Conceptual Blending in $\mathcal{EL}^{++}$

Roberto Confalonieri[1], Marco Schorlemmer[1], Oliver Kutz[2], Rafael Peñaloza[2], Enric Plaza[1], and Manfred Eppe[3]

[1] IIIA-CSIC, Spain,
{confalonieri,marco,enric}@iiia.csic.es
[2] Free University of Bozen-Bolzano, Italy,
{oliver.kutz,rafael.penaloza}@unibz.it
[3] International Computer Science Institute, Berkeley, USA,
eppe@icsi.berkeley.edu

Abstract. The cognitive theory of conceptual blending models human creativity as a mental process that combines two mental spaces into a new mental space, called a blend. According to this theory, 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 their non-common structure in a selective way to the novel blended space. In this paper, we apply this idea to blend input spaces modeled as complex $\mathcal{EL}^{++}$ concepts. To construct the generic space of two $\mathcal{EL}^{++}$ concepts, these need to be generalised in a controlled manner. For this, we propose an upward refinement operator that is used for finding common generalisations of $\mathcal{EL}^{++}$ concepts.

1 Introduction

The 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* [13], a cognitive theory that inspired several algorithms and methodologies in computational creativity research [24, 26, 29].

A key problem in computational approaches to conceptual blending is that the combination of two concepts to be blended 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 states the shared structure of both concepts.

This paper addresses the formalisation of an operator for generalising input spaces modeled as $\mathcal{EL}^{++}$ concepts [3, 4]. 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 [2,6,27], approaches in ILP are concerned on learning DL descriptions from examples [20]. In both cases, however, finding an LGG is a challenging task. Its computability and existence depend on the type of DL adopted and on the assumptions made over the Tbox.

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 (although desirable) to find a *generic space* that is also an 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 a generalisation 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 [28].[4]

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 characterised by the notion of amalgams [7, 22] in order to create new $\mathcal{EL}^{++}$ concepts. Section 4 proposes the formalisation of a generalisation refinement operator for generalising $\mathcal{EL}^{++}$ concepts. In Section 5, we describe how the operator can be implemented in Answer Set Programming (ASP) [15] in order to find a generic space between $\mathcal{EL}^{++}$ concepts. Section 6 outlines several works that relate to ours from different perspectives, before concluding and providing a vision for future work.

2 Background

In this section we introduce the basic notions that will be used throughout the paper. After presenting the description logic $\mathcal{EL}^{++}$, we introduce refinement operators.

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

In DLs, concept and role descriptions are defined inductively by means of concept and role constructors over the sets N_C of concept names, N_R of role names, and 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 [3,4].

[4] 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.

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

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

A knowledge base 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 *subsumed* by D w.r.t. $\mathcal{T}$, denoted as $C \sqsubseteq_{\mathcal{T}} D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. C is *strictly subsumed* by D w.r.t. $\mathcal{T}$ ($C \sqsubset_{\mathcal{T}} D$) iff $C \sqsubseteq_{\mathcal{T}} D$ but $D \not\sqsubseteq_{\mathcal{T}} C$. 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 *subsumed* by s w.r.t. $\mathcal{T}$, denoted as $r \sqsubseteq_{\mathcal{T}} s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. The role r is *strictly subsumed* by s w.r.t. $\mathcal{T}$ iff $r \sqsubseteq_{\mathcal{T}} s$ but $s \not\sqsubseteq_{\mathcal{T}} r$.

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.

Formally 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.[5]

Definition 1. *A generalisation refinement operator γ 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 γ can be classified according to some desirable properties [28]. We say that γ 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 γ in zero or a finite number of steps).

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 [19] by taking $\langle \mathcal{S}, \preceq \rangle$ as the set of $\mathcal{EL}$ concept descriptions ordered under $\sqsubseteq_\mathcal{T}$. In this paper, we define a generalisation refinement operator for $\mathcal{EL}^{++}$ and study its properties.

3 Amalgam-based Conceptual Blending of $\mathcal{EL}^{++}$ Concepts

The process of conceptual blending can be characterised by the notion of amalgam [7, 22]. 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 $\mathcal{T}$. 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, an MGS is a description that has all the information from both original descriptions C_1 and C_2, while an LGG contains what is common to them.[6]

[5] A deeper analysis of refinement operators can be found in [28].

