Chapter 2
Computational Aspects of Concept Invention*

Roberto Confalonieri, Enric Plaza, and Marco Schorlemmer

Abstract In this chapter, we present a computational framework that models concept invention. The framework is based on and extends conceptual blending. Apart from the blending mechanism modeling the creation of new concepts, the framework considers two extra dimensions, namely, origin and destination. For the former, we describe how a Rich Background supports the discovery of input concepts to be blended. For the latter, we show how arguments, promoting or demoting the values of an audience, to which the invention is headed, can be used to evaluate the candidate blends created. We also address the problem of how newly invented concepts are evaluated with respect to a Rich Background so as to decide which of them are to be accepted into a system of familiar concepts, and how this, in turn, may affect the previously accepted conceptualisation. As technique to tackle this problem we explore the applicability of Paul Thagard's computational theory of coherence, in particular, his notion of *conceptual coherence*. The process model is exemplified using two structured representation languages, namely order-sorted feature terms and description logic.

Roberto Confalonieri
Free University of Bozen-Bolzano, Faculty of Computer Science, Dominikanerplatz 3, 39100, Bozen-Bolzano, Italy
e-mail: Roberto.Confalonieri@unibz.it

Enric Plaza · Marco Schorlemmer
Artificial Intelligence Research Institute, Spanish National Research Council (IIIA-CSIC), Campus UAB, c/ Can Planes s/n, 08193, Bellaterra, Catalonia (Spain)
e-mail: enric@iiia.csic.es, marco@iiia.csic.es

* This chapter draws on material published in (Confalonieri et al., 2016b) and (Schorlemmer et al., 2016).

2.1 A Process Model for Concept Invention

Existing computational models for concept invention (see Section 2.7 for an overview) especially focus on the core mechanism of blending, that is, how blends are created, and re-interpret the optimality principles to evaluate the blends. In this chapter, we propose that a computational model also needs to deal with two extra dimensions to which we refer as the *origin* and *destination* of concept invention. The origin considers from where and how input spaces are selected, whereas the destination considers to whom the creation is headed.

A first assumption is that there is no creation *ex nihilo*. This is a widely held assumption in human creativity, be it scientific or artistic; we apply this assumption to any creative agent be it human or artificial. Specifically, combinatorial creativity depends on the experience and expertise of the creative agent (human or artificial) in a given domain, which in turn depends on the externally established 'state of the art' and prevalent assumptions, biases, and preferences on that domain. We express this assumption by claiming that every creative process has an *origin*, where the notion of origin is intended to capture and contain these individual and social preexisting tenets and assets that can potentially be used in a creative process. Specifically, we will model the notion of origin in a particular instance of a creative process as the Rich Background possessed by a creative agent on a particular domain.

The second assumption is that a given creative process has usually a purpose in creating something new. We express this assumption by saying that a creative process has a *destination*. A destination is different from a goal as usually understood in problem solving and Artificial Intelligence systems, where goals are related to the notion of satisfaction of specified sets of requirements or properties. A destination, in our approach, is a notion that is related, for instance in artistic domains, to the notions of audience or genre; different audiences or genres value different sets of properties or aspects as being worthy or even indispensable. Although we do not assume that a creative process has a specific goal, we do assume that a creative process is purposeful in producing an output that is destined to some 'target' audience, be it jazz aficionados in music, or academic colleagues in science. Specifically, we will model the notion of destination as the collection of values held dear by an intended audience. This approach gives us enough concretion to be able to talk about adequacy, significance, or interest of a creative outcome (if those are values held by an audience), while having enough leeway to encompass differences in subjective or individual appreciation or evaluation of a creative outcome by members of an actual audience.

To this end, we propose the following process model of concept invention (Figure 2.1):

- **Rich Background and Discovery**: The origin consists of a Rich Background, the set of concepts available to be blended. This set is finite but complex, diverse, polymathic and heterogeneous. Concepts are associated with a background, understood as the education, experience, and social circumstances of

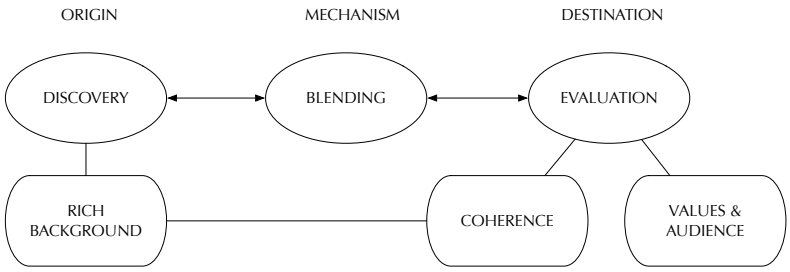


Fig. 2.1: A process model for concept invention

a (creative) individual. The Rich Background supports a discovery process that finds pairs of concepts that can be blended.

- **Blending**: Conceptual blending is the mechanism according to which two concepts are combined into a blended concept. Blending is here characterised in terms of amalgams, a notion that was developed for combining cases in case-based reasoning (Ontañón and Plaza, 2010). Conceptual blending is modeled in terms of an amalgam-based workflow. The blending of two concepts may result in a large number of blends, that need to be evaluated.
- **Arguments, Values, Audiences and Evaluation**: Values are properties expected from a good blend. Values are considered as points of view and can be of different kinds, e.g., moral, aesthetic, etc. A destination or audience is characterised by a preference relation over these values.[2] Arguments in favor or against a blend are built to evaluate the generated blends. An argument can promote or demote a value. In this way, the blends are evaluated depending on the audience for which they are created.
- **Conceptual Coherence and Evaluation**: The notion of coherence developed by Thagard (2000), when used to explain human reasoning, proposes that humans *accept* or *reject* a cognition (a percept, image, proposition, concept, etc.) depending on how much it contributes to maximising the number of constraints, that are imposed by situational context and other relevant cognitions. Among the different types of coherence proposed by Thagard (2000), conceptual coherence can be used to evaluate conceptual blends by measuring to what extent a blend coheres or incoheres with the Rich Background.

The rest of the chapter is organised as follows. The first four sections develop a general model that enacts the concept invention process depicted above. In Section 2.2, we model the notion of Rich Background and similarity-based discovery. In Section 2.3 we characterise a blend in terms of amalgams. In Section 2.4, we propose an argumentation framework based on values and audiences that can be used to

[2] Therefore, if the values are for example {*jazz*,*classical*}, then two audiences can be defined, one where *jazz* is preferred to *classical*, and another one, where *classical* is preferred to *jazz*. We will formalise these notions in Section 2.4.

evaluate conceptual blends by means of decision-criteria. Section 2.5 describes the computational coherence theory by Thagard (2000) and how it can be used in blend evaluation. In Section 2.6, we describe two instantiations of the process model by using two structured representation languages, feature-terms and description logic. Section 2.7 presents a survey of existing computational models for concept invention and how our concept invention process relates. Finally, Section 2.8 concludes the chapter.

2.2 Rich Background and Discovery

In cognitive theories of conceptual blending, input spaces to be blended are givens that represent how humans package some relevant information in the context in which the blend is created.

In our process model, an input space is a concept belonging to a library of concepts that we call Rich Background. Concepts can be represented by means of structured representations such as feature terms (Smolka and Aït-Kaci, 1989; Carpenter, 1992) or description logics (Baader et al., 2003), as we shall see in Section 2.6. The packaging of some relevant information corresponds to a discovery process that takes certain properties, which the blends need to satisfy, into account. The discovery takes a query as input, looks for concepts in the Rich Background, and returns an ordered set of pairs of concepts that can be blended.

2.2.1 Rich Background

The Rich Background consists of a finite set of concepts $\mathscr{C} = \{\psi_1, \psi_2, \ldots, \psi_n\}$ specified according to a language $\mathscr{L}$ for which a *subsumption* relation between formulas (or descriptions) of $\mathscr{L}$ can be defined.

Intuitively, the subsumption between formulas captures the idea of generality or specificity between two concepts. We say that a concept ψ_1 is subsumed by a concept ψ_2, denoted as $\psi_1 \sqsubseteq \psi_2$, if all information in ψ_1 is also in ψ_2. The subsumption relation induces a partial order on the set of all concept descriptions that can be formed using $\mathscr{L}$, i.e., the pair $\langle \mathscr{L}, \sqsubseteq \rangle$ is a *poset* for a given set of formulas. Additionally, $\mathscr{L}$ contains the elements $\bot$ and $\top$ representing the infimum element or supremum element w.r.t. the subsumption order, respectively.

Given the subsumption relation, for any two concepts ψ_1 and ψ_2, we can define the *anti-unification* and *unification* as their *least general generalisation* (LGG) and *most general specialisation* (MGS) respectively. These operations are relevant for defining both a similarity measure for comparing concepts, and the blend of two concepts as an amalgam (Confalonieri et al., 2016a,b).

