalonieri R, Eppe M, Schorlemmer M, Kutz O, Peñaloza R, Plaza E (2015) Upward Refinement for Conceptual Blending in Description Logic —An ASP-based Approach and Case Study in $\mathcal{EL}^{++}$. In: Proceedings of 1st International workshop of Ontologies and Logic Programming for Query Answering, ONTOLP 2015, co-located with IJCAI-2015

[15] Cornet R, de Keizer N (2008) Forty years of SNOMED: a literature review. BMC medical informatics and decision making 8 Suppl 1
[16] Eiter T, Ianni G, Lukasiewicz T, Schindlauer R, Tompits H (2008) Combining answer set programming with description logics for the semantic web. Artificial Intelligence 172(12–13):1495–1539
[17] Eppe M, Bhatt M (2015) Approximate Postdictive Reasoning with Answer Set Programming. Journal of Applied Logic 13(4, Part 3):676–719
[18] Eppe M, Bhatt M, Dylla F (2013) Approximate epistemic planning with postdiction as answer-set programming. In: Cabalar P, Son TC (eds) Logic Programming and Nonmonotonic Reasoning: 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 290–303
[19] Eppe M, Confalonieri R, Maclean E, Kaliakatsos-Papakostas MA, Cambouropoulos E, Schorlemmer WM, Codescu M, Kühnberger K (2015) Computational Invention of Cadences and Chord Progressions by Conceptual Chord-Blending. In: Yang Q, Wooldridge M (eds) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, pp 2445–2451
[20] Eppe M, Maclean E, Confalonieri R, Kutz O, Schorlemmer WM, Plaza E (2015) ASP, Amalgamation, and the Conceptual Blending Workflow. In: Calimeri F, Ianni G, Truszczynski M (eds) Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings, pp 309–316
[21] Fauconnier G, Turner M (2002) The Way We Think: Conceptual Blending And The Mind's Hidden Complexities. Basic Books
[22] Gebser M, Kaminski R, Kaufmann B, Schaub T (2014) Clingo = ASP + control: Preliminary report. CoRR abs/1405.3694
[23] Gebser M, Kaminski R, Kaufmann B, Lindauer M, Ostrowski M, Romero J, Schaub T, Thiele S (2015) Potassco User Guide 2.0. Tech. rep., University of Potsdam
[24] Gelfond M, Kahl Y (2014) Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach. Cambridge University Press, New York, NY, USA
[25] Gelfond M, Lifschitz V (1988) The stable model semantics for logic programming. In: Proceedings of the Fifth International Conference on Logic Programming, (ICLP'88), The MIT Press, pp 1070–1080
[26] Hois J, Kutz O, Mossakowski T, Bateman J (2010) Towards ontological blending. In: Dicheva D, Dochev D (eds) Artificial Intelligence: Methodology, Systems, and Applications, Lecture Notes in Computer Science, vol 6304, Springer Berlin Heidelberg, pp 263–264
[27] Horrocks I, Kutz O, Sattler U (2006) The Even More Irresistible SROIQ. In: Doherty P, Mylopoulos J, Welty CA (eds) Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006, AAAI Press, pp 57–67
[28] Kowalski R (1974) Predicate Logic as Programming Language. In: Proceedings of International Federation for Information Processing, pp 569– 574
[29] Kutz O, Bateman J, Neuhaus F, Mossakowski T, Bhatt M (2014) E pluribus unum: Formalisation, Use-Cases, and Computational Support for Conceptual Blending. In: Computational Creativity Research: Towards Creative Machines, Thinking Machines, Atlantis/Springer
[30] van der Laag PR, Nienhuys-Cheng SH (1998) Completeness and properness of refinement operators in inductive logic programming. The Journal of Logic Programming 34(3):201 – 225

[31] Lee J, Palla R (2012) Reformulating the Situation Calculus and the Event Calculus in the General Theory of Stable Models and in Answer Set Programming. Journal of Artificial Intelligence Research 43:571–620

[32] Lehmann J, Haase C (2010) Ideal Downward Refinement in the EL Description Logic. In: Proc. of the 19th Int. Conf. on Inductive Logic Programming, Springer-Verlag, Berlin, Heidelberg, ILP'09, pp 73–87

[33] Lehmann J, Hitzler P (2010) Concept learning in description logics using refinement operators. Machine Learning 78(1-2):203–250

[34] Ma J, Miller R, Morgenstern L, Patkos T (2013) An Epistemic Event Calculus for ASP-based Reasoning About Knowledge of the Past, Present and Future. In: International Conference on Logic for Programming, Artificial Intelligence and Reasoning

[35] McCarthy J (1986) Applications of circumscription to formalizing common-sense knowledge. Artificial Intelligence 28(1):89–116

[36] Mendez J (2012) jcel: A Modular Rule-based Reasoner. In Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012) 858

[37] Ontañón S, Plaza E (2010) Amalgams: A Formal Approach for Combining Multiple Case Solutions. In: Bichindaritz I, Montani S (eds) Proceedings of the International Conference on Case Base Reasoning, Springer, Lecture Notes in Computer Science, vol 6176, pp 257–271

[38] Ontañón S, Plaza E (2012) Similarity measures over refinement graphs. Machine Learning Journal 87(1):57–92

[39] Ricca F, Gallucci L, Schindlauer R, Dell'Armi T, Grasso G, Leone N (2009) OntoDLV: An ASP-based System for Enterprise Ontologies. Journal of Logic and Computation 19(4):643–670

[40] Sánchez-Ruiz A, Ontañón S, González-Calero P, Plaza E (2013) Refinement-Based Similarity Measure over DL Conjunctive Queries. In: Delany S, Ontañón S (eds) Case-Based Reasoning Research and Development, Lecture Notes in Computer Science, vol 7969, Springer Berlin, pp 270–284

[41] Spackman K, Campbell K, Cote R (1997) SNOMED RT: A reference terminology for health care. Journal of the American Medical Informatics Association

[42] Swift T (2004) Deduction in Ontologies via ASP. In: Lifschitz V, Niemelä I (eds) Logic Programming and Nonmonotonic Reasoning, Lecture Notes in Computer Science, vol 2923, Springer Berlin, pp 275–288

[43] Toivonen H, Gross O (2015) Data mining and machine learning in computational creativity. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5(6):265–275

[44] Turhan A, Zarrieß B (2013) Computing the lcs w.r.t. general $\mathcal{EL}^+$-TBoxes. In: Proceedings of the 26th International Workshop on Description Logics, pp 477–488

[45] Zarrieß B, Turhan AY (2013) Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes. In: Proceedings of the 23th International Joint Conference on Artificial Intelligence, AAAI Press, IJCAI '13, pp 1191–1197