[6] In [22], the LGG and MGS of two concept descriptions are also known as their anti-unification and unification respectively.

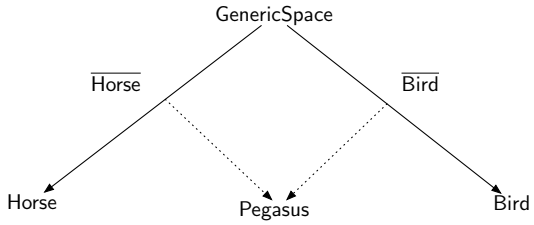

Fig. 1: A diagram of an amalgam Pegasus from descriptions Horse and Bird and their respective generalisations $\overline{\mathsf{Horse}}$ and $\overline{\mathsf{Bird}}$. Arrows indicate the subsumption of the target by the source of the arrow.

An *amalgam* or *blend* of two descriptions is a new description that contains *parts from these original descriptions*. For the purposes of this paper we can define an amalgam of two descriptions as follows.

Definition 2 (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 LGG C_{lgg}) if there exist two descriptions $\overline{C_1}$ and $\overline{C_2}$ such that: $C_1 \sqsubseteq_\mathcal{T} \overline{C_1} \sqsubseteq_\mathcal{T} C_{lgg}$, $C_2 \sqsubseteq_\mathcal{T} \overline{C_2} \sqsubseteq_\mathcal{T} C_{lgg}$, and C_{am} is an MGS of $\overline{C_1}$ and $\overline{C_2}$.*

This definition is illustrated in Figure 1 by means of a typical blend example: Pegasus, the winged divine stallion. From a conceptual blending point of view, Pegasus is a blend between a horse and a bird, maintaining pretty much the horse characteristics but adding the bird-like features such as the wings and the ability to fly. A *horse* and a *bird* can be described by concepts having different types of *clade*, some specific *body-parts* and *abilities*. For instance, a horse is a *mammal*, with a *torso* and *legs*, and with the *ability* to *walk* and to *trot*. A stereotypical characterisation (concept definition) of a horse and a bird modeled in $\mathcal{EL}^{++}$ is shown below.

$$\begin{aligned}\mathsf{Horse} \equiv\ & \mathsf{Mammal} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Torso} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Legs} \sqcap \\ & \exists \mathsf{hasAbility}.\mathsf{Walk} \sqcap \exists \mathsf{hasAbility}.\mathsf{Trot} \\ \mathsf{Bird} \equiv\ & \mathsf{Avialae} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Torso} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Legs} \sqcap \\ & \exists \mathsf{hasBodyPart}.\mathsf{Wings} \sqcap \exists \mathsf{hasAbility}.\mathsf{LayEggs} \sqcap \exists \mathsf{hasAbility}.\mathsf{Fly}\end{aligned}$$

The combination of these concepts violates the common sense knowledge that mammals do not generally lay eggs and that avialae do not trot.[7] Therefore, these abilities need to be generalised in a controlled manner before these concepts can be blended. The common descriptions between a horse and a bird—a clade with body-parts torso and legs—is a lower bound in the space of generalisations that need to be explored in order to generalise these concepts and to blend them into Pegasus. Then, a generalised version of the bird concepts is:

$$\begin{aligned}\overline{\mathsf{Bird}} \equiv\ & \mathsf{Clade} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Torso} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Legs} \sqcap \\ & \exists \mathsf{hasBodyPart}.\mathsf{Wings} \sqcap \exists \mathsf{hasAbility}.\mathsf{Fly}\end{aligned}$$

[7] This common sense knowledge can be modeled in $\mathcal{EL}^{++}$ by means of two axioms: $\mathsf{Mammals} \sqcap \exists \mathsf{hasAbility}.\mathsf{LayEggs} \sqsubseteq \bot$ and $\mathsf{Avialae} \sqcap \exists \mathsf{hasAbility}.\mathsf{Trot} \sqsubseteq \bot$. For the sake of this example, we do not consider the case of the platypus, an egg-laying mammal.