Definition 2.1 (Least General Generalisation). The least general generalisation of two concepts ψ_1 and ψ_2, denoted as $\psi_1 \sqcap \psi_2$, is defined as the most specific concept

that subsumes both:

$$\psi_1 \sqcap \psi_2 = \{\psi \mid \ \psi_1 \sqsubseteq \psi \wedge \psi_2 \sqsubseteq \psi \text{ and } \nexists \psi' : \psi' \sqsubset \psi \wedge \psi_1 \sqsubseteq \psi' \wedge \psi_2 \sqsubseteq \psi'\}$$

The least general generalisation encapsulates all the information that is common to both ψ_1 and ψ_2. For this reason, it is relevant for defining a similarity measure. If two concepts have nothing in common, then $\psi_1 \sqcap \psi_2 = \bot$. The complementary operation to the least general generalisation is the most general specialisation of two descriptions.

Definition 2.2 (Most General Specialisation). The most general specialisation of two concepts ψ_1 and ψ_2, denoted as $\psi_1 \sqcup \psi_2$, is defined as the most general concept that is subsumed by both:

$$\psi_1 \sqcup \psi_2 = \{\psi \mid \psi \sqsubseteq \psi_1 \wedge \psi \sqsubseteq \psi_2 \text{ and } \nexists \psi' : \psi \sqsubset \psi' \wedge \psi' \sqsubseteq \psi_1 \wedge \psi' \sqsubseteq \psi_2\}$$

If two descriptions have contradictory information, then they do not have a most general specialisation.

The least general generalisation and the most general specification can be characterised as operations over a refinement graph of descriptions. The *refinement graph* is derived from the poset $\langle \mathscr{L}, \sqsubseteq \rangle$ as the poset $\langle \mathscr{G}, \prec \rangle$, where $\psi_1 \prec \psi_2$ denotes that ψ_2 is a generalisation refinement of ψ_1 (or equivalently ψ_1 is a specialisation refinement of ψ_2).

The refinement graph is defined by means of a *generalisation refinement operator* γ.

$$\gamma(\psi) = \{\psi' \in \mathscr{L} \mid \psi \sqsubseteq \psi'\}$$

The above definition states that γ is an operation that generalises a description to a set of descriptions. The refinement graph, then, is a directed graph whose nodes are descriptions, and for which there is an edge from a description ψ_1 to a description ψ_2, whenever $\psi_2 \in \gamma(\psi_1)$.

The refinement graph can be more or less complex depending on the representation language adopted and the type of refinement operator used.

A refinement operator γ can be characterised according to some desirable properties (van der Laag and Nienhuys-Cheng, 1998). We say that γ is:

- *locally finite*, if the number of generalisations generated for any given element by the operator is finite, that is, $\forall \psi \in \mathscr{L} : \gamma(\psi)$ is finite;
- *proper*, if an element is not equivalent to any of its generalisations, i.e., $\forall \psi_1, \psi_2 \in \mathscr{L}$, if $\psi_2 \in \gamma(\psi_1)$, then ψ_1 and ψ_2 are not equivalent;
- *complete*, if there are no generalisations that are not generated by the operator, i.e., $\forall \psi_1, \psi_2 \in \mathscr{L}$ it holds that if $\psi_1 \sqsubseteq \psi_2$, then $\psi_2 \in \gamma^*(\psi_1)$ (where $\gamma^*(\psi_1)$ denotes the set of all elements which can be reached from ψ_1 by means of γ in zero or a finite number of steps).

Designing a generalisation refinement operator that fulfills all the above properties is not possible in general, because one usually has to sacrifice completeness for finiteness, and let the computation of the operator terminate. This is the case also for

the generalisation refinement operators that we design for the ordered-sorted feature terms and description logic (see Section 2.6.1 and Section 2.6.2 respectively).

2.2.2 Similarity-Based Discovery

The main idea behind the similarity-based discovery is that, for each concept ψ_i in the Rich Background, we measure how ψ_i and a concept ψ_q—modeling a query—are similar and we use this measure to rank the results. The similarity between two descriptions can be defined by means of their LGG.

As previously stated, the least general generalisation of two descriptions $\psi_1 \sqcap \psi_2$ is a symbolic representation of the information shared by ψ_1 and ψ_2. It can be used to measure the similarity between concepts in a quantitative way. The refinement graph allows us to estimate the quantity of information of any description ψ. It is the length of the (minimal) *generalisation path* that leads from ψ to the most general term $\top$.

Definition 2.3 (Generalisation Path). A finite sequence of descriptions $\langle \psi_1, \ldots, \psi_m \rangle$ is a generalisation path $\psi_1 \xrightarrow{\gamma} \psi_m$ between ψ_1 and ψ_m when for each $1 \leq i \leq m$, $\psi_{i+1} \in \gamma(\psi_i)$. The length of $\langle \psi_1, \ldots, \psi_m \rangle$ is denoted as $\lambda(\psi_1 \xrightarrow{\gamma} \psi_m)$.

Therefore, the length $\lambda(\psi_1 \sqcap \psi_2 \xrightarrow{\gamma} \top)$ estimates the informational content that is common to ψ_1 and ψ_2. In order to define a similarity measure, we need to compare what is common to ψ_1 and ψ_2 with what is not common. To this end, we take the lengths $\lambda(\psi_1 \xrightarrow{\gamma} \psi_1 \sqcap \psi_2)$ and $\lambda(\psi_2 \xrightarrow{\gamma} \psi_1 \sqcap \psi_2)$ into account (see Figure 2.2). Then a similarity measure can be defined as follows.

Definition 2.4 (LGG-based similarity). The LGG-based similarity between two descriptions ψ_1 and ψ_2, denoted by $S_\lambda(\psi_1, \psi_2)$, is:

$$S_\lambda(\psi_1, \psi_2) = \frac{\lambda(\psi_1 \sqcap \psi_2 \xrightarrow{\gamma} \top)}{\lambda(\psi_1 \sqcap \psi_2 \xrightarrow{\gamma} \top) + \lambda(\psi_1 \xrightarrow{\gamma} \psi_1 \sqcap \psi_2) + \lambda(\psi_2 \xrightarrow{\gamma} \psi_1 \sqcap \psi_2)}$$

The measure S_λ estimates the ratio between the amount of information that is shared and the total information content. From a computational point of view, S_λ requires to compute two things: the LGG and the three lengths defined in the above equation. The computation of the LGG depends on the language representation used (see Section 2.6).

Given the above definitions, the discovery of concepts can be implemented by the following discovery algorithm.

Algorithm Discovery($\mathscr{C}, \gamma, \psi_q$)
  ForEach ($\psi_j \in \mathscr{C}$) Do
    $\lambda_i = S_\lambda(\psi_j, \psi_q)$
    $\mathscr{T} = \mathscr{T} \cup \langle \psi_j, \lambda_j \rangle$

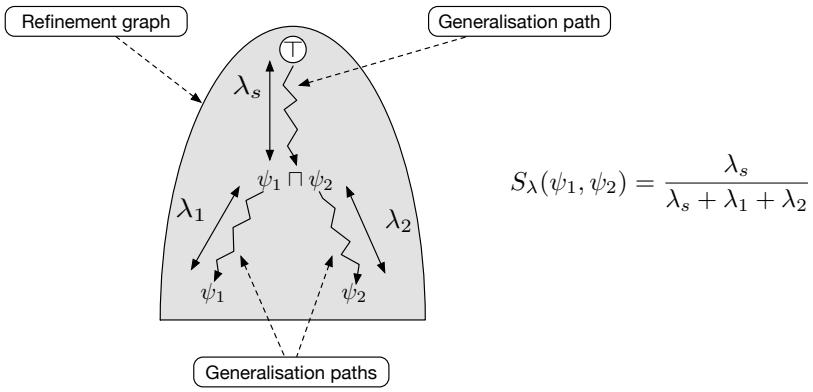


Fig. 2.2: Illustration of the LGG-based similarity, adapted from (Ontañón and Plaza, 2012)

EndForEach
$\mathscr{P} = conceptsPairs(\mathscr{T})$
Return $\mathscr{P}$
EndAlgorithm

The algorithm accepts a Rich Background of concepts $\mathscr{C}$, a query ψ_q, and a generalisation operator γ as inputs. $\dot{\cup}$ ranks the concepts discovered according to their similarity value λ_j.

The algorithm returns a ranked set of pairs of concepts. This ranking can be done according to different strategies. One way is to build all pairs of concepts and to rank them in a lexicographical order. The function *conceptsPairs* builds $\mathscr{P}$, as the set of pairs of concepts $\langle(\psi_j, \lambda_j), (\psi_k, \lambda_k)\rangle$ in which $\lambda_j \geq \lambda_k$ ($j \neq k$).

2.3 Blends as Amalgams

The computational model of concept blending is based on the notion of *amalgams* (Ontañón and Plaza, 2010). This notion was proposed in the context of case-based reasoning. Amalgams have also been used to model analogy (Besold and Plaza, 2015). According to this approach, input concepts are generalised until a generic space is found, and pairs of generalised input concepts are 'unified' to create blends.

Formally, the notion of amalgams can be defined in any representation language $\mathscr{L}$ for which a subsumption relation $\sqsubseteq$ between formulas (or descriptions) of $\mathscr{L}$ can be defined, together with the least general generalisation operation—playing the role of the generic space—and a most general specialisation (see Definitions 2.1 and 2.2).

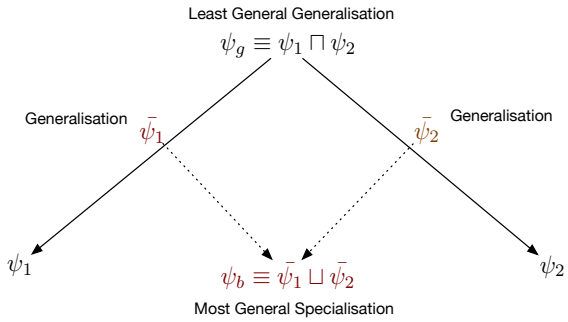


Fig. 2.3: A diagram of a blend ψ_b from inputs ψ_1 and ψ_2

A *blend* of two descriptions is a new description that contains *parts from these two descriptions*. For instance, an amalgam of 'a red French sedan' and 'a blue German minivan' is 'a red German sedan'; clearly, there are always multiple possibilities for amalgams, like 'a blue French minivan'.

For our purposes, we define a *blend* of two input descriptions as follows:

Definition 2.5 (Blend as Amalgam). A description $\psi_b \in \mathscr{L}$ is a blend of two inputs ψ_1 and ψ_2 (with LGG $\psi_g = \psi_1 \sqcap \psi_2$) if there exist two generalisations $\bar{\psi}_1$ and $\bar{\psi}_2$ such that:

1. $\psi_g \sqsubseteq \bar{\psi}_1 \sqsubseteq \psi_1$,
2. $\psi_g \sqsubseteq \bar{\psi}_2 \sqsubseteq \psi_2$, and
3. $\psi_b \equiv \bar{\psi}_1 \sqcup \bar{\psi}_2$.

The above definition is illustrated in Figure 2.3, where the LGG of the inputs is indicated as ψ_g, and the blend ψ_b is the unification of two concrete generalisations $\bar{\psi}_1$ and $\bar{\psi}_2$ of the inputs. Equality ($\equiv$) here should be understood as $\sqsubseteq$-equivalence, that is, $\psi_1 \equiv \psi_2$ iff $\psi_1 \sqsubseteq \psi_2$ and $\psi_2 \sqsubseteq \psi_1$.

Usually one is interested only in *maximal blends*, i.e., in those blends that contain the maximal information of their inputs. A blend ψ_b of two inputs ψ_1 and ψ_2 is maximal if there is no other blend ψ'_b of ψ_1 and ψ_2 such that $\psi_b \sqsubset \psi'_b$. The reason why one is interested in maximal blends is that a maximal blend captures as much information as possible from the inputs. Moreover, any non-maximal blend can be obtained by generalising a maximal blend.

However, the number of blends that satisfies the above definition can still be very large and selection criteria for filtering and ordering them are therefore needed. Fauconnier and Turner (2002) discussed optimality principles, however, the computational realisation of these principles lacks some flexibility, especially if we think that blend evaluation should not be limited to a merely accept or reject affair. It should be the output of a more open discussion, and the reasons that lead to that decision need to be made explicit.

To this end, we propose two alternative tools for blend evaluation. On the one hand, by taking the notion of argument into account, we define an argument-based decision making framework that allows us to select the best blend w.r.t. some values and audiences. On the other hand, we explore how coherence theory can serve for guiding the process of conceptual blending and for evaluating conceptual blends.

2.4 Arguments, Values and Audiences

An argument is a central notion in several frameworks for reasoning about defeasible information (Dung, 1995; Pollock, 1992), decision making (Amgoud and Prade, 2009; Bonet and Geffner, 1996), practical reasoning (Atkinson et al., 2004), and modelling different types of dialogues such as persuasion (Bench-Capon, 2003). In most existing works on argumentation, an argument is a reason for believing a statement, choosing an option, or doing an action. Depending on the application domain, an argument is either considered as a purely abstract entity, or it is a logical proof for a statement where the proof is built from a knowledge base.

In our model, arguments are reasons for accepting or rejecting a given blend. They are built by the agent when calculating the different values associated with a blend. Values are considered as points of view, and can have different origins, e.g., they can be moral, aestethic, etc.

Generally, there can be several values $\mathscr{V} = \{v_1, \ldots, v_k\}$. Each value is associated with a degree that belongs to the scale $\Delta = (0, \ldots, 1]$, where 0 and 1 are considered the worst and the best degree respectively.

Values play a different role depending on the target or audience towards which the creation is headed. Audiences are characterised by the values and by preferences among these values. Given a set of values $\mathscr{V}$, there are potentially as many audiences as there are orderings on $\mathscr{V}$.

Definition 2.6 (Audience). An audience is a binary relation $\mathscr{R} \subseteq \mathscr{V} \times \mathscr{V}$ which is irreflexive, asymmetric, and transitive. We say that v_i is preferred to v_j in the audience $\mathscr{R}$, denoted as $v_i \succ_{\mathscr{R}} v_j$, if $\langle v_i, v_j \rangle \in \mathscr{R}$.

Definition 2.7 (Cover relation). We say that a value v_j covers v_i in the audience $\mathscr{R}$, denoted as $v_i \dot{\succ}_{\mathscr{R}} v_j$, if $v_i \succ_{\mathscr{R}} v_j$ and $\nexists v_{i'}$ such that $v_i \succ_{\mathscr{R}} v_{i'} \succ_{\mathscr{R}} v_j$.

Given a blend, an argument is generated for each value. The degree of the value characterises the 'polarity' of the argument which can be *pro* or *con* a blend. Arguments pro promote a blend whereas arguments con demote it. Given a set of blends $\mathscr{B}$, the tuple $\langle \mathscr{B}, \mathscr{V}, \Delta \rangle$ will be called an argumentation framework.

Definition 2.8 (Argument). Let $\langle \mathscr{B}, \mathscr{V}, \Delta \rangle$ be an argumentation framework. Then:

- An *argument pro* a blend b is a tuple $\langle (v, \delta), b \rangle$ where $v \in \mathscr{V}$, $\delta \in \Delta$ and $0.5 \leq \delta \leq 1$
- An *argument con* b is a pair $\langle (v, \delta), b \rangle$ where $v \in \mathscr{V}$, $\delta \in \Delta$ and $0 < \delta < 0.5$

A function Val returns the value v associated with an argument and a function Deg returns δ.

The blend evaluation can be formulated as a decision problem in which one has to decide an order relation $\succeq_{\mathscr{B}}$ on the set of candidate blends $\mathscr{B}$. The definition of this relation is based on the set of arguments pro and con associated with the candidate blends. Depending on the kind of arguments that are considered and how they are handled, different decision criteria can be defined (Amgoud and Prade, 2009):

- **Unipolar decision criteria:** they focus either only on arguments pro or arguments con;
- **Bipolar decision criteria:** they take both arguments pro and con into account;
- **Meta-criteria:** they aggregate arguments pro and con into a meta-argument.

In what follows, we denote the set of arguments pro and con as $\mathscr{A}_p = \{\alpha_1, \ldots, \alpha_n\}$ and $\mathscr{A}_c = \{\alpha_1, \ldots, \alpha_m\}$ respectively. Besides, we assume to have the following functions: $\mathscr{M}_p : \mathscr{B} \to 2^{\mathscr{A}_p}$ and $\mathscr{M}_c : \mathscr{B} \to 2^{\mathscr{A}_c}$ that return the set of arguments pro and the set of arguments con associated with a blend respectively; $\mathscr{M} : \mathscr{B} \to 2^{\mathscr{A}_p \cup \mathscr{A}_c}$ that returns all arguments associated with a blend.

A basic decision criterion for comparing candidate blends can be defined by comparing the number of arguments pro associated with them.

Definition 2.9. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if $|\mathscr{M}_p(b_1)| \geq |\mathscr{M}_p(b_2)|$.

Notice that the above criterion guarantees that any pair of blends can be compared.

When the audience is taken into account, one may think of preferring a blend that has an argument pro whose value is preferred to the values of any argument pro the other blends.

Definition 2.10. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if $\exists \alpha \in \mathscr{M}_p(b_1)$ such that $\forall \alpha' \in \mathscr{M}_p(b_2)$, $\mathsf{Val}(\alpha) \succ_{\mathscr{R}} \mathsf{Val}(\alpha')$.

In the above definition, $\succeq_{\mathscr{B}}$ depends on the relation $\succ_{\mathscr{R}}$. Since $\succ_{\mathscr{R}}$ is a preference relation, some of the values of the arguments can be incomparable. In this case, b_1 and b_2 will not be comparable, either. This definition can be relaxed, for instance, by ignoring these arguments.

The counter-part decision criteria of Definitions 2.9-2.10 for the case of arguments con can be defined in a similar way.

Definition 2.11. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if $|\mathscr{M}_c(b_1)| \leq |\mathscr{M}_c(b_2)|$.

Definition 2.12. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if $\exists \alpha \in \mathscr{M}_c(b_1)$ such that $\forall \alpha' \in \mathscr{M}_c(b_2)$, $\mathsf{Val}(\alpha) \succ_{\mathscr{R}} \mathsf{Val}(\alpha')$.