When we blend $\overline{\mathsf{Bird}}$ with Horse, we obtain a concept describing Pegasus. Please notice that in this case we can use a special case of amalgam (called asymmetric amalgam), in which $\overline{\mathsf{Horse}}$ and Horse coincide.

$$\begin{aligned}\mathsf{Pegasus} \equiv\ & \mathsf{Mammal} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Torso} \sqcap \exists \mathsf{hasBodyPart}.\mathsf{Legs} \sqcap \\ & \exists \mathsf{hasBodyPart}.\mathsf{Wings} \sqcap \exists \mathsf{hasAbility}.\mathsf{Walk} \sqcap \exists \mathsf{hasAbility}.\mathsf{Trot} \sqcap \\ & \exists \mathsf{hasAbility}.\mathsf{Fly}\end{aligned}$$

In the next section, we define a generalisation refinement operator that allows us to find generalisations of $\mathcal{EL}^{++}$ concept descriptions needed for computing the amalgams as described above.

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.[8] In order to define a generalisation refinement operator for $\mathcal{EL}^{++}$, we need some auxiliary definitions.

Definition 3. *Let $\mathcal{T}$ be an $\mathcal{EL}^{++}$ TBox. The set of* subconcepts *of $\mathcal{T}$ is given as*

$$\mathsf{sub}(\mathcal{T}) = \{\top, \bot\} \cup \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(C) \cup \mathsf{sub}(D) \tag{1}$$

where sub *is inductively defined over the structure of concept descriptions.*

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.

Definition 4. *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:*

$$\begin{aligned}\mathsf{UpCov}(A) := \{C \in \mathsf{sub}(\mathcal{T}) \mid\ & A \sqsubseteq_{\mathcal{T}} C \\ & \textit{and there is no } C' \in \mathsf{sub}(\mathcal{T}) \textit{ such that } A \sqsubset_{\mathcal{T}} C' \sqsubset_{\mathcal{T}} C\}\end{aligned} \tag{2}$$

$$\begin{aligned}\mathsf{UpCov}(r) := \{s \in N_R \mid\ & r \sqsubseteq_{\mathcal{T}} s \\ & \textit{and there is no } s' \in N_R \textit{ such that } r \sqsubset_{\mathcal{T}} s' \sqsubset_{\mathcal{T}} s\}\end{aligned} \tag{3}$$

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

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

Definition 5. *Let $\mathcal{T}$ be an $\mathcal{EL}^{++}$ TBox. We define the* generalisation refinement operator *γ inductively over the structure of concept descriptions as follows:*

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

where γ_r and γ_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 γ, $\mathcal{EL}^{++}$ concepts are related by refinement paths as described next.

Definition 6. *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 γ 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 γ in a finite number of steps.*

That γ is indeed a generalisation refinement operator as expressed by Definition 1 can be proven by applying structural induction on $\mathcal{EL}^{++}$ concepts [10].

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

Our definition of UpCov for basic concepts and roles only considers the set of sub-concepts present in a TBox $\mathcal{T}$. This guarantees that γ is locally finite, since at each generalisation step, the set of possible generalisations is finite.

Proposition 2. *The generalisation refinement operator γ is locally finite.*

This proposition can be proven by showing that for every $\mathcal{EL}^{++}$ concept C, $\gamma(C)$ is finite by induction on the structure of C [10].

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 γ is proper.

Example 1. Let $\mathcal{T} := \{A \sqsubseteq B\}$. Then, following Definition 5, 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, γ is not proper.

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

$$\gamma'(C) := \gamma(C) \backslash \{D \in \gamma(C) \text{ such that } D \equiv_{\mathcal{T}} C\} \tag{4}$$

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

Proposition 3. *The generalisation refinement operator γ' is proper.*

The repetitive application of the generalisation refinement operator allows 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 7. *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 2. Let us consider the Horse and Bird concepts. Their generic space is Clade$\sqcap$ $\exists$hasBodyPart.Torso $\sqcap$ $\exists$hasBodyPart.Legs and is obtained as follows. In the Horse concept, Mammal is generalised to Clade and $\exists$hasBodyPart.Trot and $\exists$hasAbility.Walk are removed. In the Bird concept, Avialae is generalised to Clade and the relations $\exists$hasBodyPart.Wings and $\exists$hasAbility.LayEggs are removed.