In the case of bipolar decision criteria, we can combine the criterion dealing with arguments pro with the criterion dealing with arguments con.

Definition 2.13. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if $|\mathscr{M}_p(b_1)| \geq |\mathscr{M}_p(b_2)|$ and $|\mathscr{M}_c(b_1)| \leq |\mathscr{M}_c(b_2)|$.

Unfortunately, the above definition does not ensure that we can compare all the blends.

Finally, meta-criteria for deciding which blends are preferred can be defined by aggregating arguments pro and con into a meta-argument. Then, comparing two blends amounts to comparing the resulting meta-arguments. A simple criterion can be defined by aggregating the degrees of the arguments associated with a blend.

Definition 2.14. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if

$$\sum_{\alpha \in \mathscr{M}(b_1)} \mathsf{Deg}(\alpha) \geq \sum_{\alpha' \in \mathscr{M}(b_2)} \mathsf{Deg}(\alpha')$$

This definition can be extended to take the audience into account. To this end, we consider a rank function that maps each value of $\mathscr{R}$ to an integer. The rank function is defined as follows:

$$\mathsf{Rank}_{\mathscr{R}}(v) = \begin{cases} 1 & \text{if } \nexists v' \text{ s.t. } v' \dot{\succ}_{\mathscr{R}} v \\ \max\limits_{v' \dot{\succ}_{\mathscr{R}} v} \{\mathsf{Rank}_{\mathscr{R}}(v')\} + 1 & \text{otherwise} \end{cases}$$

Essentially, Rank counts how many values a certain value covers. This ranking is then used to define the following audience-based aggregation decision criterion.

Definition 2.15. Let $b_1, b_2 \in \mathscr{B}$. $b_1 \succeq_{\mathscr{B}} b_2$ if and only if

$$\sum_{\alpha \in \mathscr{M}(b_1)} \frac{\mathsf{Deg}(\alpha)}{\mathsf{Rank}_{\mathscr{R}}(\mathsf{Val}(\alpha))} \geq \sum_{\alpha' \in \mathscr{M}(b_2)} \frac{\mathsf{Deg}(\alpha')}{\mathsf{Rank}_{\mathscr{R}}(\mathsf{Val}(\alpha'))}$$

This last definition is based on an audience-based aggregation that sums the arguments' degrees by taking the preference order over values into account. This definition also guarantees that all the blends are comparable.

2.5 Coherence Theory

Thagard addresses the problem of determining which pieces of information, such as hypotheses, beliefs, propositions or concepts, should be accepted and which should be rejected based on the relationships of coherence and incoherence among them. That is, when two elements cohere, they tend to be accepted together or rejected together, and when two elements incohere, one tends to be accepted while the other tends to be rejected (Thagard, 2000).

This can be reformulated as a constraint satisfaction problem as follows. Pairs of elements that cohere form positive constraints, and pairs of elements that incohere form negative constraints. If we partition the set of pieces of information we are dealing with into a set of accepted elements and a set of rejected elements, then a positive constraint is satisfied if both elements of the constraint are either among the

accepted elements or among the rejected ones; and a negative constraint is satisfied if one element of the constraint is among the accepted ones and the other is among the rejected ones. The coherence problem is to find the partition that maximises the number of satisfied constraints.

Note that in general we may not be able to partition a set of elements so as to satisfy *all* constraints, thus ending up accepting elements that incohere between them or rejecting an element that coheres with an accepted one. The objective is to minimise these undesired cases. The coherence problem is known to be NP-complete, though there exist algorithms that find good enough solutions of the coherence problem while remaining fairly efficient.

Depending on the kind of pieces of information we start from, and on the way the coherence and incoherence between these pieces of information is determined, we will be dealing with different kinds of coherence problems. So, in *explanatory coherence* we seek to determine the acceptance or rejection of hypotheses based on how they cohere and incohere with given evidence or with competing hypotheses; in *deductive coherence* we seek to determine the acceptance or rejection of beliefs based on how they cohere and incohere due to deductive entailment or contradiction; in *analogical coherence* we seek to determine the acceptance or rejection of mapping hypotheses based on how they cohere or incohere in terms of structure; and in *conceptual coherence* we seek to determine the acceptance or rejection of concepts based on how they cohere or incohere as the result of the positive or negative associations that can be established between them. Thagard discusses these and other kinds of coherence.

Although Thagard provides a clear technical description of the coherence problem as a constraint satisfaction problem, and he enumerates concrete principles that characterise different kinds of coherences, he does not clarify the actual nature of the coherence and incoherence relations that arise between pieces of information, nor does he suggest a precise formalisation of the principles he discusses. Joseph et al. (2010) have proposed a concrete formalisation and realisation of deductive coherence, which they applied to tackle the problem of norm adoption in a normative multi-agent system. Here, we will focus on the problem of conceptual coherence and its applicability to conceptual blending as we shall see in Section 2.6.2.

2.5.1 Coherence Graphs

In this section we give precise definitions of the concepts intuitively introduced in the previous section.

Definition 2.16. A *coherence graph* is an edge-weighted, undirected graph $G = \langle V, E, w \rangle$, where:

1. V is a finite set of nodes representing pieces of information.
2. $E \subseteq V^{(2)}$ (where $V^{(2)} = \{\{u, v\} \mid u, v \in V\}$) is a finite set of edges representing the coherence or incoherence between pieces of information.

3. $w : E \to [-1,1] \setminus \{0\}$ is an edge-weighted function that assigns a value to the coherence between pieces of information.

Edges of coherence graphs are also called *constraints*.

When we partition the set V of vertices of a coherence graph (i.e., the set of pieces of information) into a set A of accepted elements and a set $R = V \setminus A$ of rejected elements, then we can say when a constraint—an edge between vertices—is satisfied or not by the partition.

Definition 2.17. Given a coherence graph $G = \langle V, E, w \rangle$, and a partition (A,R) of V, the set of *satisfied constraints* $C_{(A,R)} \subseteq E$ is given by:

$$C_{(A,R)} = \left\{ \{u,v\} \in E \;\middle|\; \begin{array}{l} u \in A \text{ iff } v \in A, \text{ whenever } w(\{u,v\}) > 0 \\ u \in A \text{ iff } v \in R, \text{ whenever } w(\{u,v\}) < 0 \end{array} \right\}$$

All other constraints (i.e., those in $E \setminus C_{(A,R)}$) are said to be *unsatisfied*.

The coherence problem is to find the partition of vertices that satisfies as many constraints as possible, i.e., to find the partition that maximises the coherence value defined as follows, which makes coherence independent of the size of the coherence graph.

Definition 2.18. Given a coherence graph $G = \langle V, E, w \rangle$, the *coherence of a partition* (A,R) of V is given by

$$\kappa(G,(A,R)) = \frac{\sum\limits_{\{u,v\} \in C_{(A,R)}} |w(\{u,v\})|}{|E|}$$

Notice that there may not exist a unique partition with a maximum coherence value. Actually, at least two partitions have the same coherence value, since $\kappa(G,(A,R)) = \kappa(G,(R,A))$ for any partition (A,R) of V.

2.5.2 Blend Evaluation by Means of Coherence

This section describes how coherence is used to evaluate blends. The overall idea is to compute the coherence graph and maximising partitions for each blend, and use the maximal coherence degree of the coherence graphs to rank the blends.

The process of evaluating blends according to conceptual coherence can be described as follows:

1. Given two input concepts, we generate a candidate blend according to Definition 2.5.

2. We form the coherence graph using the input concepts and the blend.[3]
3. We compute the coherence maximising partitions according to Definition 2.18 and we associate it to the blend.
4. We repeat this procedure for all the blends that can be generated from the mental spaces.
5. "Good" blends are those with maximal coherence degree.

Once the maximising partitions are computed, the coherence of the blend could be measured in terms of the coherence value of the coherence-maximising partitions. The degree of the coherence graph directly measures how much a blend coheres with the Rich Background.

Definition 2.19. Let $G = \langle V, E, w \rangle$ be the coherence graph of a blend B and let $\mathscr{P}$ be the set of partitions of G. The maximal coherence value of B of G is $\deg(B) = \max_{P \in \mathscr{P}} \{\kappa(G,P)\}$.

This maximal coherence value can be used to rank blends as follows.

Definition 2.20. For each $b_1, b_2 \in \mathscr{B}$, we say that b_1 is preferred to b_2 ($b_1 \succeq_{\mathscr{C}} b_2$) if and only if $\deg(b_1) \geq \deg(b_2)$.

The above criterion guarantees that any pair of blends can be compared.

2.6 Exemplifying the Process Model

In this section, we exemplify the process of concept invention making use of two use-cases, modeled according to two structured representation languages, i.e., feature terms and description logic.

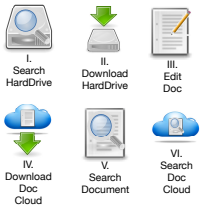
First, we show how a Rich Background of concepts representing computer icons is modeled in terms of feature terms and how conceptual blending can be used to model the creation of new computer icons. Following the process model, computer icons belonging to a Rich Background of icons are retrieved using a similarity measure (see Section 2.2); new blended icons are generated as amalgams (see Section 2.3), and evaluated by means of the argumentation framework introduced in Section 2.4.

Second, we exemplify how a certain form of coherence of Thagard, namely conceptual coherence, can be used to evaluate how new conceptual blends cohere w.r.t. a Rich Background of concepts. To this end, we propose a formalisation of conceptual coherence for concepts represented in the $\mathscr{AL}$ description logic, and explore by means of an illustrative example the role coherence may play in blend evaluation.

[3] This depends on the representation language used and the type of coherence considered. In Section 2.6, we show how a coherence graph for conceptual coherence can be built from a Rich Background of $\mathscr{AL}$ concepts.

$$x_1 : \text{ICON}\left[\begin{array}{l} form = \left\{\begin{array}{l} x_2 : \text{MAGNIFYINGGLASS}\left[\begin{array}{l} action = x_4 \\ on = x_3 \end{array}\right] \\ x_3 : \text{HARDDISK}\left[\; objectType = x_5 \;\right] \end{array}\right\} \\ meaning = \left\{\begin{array}{l} x_4 : \text{SEARCH} \\ x_5 : \text{HARDDRIVE} \end{array}\right\} \end{array}\right]$$

(a) Feature term representation of a computer icon



(b) Examples of computer icons

Fig. 2.4: Rich Background about computer icons

2.6.1 Creating Computer Icon Concepts

We assume that concept blending is the implicit process which governs the creative behavior of icon designers who *create* new icons by blending existing icons and signs. To this end, we propose a simple semiotic system for modeling computer icons. We consider computer icons as combinations of signs (e.g., document, magnifying glass, arrow, etc.) that are described in terms of *meanings*. Meanings convey *actions-in-the-world* or *types of objects* and are associated to signs. Signs are related by sign-patterns modeled as qualitative spatial relations such as *on*, *left*, etc.

2.6.1.1 A Rich Background of Computer Icons

Let the Rich Background be a collection of computer icons. We assume that computer icons are described in terms of *form* and a *meaning*. The form consists of a finite set of signs which are related by spatial relationships. Figure 2.4b(I) shows an example of an icon in which two signs, a MAGNIFYINGGLASS and a HARDDISK, are related by relation *on*. The meaning, on the other hand, is the interpretation that is given to an icon. For instance, a possible meaning associated to the icon in Figure 2.4b(I) is SEARCH-HARDDRIVE. We allow a sign to have different interpretations depending on the icons in which it is used.

We shall model the Rich Background by means of a finite set $\mathscr{C}$ of feature terms (Smolka and Aït-Kaci, 1989; Carpenter, 1992), each representing a concept. Here, feature terms are defined over a signature $\Sigma = \langle \mathscr{S}, \mathscr{F}, \preceq, \mathscr{X} \rangle$, where $\mathscr{S}$ is finite set of sort symbols, including $\top$ and $\bot$, which represent the most specific and the most general sort, respectively; $\mathscr{F}$ is a finite set of feature symbols; $\preceq$ is an order relation inducing an inheritance hierarchy such that $\bot \preceq s \preceq \top$, for all $s \in \mathscr{S}$; and $\mathscr{X}$ is a denumerable set of variables. Then, a feature term ψ has the form:

$$\psi := x : s[f_1 = \Psi_1, \ldots, f_n = \Psi_n]$$

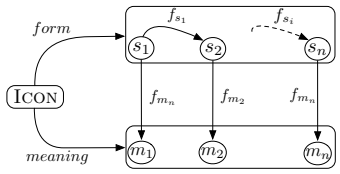


Fig. 2.5: Feature term representation of a computer icon

with $n \geq 0$, and where $x \in \mathscr{X}$ is called the root variable of ψ (denoted as $\mathsf{root}(\psi)$), $s \in \mathscr{S}$ is the sort of x (denoted as $\mathsf{sort}(x)$), and, for all j with $1 \leq j \leq n$, $f_j \in \mathscr{F}$ are the features of x (denoted as $\mathsf{features}(x)$) and the values[4] Ψ_j of the features are finite, non-empty sets of feature terms and/or variables (provided they are root variables of feature terms occurring in ψ). When the set of values of a feature is a singleton set, we will omit the curly brackets in our notation. We will write $\mathsf{vars}(\psi)$ to denote the set of variables occurring in a feature term ψ.

We choose to model icons as concepts represented by feature terms over the signature with the following sort hierarchy $\mathscr{S}$:[5]

ICON
SIGN $\prec$ {ARROW, MAGNIFYINGGLASS, DOCUMENT, PEN, HARDDISK, CLOUD}
MEANING $\prec$ {ACTION, OBJECTTYPE}
ACTION $\prec$ {MODIFY, VIEWSEARCH, TRANSFER}
MODIFY $\prec$ {EDIT, WRITE}
VIEWSEARCH $\prec$ {SEARCH, FIND, ANALYSE}
TRANSFER $\prec$ {UPLOAD, DOWNLOAD}
OBJECTTYPE $\prec$ {INFOCONTAINER, DATACONTAINER}
INFOCONTAINER $\prec$ {PAGE, DOC, FILE}
DATACONTAINER $\prec$ {HARDDRIVE, CLOUD}

and features $\mathscr{F} = \{form, meaning, on, below, left, right, action, objectType\}$.

In addition, feature terms representing icons need to have the structure represented in Figure 2.5. Root variables are of sort ICON and have at most two features *form* and *meaning*, modelling the signs $(s_1, \ldots, s_n)$ and the meaning $(m_1, \ldots, m_n)$ of these signs in the context of the icon. Each sign is again represented by means of a feature term whose root variable is of sort $s \succeq$ SIGN, and each meaning by means of feature terms whose root variable is of sort $s \succeq$ MEANING.

Features of sign terms ($f_{s_1}, \ldots f_{s_n}$ in the schema above) are at most one of *on*, *left*, *right*, or *below*, specifying the spatial relationship between signs; and at most one of *action* or *objectType*, specifying the meaning of signs ($f_{m_1}, \ldots f_{m_n}$ in the schema above). The values of spatial relation features are root variables of feature terms

[4] The meaning of 'values' in this section is different from the idea of 'values' in the argumentation framework presented in Section 2.4.

[5] The notation $s \prec \{s_1, \ldots, s_n\}$ denotes that $s_1, \ldots, s_n$ are sub-sorts of s.

that are in the value of the *form* feature; and those of features *action* and *objectType* are root variables of feature terms that are in the value of the *meaning* feature. In addition the root variables in the value of the *action* feature are of sort $s \succeq$ ACTION, while those of the *objectType* feature are of sort $s \succeq$ OBJECTTYPE. Figure 2.4a shows the feature term representation of the icon in Figure 2.4b(I).

A fundamental relation between feature terms is that of subsumption ($\sqsubseteq$). Intuitively, a feature term ψ_1 subsumes a feature term ψ_2 (ψ_1 is more general than ψ_2), if all the information in ψ_1 is also in ψ_2.[6] We write $\psi_1 \sqsubseteq \psi_2$ to denote that ψ_1 subsumes ψ_2. We omit the formal definition of subsumption, which can be found in (Ontañón and Plaza, 2012) for feature terms as represented here. The subsumption relation induces a partial order on the set of all features terms $\mathscr{L}$ over a given signature, that is, $\langle \mathscr{L}, \sqsubseteq \rangle$ is a poset.

2.6.1.2 Discovery

The discovery takes a query over the meaning of an icon concept as input, looks for concepts in the Rich Background, and returns an ordered set of pairs of concepts that can be blended. The query is modeled as a feature term ψ_q in which only the meaning part of an icon is specified. For instance, a query asking for an icon with meaning SEARCH-DOC is modeled as:

$$\psi_q := x_1 : \text{ICON} \left[\textit{meaning} = \left\{ \begin{array}{l} x_2 : \text{SEARCH} \\ x_3 : \text{DOC} \end{array} \right\} \right] \tag{2.1}$$

The matching of the query is not always a perfect match, since icon concepts in the Rich Background can have only one part of the meaning or similar meanings w.r.t. the meaning searched. To this end, the query resolution is modeled as a *similarity-based search*.

As seen in Section 2.2, the similarity between two concepts can be defined using the similarity measure S_λ. From a computational point of view, S_λ requires two things to be computed: the LGG and the three lengths defined in Eq. 2.4.

The algorithms for computing S_λ can be found in (Ontañón and Plaza, 2012). They implement the generalisation refinement operator shown in Figure 2.6. It consists of the following operations:

Sort generalisation, which generalises a term by substituting the sort of one of its variables by a more general sort;

Variable elimination, which generalises a term by removing the value of one of the features in one variable of the term (a variable is removed only when the variable does not have any features);

[6] Notice that, in Description Logics, $A \sqsubseteq B$ has the inverse meaning "A is subsumed by B", since subsumption is defined from the set inclusion of the interpretations of A and B. Also, this is the way in which we understand $\sqsubseteq$ in all the chapter apart from this section, in which we adopt the feature-term interpretation for $\sqsubseteq$.