Unfortunately, due to the fact that the upward cover set we defined only takes subconcepts already present in the TBox into account, neither γ nor its refinement γ' are complete; that is, these operators may fail to compute some of the generalisations of a given $\mathcal{EL}^{++}$ concept w.r.t. $\sqsubseteq_{\mathcal{T}}$.

Example 3. 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 (and less general) 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 [10].

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

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 compute a generic space of a set of $\mathcal{EL}^{++}$ concepts. Indeed, as the following example shows, γ' can produce an infinite chain of generalisations.

Example 4. 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 γ'.

Not finding a least general 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, 27, 30]. 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 8. *The role depth of an $\mathcal{EL}^{++}$ concept description C is defined as the maximum number of nested (existential) quantifiers in C:*

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

Based on the role depth of a concept we modify the definition of the generalisation operator γ' to take a fixed constant $k \in \mathbb{N}_{>0}$ of nested quantifiers into account. More precisely, let γ'_k be defined as γ', 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 (trivial) generic space between $\mathcal{EL}^{++}$ concepts can always be found.

Definition 9. *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 $C_1, \ldots, C_n$.*

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 Implementation

In [10], we describe an algorithm implementing the cognitive theory of conceptual blending by Fauconnier & Turner [13] in which the input (mental) spaces are modeled in terms of $\mathcal{EL}^{++}$ concept descriptions.

The conceptual blending of $\mathcal{EL}^{++}$ concepts is implemented as an amalgam-based workflow consisting of two phases: blend generation and blend evaluation. The first phase finds a generic space between $\mathcal{EL}^{++}$ concepts and creates new blended concepts by taking their generalisations into account. The second phase evaluates the blends by checking if they are consistent or satisfy certain properties.

The generic space search is implemented in Answer Set Programming (ASP) [15], a well-known declarative programming paradigm to solve non-monotonic search problems. A domain-independent ASP program generalises $\mathcal{EL}^{++}$ concepts in a step-wise transition process. To this end, we consider each step of the generalisation refinement

operator in Definition 5 as an action. The domain-independent ASP program is instantiated with domain knowledge. The domain knowledge is obtained by translating the $\mathcal{EL}^{++}$ TBox into ASP facts and predicates. $\mathcal{EL}^{++}$ concepts are generalised until their descriptions are equal. The stable models of the ASP program contain the generalisation steps to be applied in order to generalise the $\mathcal{EL}^{++}$ concepts until a generic space is reached. Each stable model is used to generate a set of blends.

We use the ASP solver *clingo4* [14] as main reasoning engine, which allows us not only to implement the search in an incremental manner, but also to use external programs via a Python interface. In our case, we control the amalgam-based workflow by a Python script. The script also calls the *jcel* reasoner [21] as an external tool in order to check that the generalisations obtained at a given step are not equivalent to the concept being generalised —thus guaranteeing properness of the generalisation refinement operator— and to evaluate the blends.

Blend evaluation essentially consists of a logical check and a ranking function. The logical check discards those blends that are not consistent or does not satisfy some consequence requirements. Consequence satisfaction and consistency checking are realised through the *jcel* reasoner. A heuristic is used to rank the blends. Further details about the implementation can be found in [10].

6 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 [8,26], and implemented for CASL theories in [11] in order to blend chords in cadences. This model has also been used to study the role of blending in mathematical invention [9, 12]. This concept blending model, as the one presented here, is based on the notion of amalgam defined over a space of generalisations [22]. The space of generalisations is defined by refinement operators, that can be specialisation operators or generalisation operators, notions developed by the ILP community for inductive learning. These notions can be specified in any language where refinement operators define a generalisation space like ILP [28], description logics [25], or order-sorted feature terms [23].

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 [1,2,6,27,30]. 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. In [6], the authors propose an algorithm for computing good LGGs w.r.t. a background terminology. In [27, 30], some conditions for the existence of the LGG for general TBoxes based on canonical models are shown. As already pointed out in the introduction, our work relates to these approaches, but it is different in spirit.