(γ_s) SORT GENERALISATION:

$$\left[\begin{array}{l} s_1 \prec s \wedge \nexists s_2 : \; s_1 \prec s_2 \prec s \wedge \\ \forall x.f = y \in \phi \; \exists s_1.f = s_3 \in O \wedge \\ \mathsf{sort}(y) \preceq s_3 \end{array}\right] \frac{\phi \;\&\; x : s}{\phi \;\&\; x : s_1}$$

(γ_v) VARIABLE ELIMINATION:

$$\left[\begin{array}{l} s.f = s' \in O \wedge \\ \mathsf{features}(y) = \emptyset \end{array}\right] \qquad \frac{\phi \;\&\; x : s \;\&\; x.f = y \;\&\; y : s'}{\phi \;\&\; x : s}$$

(γ_e) VARIABLE EQUALITY ELIMINATION:

$$\left[\, z_1 \notin \mathsf{vars}(\phi) \,\right] \qquad \frac{\phi \;\&\; x.f = z \;\&\; y.f' = z}{\phi \;\&\; z.f = z \;\&\; y.f' = z_1 \;\&\; z_1 : \mathsf{sort}(z)}$$

(γ_r) ROOT VARIABLE EQUALITY ELIMINATION:

$$\left[\begin{array}{l} z_1 \notin \mathsf{vars}(\phi) \wedge \\ \mathsf{root}(\psi) = z \end{array}\right] \qquad \frac{\phi \;\&\; x.f = z}{\phi \;\&\; x.f = z_1 \;\&\; z_1 : \mathsf{sort}(z)}$$

Fig. 2.6: Generalisation operators for feature terms (Ontañón and Plaza, 2012), in which feature terms are represented in clause notation. The term form of any feature term $\psi := x : s[f_1 = \psi_1, \ldots, f_n = \psi_n]$ can be rewritten into the equivalent clause form $\phi := x : s \;\&\; x.f_1 = x_1 \;\&\; \ldots \;\&\; x.f_n = x_n$. Notice that these operators ensure that it is possible to reach $\bot$ from any feature term in the language.

Variable equality elimination, which generalises a term by removing a variable equality and ensuring that $\bot$ can be reached from any term.

It is worth noticing that, in case of variable equalities, it is not possible to define a generalisation operator for feature terms that is complete and still locally finite. However, for the purpose of defining a least general generalisation-based similarity, an operator which ensures that $\bot$ is reachable in a finite number of steps will suffice.

Example 2.1 (LGG example). Let us consider the feature terms ψ_q in Eq. 2.1 and ψ_1 in Figure 2.4a. The LGG $\psi_q \sqcap \psi_1$ is:

$$x_1 : \text{ICON} \left[\textit{meaning} = \left\{ \begin{array}{l} x_2 = \text{SEARCH} \\ x_3 = \text{OBJECTTYPE} \end{array} \right\} \right]$$

$\psi_q \sqcap \psi_1$ captures the information shared among the icon concept ψ_1 and the query ψ_q. Both of them have two meanings. According to the ontology previously defined, the most general sorts for variables x_2 and x_3 are SEARCH and OBJECTTYPE respectively. The *form* feature of ψ_1 is removed, since ψ_q does not contain this information.

The measure S_λ estimates the ratio between the amount of information that is shared and the total information content.

Example 2.2 (Similarity example). Let us consider the feature terms ψ_q in Eq. 2.1, ψ_1 in Figure 2.4a and their LGG in Example 2.1. Lengths $\lambda_1 = \lambda(\psi_1 \sqcap \psi_q \xrightarrow{\gamma} \bot) = 8$, $\lambda_2 = \lambda(\psi_1 \xrightarrow{\gamma} \psi_1 \sqcap \psi_q) = 12$, and $\lambda_3 = \lambda(\psi_q \xrightarrow{\gamma} \psi_1 \sqcap \psi_q) = 2$. Notice that λ_3 is very small (two generalisations), while λ_2 is larger since ψ_1 has more generalised content. Therefore, the similarity between ψ_q and ψ_1 is:

$$S_\lambda(\psi_1, \psi_q) = \frac{8}{12+2+8} = 0.36$$

$S_\lambda(\psi_1, \psi_q)$ expresses that these two concepts share 36% of their information.

This measure is used to retrieve and rank input concepts as shown in the following example.

Example 2.3. Let us imagine an agent that has access to a Rich Background $\mathscr{C} = \{\psi_1, \psi_2, \psi_3, \psi_4\}$ consisting of four of the icons depicted in Figures 2.4b(I-II-III-IV). As previously described, ψ_1 is a feature term representing an icon with meaning SEARCH-HARDDISK. ψ_2 represents an icon that consists of two sorts of type SIGN, an ARROW and a CLOUD, whose meaning is DOWNLOAD-CLOUD. ψ_3 represents an icon with two sorts of type SIGN, a PEN and a DOCUMENT, whose meaning is EDIT-DOC; finally, ψ_4 is a feature term that consists of three sorts, ARROW, DOCUMENT and CLOUD with the intended meaning of DOWNLOAD-DOC-CLOUD.

The agent receives as input a query asking for an icon with meaning SEARCH-DOC, ψ_q (Eq. 2.1). The discovery retrieves the following pairs of concepts:

$$\{\langle(\psi_1, 0.36), (\psi_3, 0.36)\rangle\}, \{\langle(\psi_1, 0.36), (\psi_2, 0.27)\rangle\}$$

$$\{\langle(\psi_3, 0.36), (\psi_2, 0.27)\rangle\}, \{\langle(\psi_1, 0.36), (\psi_4, 0.25)\rangle\}$$

$$\{\langle(\psi_3, 0.36), (\psi_4, 0.25)\rangle\}, \{\langle(\psi_2, 0.27), (\psi_4, 0.25)\rangle\}$$

The agent proceeds to blend the first pair in the list. To this end, it applies the amalgam-based blending.

2.6.1.3 Blending Computer Icons

The least general generalisation of ψ_1 and ψ_3 is an icon with two sorts of type SIGN, one *on* the other one, and with meaning ACTION and OBJECTTYPE respectively. The agent explores the space of generalisations and finds two maximal blends; a blend ψ_{b_1} describing an icon with two sorts of type MAGNIFYINGGLASS and DOCUMENT whose meaning is SEARCH-DOC; another blend ψ_{b_2} describing an icon with sorts of type PEN and HARDDISK whose meaning is EDIT-HARDDRIVE. Since ψ_{b_2} does not satisfy the query, it is discarded, and only ψ_{b_1} is kept. The creation of ψ_{b_1} is illustrated in Figure 2.7.

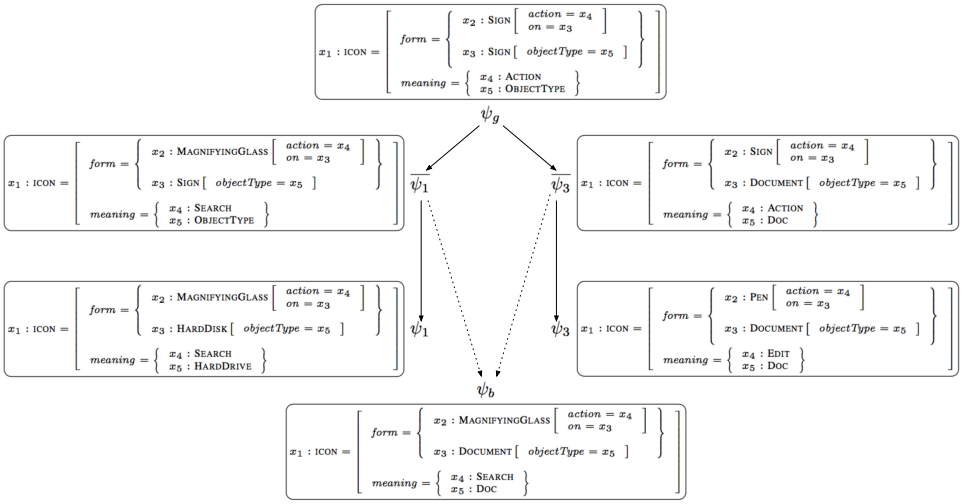


Fig. 2.7: A blend of feature terms ψ_1 and ψ_3

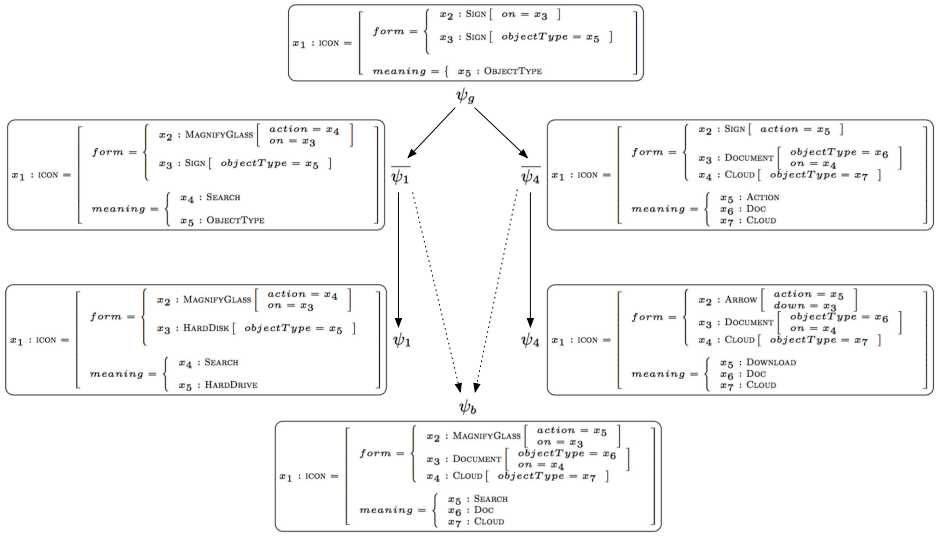


Fig. 2.8: A blend of feature terms ψ_1 and ψ_4

The agent repeats the above procedure for each pair discovered. Finally, it finds another blend, which satisfies ψ_q, by blending the pair ψ_1 and ψ_4. It is a blend describing an icon with three sorts of type MAGNIFYINGGLASS, DOCUMENT, and CLOUD whose meaning is SEARCH-DOC-CLOUD. Intuitively, this blend can be obtained by generalising HARDDISK from ψ_1 and ARROW from ψ_4, and by keeping the other input icons' specifics (see Figure 2.8). We denote this blend as ψ_{b_2}. The set of blends is $\mathscr{B} = \{\psi_{b_1}, \psi_{b_2}\}$. A representation of ψ_{b_1} and ψ_{b_2} is given in Figures 2.4b(V-VI).

2.6.1.4 Evaluating Conceptual Blends by Means of Arguments

The agent evaluates newly created concepts on the basis of some values and the audience to which these blends are headed.

In the case of evaluating blends representing new computer icons, we can imagine that the agent is equipped with values such as *simplicity* and *unambiguity*.

The main idea behind simplicity is that the agent estimates how simple an icon is from a representation point of view. This can be done by counting the quantity of information used in the feature term describing an icon. We can assume that simple icons are those described with less information. Therefore, simplicity is defined to be inversely proportional to the total number of features and sorts used in the variables of a feature term ψ_b.

$$\mathsf{Simplicity}(\psi_b) = \frac{1}{\sum\limits_{x \in \mathsf{vars}(\psi_b)} \mathsf{features}(x) + \mathsf{sorts}(x)}$$

Unambiguity, on the other hand, measures how many interpretations an icon has w.r.t. the Rich Background. Since icons are *polysemic*—they can be interpreted in different ways—there can be icons that contain the same sign but the sign is associated with a different meaning. To define the unambiguity value, let us first define the polysemic set of ψ_b as:

$$\begin{aligned}\mathsf{Pol}(\psi_b) = \{\psi_j \in \mathscr{C} \mid \exists s \in \mathsf{form}(\psi_j) \cap \mathsf{form}(\psi_b) \\ \wedge\, \mathsf{meaning}(\psi_j, s) \neq \mathsf{meaning}(\psi_b, s)\}\end{aligned}$$

where $\mathsf{form}(\psi_j)$ is a function that returns the value of feature *form*, i.e., the set of signs used in the icon represented by feature term ψ_j; and $\mathsf{meaning}(\psi_j, s)$ is a function that returns the sort of the variable that is the value of feature *action* or *objectType* of the variabe of sort s, i.e., the meaning used for the sign represented by sort s in feature term ψ_j. Then, the unambiguity value is defined to be inversely proportional to the cardinality of Pol.

$$\mathsf{Unambiguity}(\psi_b) = \begin{cases} 1/|\mathsf{Pol}(\psi_b)| & \text{if } |\mathsf{Pol}(\psi_b)| \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

Example 2.4. The agent evaluates the set of blends $\mathscr{B} = \{\psi_{b_1}, \psi_{b_2}\}$ by means of the values above. The blend ψ_{b_1} contains 10 variables whereas ψ_{b_2} contains 14. Therefore, the simplicity value's degrees of ψ_{b_1} and ψ_{b_2} are 0.1 and 0.07 respectively. Their unambiguity, on the other hand, is 1, since the Rich Background does not contain icons with the same signs used in ψ_{b_1} and ψ_{b_2}, but with a different meaning. The arguments built by the agent are:

	Simplicity	Unambiguity
ψ_{b_1}	0.1	1
ψ_{b_2}	0.07	1

Therefore, both blends have an argument pro regarding their simplicity and an argument con w.r.t. their unambiguity value. It is easy to see that the blends are ranked in different ways when using the criteria we defined. For instance, ψ_{b_1} and ψ_{b_2} are equally preferred when counting their arguments pro (or con) (Definition 2.9), and when considering both arguments pro and con (Definition 2.13).

Instead, when considering the audience $\mathsf{Simplicity} \succ_{\mathscr{R}} \mathsf{Unambiguity}$, ψ_{b_1} is preferred to ψ_{b_2} (Definition 2.15).

2.6.2 Coherent Conceptual Blending

The process model introduced in Section 2.1 can be instantiated in another formal structured representation language such as Description Logics (DLs).

Description logics play an important role in conceptual blending, as witnessed by other approaches (see Chapter 3) that make use of ontological descriptions as formal backbones for modelling conceptual blending in a computational way.

In the following, we will focus on how a specific description logic, namely $\mathscr{AL}$, can be used to model concepts belonging to a Rich Background, amalgam-based blending and conceptual coherence. The main reason for choosing $\mathscr{AL}$ is that it is a subset of OWL 2, the Web Ontology Language recommended by the World Wide Web Consortium (W3C, `http://www.w3.org`), and supported by the DOL metalanguage (see Chapter 3). In this way, our approach could be integrated in the DOL-based computational blending framework presented in the next chapter rather straightforwardly.

2.6.2.1 Rich Background in $\mathcal{AL}$

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.

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

Table 2.1: Syntax and semantics of $\mathscr{AL}$ contructors

House $\sqsubseteq$ Object	Resident $\sqsubseteq$ Person
Boat $\sqsubseteq$ Object	Passenger $\sqsubseteq$ Person
Land $\sqsubseteq$ Medium	Person $\sqcap$ Medium $\sqsubseteq \bot$
Water $\sqsubseteq$ Medium	Object $\sqcap$ Medium $\sqsubseteq \bot$
Water $\sqcap$ Land $\sqsubseteq \bot$	Object $\sqcap$ Person $\sqsubseteq \bot$

Fig. 2.9: The Rich Background for the House and Boat

The $\mathscr{AL}$ language was introduced by Schmidt-Schauß and Smolka (1991) as a minimal language of practical interest. Concept descriptions in $\mathscr{AL}$ are formed according to the syntax rules shown in the left column in Table 2.1.

The semantics of concept and role descriptions is defined in terms of an interpretation $\mathscr{I} = (\Delta^{\mathscr{I}}, \cdot^{\mathscr{I}})$, where $\Delta^{\mathscr{I}}$ is a non-empty domain and $\cdot^{\mathscr{I}}$ is an interpretation function assigning a set $A^{\mathscr{I}} \subseteq \Delta^{\mathscr{I}}$ to each concept name $A \in N_C$, a set $r^{\mathscr{I}} \subseteq \Delta^{\mathscr{I}} \times \Delta^{\mathscr{I}}$ to each role name $r \in N_r$, and an element $a^{\mathscr{I}} \in \Delta^{\mathscr{I}}$ for each individual name $a \in N_I$, which is extended to general concept and role descriptions. Table 2.1 shows the interpretation of the constructors of the description logic $\mathscr{AL}$.

The bottom concept $\bot$, in combination with general concept inclusions (GCIs), allows one to express disjointness of concept descriptions, e.g., $C \sqcap D \sqsubseteq \bot$ tells that C and D are disjoint. In $\mathscr{AL}$, and generally speaking in any description logic, there are two sets of axioms, namely, a TBox and an ABox.

The TBox, denoted as $\mathscr{T}$, consists of terminological axioms that describe intensional knowledge defining the main notions relevant to the domain of discourse. The ABox, denoted as $\mathscr{A}$, consists of assertional axioms that describe extensional knowledge about individual objects of the domain.

An interpretation $\mathscr{I}$ is a model of a TBox $\mathscr{T}$ if and only if it satisfies all axioms in $\mathscr{T}$. The basic reasoning task in $\mathscr{AL}$ is subsumption. Given a TBox $\mathscr{T}$ and two concept descriptions C and D, we say that C is (strictly) subsumed by D w.r.t. $\mathscr{T}$, denoted as $C \sqsubseteq_{\mathscr{T}} D$ ($C \sqsubset_{\mathscr{T}} D$), iff $C^{\mathscr{I}} \subseteq D^{\mathscr{I}}$ ($C^{\mathscr{I}} \subseteq D^{\mathscr{I}}$ and $C^{\mathscr{I}} \neq D^{\mathscr{I}}$) for every model $\mathscr{I}$ of $\mathscr{T}$.