Our work also seems to be related to the problem of concept unification in $\mathcal{EL}$ [5] that focuses on finding the substitutions needed to make two $\mathcal{EL}^{++}$ concepts equivalent. In a certain sense, we also try to make two concepts equivalent, but we generalise them by taking the axioms in the TBox into account.

An approach in DL that uses refinement operators is [25], 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 [23], which was conducive to later develop the notion of amalgam [22]. The notion of refinement operator has been more studied in the space of Horn clauses [28], but LGG in particular has not been a topic intensively pursued in the context of inductive learning in ILP.

7 Conclusions and Future Work

In this paper we defined a generalisation refinement operator for generalising $\mathcal{EL}^{++}$ concepts for conceptual blending. The operator works by recursively traversing their descriptions. We discussed the properties of the 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 an LGG, is already a sufficient condition for conceptual blending.

We described how the generalisation refinement operator can be implemented in ASP. Essentially, ASP is used to find the generalisations needed to be applied in order to generalise two $\mathcal{EL}^{++}$ concepts until a generic space is reached. The ASP-based search process is embedded in an amalgam-based algorithm that creates new $\mathcal{EL}^{++}$ concepts by combining pair of generalised $\mathcal{EL}^{++}$ concepts. All the details can be found in our technical report [10].

We envision some directions of future research. We aim at employing a richer DL, such as $\mathcal{SROIQ}$ [17] in our conceptual blending framework. This will allow us to capture more complex concept descriptions and consequence requirements. We will also study ways of prioritising some portions of the concept descriptions as fundamental properties that should not be modified during blending.

Another extension of the framework that we wish to explore is the blending of ontologies rather than concepts. Blending ontologies has already been explored in an ontological blending framework [16, 18], where blends are computed as *colimits* of algebraic specifications. In this framework, the blending process is not characterised in terms of amalgams, the input concepts are not generalised, and the generic space is assumed to be given. Therefore, the results of this paper can be extended and applied in this framework.

We consider the work of this paper to be a fundamental step towards the challenging task of defining and implementing a computational creativity framework based on conceptual blending that employs DL as its formal underpinning language.

Acknowledgements

We thank anonymous reviewers for their valuable comments. This work is partially supported by the COINVENT project (FET-Open grant number: 611553).

References

1. F. Baader. Computing the Least Common Subsumer in the Description Logic $\mathcal{EL}$ w.r.t. Terminological Cycles with Descriptive Semantics. In B. Ganter, A. de Moor, and W. Lex, editors, *Conceptual Structures for Knowledge Creation and Communication*, volume 2746 of *Lecture Notes in Computer Science*, pages 117–130. Springer Berlin Heidelberg, 2003.
2. F. Baader. A Graph-Theoretic Generalization of the Least Common Subsumer and the Most Specific Concept in the Description Logic $\mathcal{EL}$. In J. Hromkovič, M. Nagl, and B. Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science*, volume 3353 of *Lecture Notes in Computer Science*, pages 177–188. Springer Berlin Heidelberg, 2005.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the EL Envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 364–369, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.
4. F. Baader, S. Brandt, and C. Lutz. Pushing the EL Envelope Further. In K. Clark and P. F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.
5. F. Baader and B. Morawska. Rewriting Techniques and Applications: 20th International Conference, RTA 2009 Brasília, Brazil, June 29 - July 1, 2009 Proceedings. chapter Unification in the Description Logic $\mathcal{EL}$, pages 350–364. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
6. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3):392 – 420, 2007.
7. T. R. Besold and E. Plaza. Generalize and Blend: Concept Blending Based on Generalization, Analogy, and Amalgams. In *Proceedings of the 6th International Conference on Computational Creativity, ICCC15*, 2015.
8. F. Bou, M. Eppe, E. Plaza, and M. Schorlemmer. D2.1: Reasoning with Amalgams. Technical report, COINVENT Project, October 2014. Available at `http://www.coinvent-project.eu/fileadmin/publications/D2.1.pdf`.
9. F. Bou, M. Schorlemmer, J. Corneli, D. Gomez-Ramirez, E. Maclean, A. Smail, and A. Pease. The role of blending in mathematical invention. In *Proceedings of the 6th International Conference on Computational Creativity, ICCC15*, 2015.
10. R. Confalonieri, M. Eppe, M. Schorlemmer, O. Kutz, R. Peñaloza, and E. Plaza. Upward Refinement Operators for Conceptual Blending in $\mathcal{EL}^{++}$. Technical Report TR-IIIA-2016-01, Artificial Intelligence Research Institute (IIIA-CSIC), 2016. Available at `http://www.iiia.csic.es/files/pdfs/TR-IIIA-2016-01.pdf`.
11. M. Eppe, R. Confalonieri, E. Maclean, M. A. Kaliakatsos-Papakostas, E. Cambouropoulos, W. M. Schorlemmer, M. Codescu, and K. Kühnberger. Computational Invention of Cadences and Chord Progressions by Conceptual Chord-Blending. In Q. Yang and M. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2445–2451. AAAI Press, 2015.
12. M. Eppe, E. Maclean, R. Confalonieri, O. Kutz, W. M. Schorlemmer, and E. Plaza. ASP, Amalgamation, and the Conceptual Blending Workflow. In F. Calimeri, G. Ianni, and M. Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, pages 309–316, 2015.
13. G. Fauconnier and M. Turner. *The Way We Think: Conceptual Blending And The Mind's Hidden Complexities*. Basic Books, 2002.
14. M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014.

15. M. Gelfond and Y. Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, New York, NY, USA, 2014.
16. J. Hois, O. Kutz, T. Mossakowski, and J. Bateman. Towards ontological blending. In D. Dicheva and D. Dochev, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, volume 6304 of *Lecture Notes in Computer Science*, pages 263–264. Springer Berlin Heidelberg, 2010.
17. I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 57–67. AAAI Press, 2006.
18. O. Kutz, J. Bateman, F. Neuhaus, T. Mossakowski, and M. Bhatt. E pluribus unum: Formalisation, Use-Cases, and Computational Support for Conceptual Blending. In *Computational Creativity Research: Towards Creative Machines*, Thinking Machines. Atlantis/Springer, 2014.
19. J. Lehmann and C. Haase. Ideal Downward Refinement in the EL Description Logic. In *Proc. of the 19th Int. Conf. on Inductive Logic Programming*, ILP'09, pages 73–87, Berlin, Heidelberg, 2010. Springer-Verlag.
20. J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.
21. J. Mendez. jcel: A Modular Rule-based Reasoner. *In Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*, 858, 2012.
22. S. Ontañón and E. Plaza. Amalgams: A Formal Approach for Combining Multiple Case Solutions. In I. Bichindaritz and S. Montani, editors, *Proceedings of the International Conference on Case Base Reasoning*, volume 6176 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2010.
23. S. Ontañón and E. Plaza. Similarity measures over refinement graphs. *Machine Learning Journal*, 87(1):57–92, 2012.
24. F. C. Pereira. *Creativity and Artificial Intelligence: A Conceptual Blending Approach*. Mouton de Gruyter, 2007.
25. A. Sánchez-Ruiz, S. Ontañón, P. González-Calero, and E. Plaza. Refinement-Based Similarity Measure over DL Conjunctive Queries. In S. Delany and S. Ontañón, editors, *Case-Based Reasoning Research and Development*, volume 7969 of *Lecture Notes in Computer Science*, pages 270–284. Springer Berlin, 2013.
26. M. Schorlemmer, A. Smaill, K.-U. Kühnberger, O. Kutz, S. Colton, E. Cambouropoulos, and A. Pease. COINVENT: Towards a Computational Concept Invention Theory. In *Proc. of the Fifth Int. Conf. on Computational Creativity (ICCC 2014)*. Ljubljana, Slovenia, 2014.
27. A. Turhan and B. Zarrieß. Computing the lcs w.r.t. general $\mathcal{EL}^+$-TBoxes. In *Proceedings of the 26th International Workshop on Description Logics*, pages 477–488, 2013.
28. P. R. van der Laag and S.-H. Nienhuys-Cheng. Completeness and properness of refinement operators in inductive logic programming. *The Journal of Logic Programming*, 34(3):201 – 225, 1998.
29. T. Veale and D. O. Donoghue. Computation and blending. *Cognitive Linguistics*, 11(3-4):253–282, 2000.
30. B. Zarrieß and A.-Y. Turhan. Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1191–1197. AAAI Press, 2013.