In what follows, a Rich Background in $\mathscr{AL}$ is a TBox, containing terminological axioms of the form $C \sqsubseteq D$, i.e., GCIs, and disjointness axioms. By $\mathscr{L}(\mathscr{T})$ we refer to the set of all $\mathscr{AL}$ concept descriptions we can form with the concept and role names occurring in $\mathscr{T}$.

To illustrate an example of Rich Background, we use the classical conceptual blending example of the *house-boat* (Fauconnier and Turner, 2002; Goguen, 1999). In Figure 2.9, we depict the set of axioms, necessary for defining the mental spaces of the House and Boat. The precise formalisation is not critical at this point, different ones exist (Goguen and Harrell, 2010; Pereira, 2007), but all provide similar distinctions.

The Rich Background for the house and boat consists of a taxonomy of concepts, concept descriptions, and restrictions among them. For instance, Land and Water are atomic concepts, both of type Medium, and the axiom $\mathsf{Water} \sqcap \mathsf{Land} \sqsubseteq \bot$ captures the idea that any object of type Water cannot be of type Land at the same time.

Atomic roles such as usedBy and on are used to define concept relations. The mental spaces representing the concept of a house and boat can be modeled as follows:

$$\begin{aligned} \mathsf{House} &\sqsubseteq \forall \mathsf{usedBy}.\mathsf{Resident} \sqcap \forall \mathsf{on}.\mathsf{Land} \\ \mathsf{Boat} &\sqsubseteq \forall \mathsf{usedBy}.\mathsf{Passenger} \sqcap \forall \mathsf{on}.\mathsf{Water} \end{aligned}$$

The above axioms denote that a *house is an object that is only used by residents and is located only on land*. Similarly, *boat is an object that is only used by passengers and is located only on water*.

In principle, the House and Boat theory could not be directly blended (they generate an inconsistency due to the disjointness axiom $\mathsf{Water} \sqcap \mathsf{Land} \sqsubseteq \bot$), but the blended specification is still to be considered an interesting option—from a creative point of view—that needs to be assessed. We will do it by means of conceptual coherence, as we shall see. First, we define a blend as an amalgam in the $\mathscr{AL}$ language.

2.6.2.2 Blending in $\mathscr{AL}$

As said earlier, the notion of blend as an amalgam can be defined in any representation language $\mathscr{L}$ for which a subsumption relation between formulas is defined, therefore, also in the set of all $\mathscr{AL}$ concept descriptions, which can be formed with the concept and role names occurring in an $\mathscr{AL}$ TBox $\mathscr{T}$, with the subsumption relation $\sqsubseteq_{\mathscr{T}}$. The process of conceptual blending in $\mathscr{AL}$ can be described as follows:

1. We take a Rich Background of concepts (see Figure 2.9).
2. A mental space of an atomic concept A is modelled, for the purpose of conceptual blending, by means of a subsumption $A \sqsubseteq C$ specifying the necessary conditions we are focusing on.
3. The new concept to be invented is represented by the concept description that conjoins the atomic concepts to be blended.
4. With amalgams we generalise the input spaces based on the taxonomy in our Rich Backgroud until a satisfactory blend is generated.

The definitions of most general specialisation, least general generalisation, and amalgam in $\mathscr{AL}$ follow by replacing the subsumption relation ($\sqsubseteq$) with subsumption in $\mathscr{AL}$ ($\sqsubseteq_{\mathscr{T}}$) in a straightforward way; therefore, we omit them.

The least general generalisation and the generalised descriptions, needed to compute an amalgam (see Definition 2.5), are obtained by means of a generalisation refinement operator that allows us to find generalisations of $\mathscr{AL}$ concept descriptions.

Generalising $\mathcal{AL}$ descriptions

Roughly speaking, a generalisation operator takes a concept C as input and returns a set of descriptions that are more general than C by taking a Tbox $\mathscr{T}$ into account.

In order to define a generalisation refinement operator for $\mathscr{AL}$, we need some auxiliary definitions.

Definition 2.21. Let $\mathscr{T}$ be a TBox in $\mathscr{AL}$. The set of *non-trivial subconcepts* of $\mathscr{T}$ is given as

$$\mathsf{sub}(\mathscr{T}) = \bigcup_{C \sqsubseteq D \in \mathscr{T}} \mathsf{sub}(C) \cup \mathsf{sub}(D)$$

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

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

We next define the upward cover set of atomic concepts. In the following definition, the definition of $\mathsf{sub}(\mathscr{T})$ guarantees that the upward cover set is finite.

Definition 2.22. Let $\mathscr{T}$ be an $\mathscr{AL}$ TBox with concept names from N_C. The *upward cover set* of an atomic concept $A \in N_C \cup \{\top, \bot\}$ with respect to $\mathscr{T}$ is given as:

$$\begin{aligned}
\mathsf{UpCov}(A) := \{C \in \mathsf{sub}(\mathscr{T}) \mid\ & A \sqsubseteq_{\mathscr{T}} C \\
& \text{and there is no } C' \in \mathsf{sub}(\mathscr{T}) \\
& \text{such that } A \sqsubset_{\mathscr{T}} C' \sqsubset_{\mathscr{T}} C\}
\end{aligned} \tag{2.2}$$

We can now define our generalisation refinement operator for $\mathscr{AL}$ as follows.

Definition 2.23. Let $\mathscr{T}$ be an $\mathscr{AL}$ TBox. We define the *generalisation refinement operator* γ inductively over the structure of concept descriptions as follows:

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

We should note at this point that our definition of UpCov only considers the set of subconcepts present in a Tbox $\mathscr{T}$. On the one hand, this guarantees that γ is finite, since at each generalisation step, the set of possible generalisations is finite. On the other hand, however, this implies that γ is not complete, since it cannot find all possible upward covers of a concept w.r.t. subsumption in $\mathscr{AL}$.[7] Besides, γ can return concept descriptions that are equivalent to the concept being generalised; consequently, γ is not a proper generalisation operator. One possible way to avoid this situation is to discard these generalisations. This can be achieved by an additional semantic test that can be found in (Confalonieri et al., 2016a).

Given a generalisation refinement operator γ, $\mathscr{AL}$ concepts are related by refinement paths as described next.

Definition 2.24. A finite sequence $C_1, \ldots, C_n$ of $\mathscr{AL}$ 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.

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

Definition 2.25. An $\mathscr{AL}$ concept description G is a generic space of the $\mathscr{AL}$ concept descriptions $C_1, \ldots, C_n$ if and only if $G \in \gamma'^*(C_i)$ for all $i = 1, \ldots, n$.

The House-Boat Blend

The $\mathscr{AL}$ theories for House and Boat introduce the axioms modelling the mental spaces for *house* and *boat*.

$$
\begin{aligned}
\mathsf{House} &\sqsubseteq \forall \mathsf{usedBy}.\mathsf{Resident} \sqcap \forall \mathsf{on}.\mathsf{Land} \\
\mathsf{Boat} &\sqsubseteq \forall \mathsf{usedBy}.\mathsf{Passenger} \sqcap \forall \mathsf{on}.\mathsf{Water}
\end{aligned}
$$

[7] For instance, if $\mathscr{T}$ contains two axioms $A \sqsubseteq B$, $A \sqsubseteq C$, and we generalise A (in the domain knowledge), then $\gamma(A) = \{B, C\}$ while a possible generalisation of A w.r.t. $\sqsubseteq_{\mathscr{T}}$ is $B \sqcap C$.

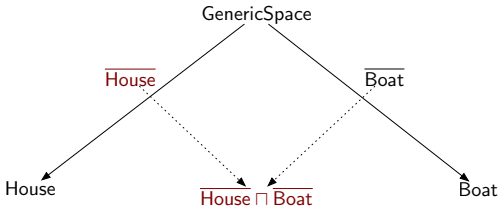


Fig. 2.10: A diagram of an amalgam from descriptions House and Boat and their respective generalisations $\overline{\mathsf{House}}$ and $\overline{\mathsf{Boat}}$. Arrows indicate the subsumption of the target by the source of the arrow

The House and Boat theories cannot be directly blended since they generate an inconsistency. This is due to the background ontology stating that the medium on which an object is situated cannot be *land* and *water* at the same time (Figure 2.9). Therefore, some parts of the House and Boat descriptions need to be generalised in a controlled manner before these concepts can be blended. The generic space between a house and a boat—an object that is on a *medium* and *used-by* a *person*—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 a *house-boat*. The generic space is obtained according to Definition 2.25 by applying the refinement operator γ.

Example 2.5. Let us consider the House and Boat concepts. Their generic space is: $\forall$usedBy.Person$\sqcap$ $\forall$on.Medium and is obtained as follows. In the House concept, the subconcepts $\forall$usedBy.Resident and $\forall$on.Land are generalised to $\forall$usedBy.Person and $\forall$on.Medium respectively. In the Boat concept, the subconcepts $\forall$usedBy.Passenger and $\forall$on.Water are generalised in a similar way.

From a conceptual blending point of view, the *house-boat* blend can be created when the medium on which a house is situated (land) becomes the medium on which boat is situated (water), and the resident of the house becomes the passenger of the boat. This blend can be obtained when the input concepts house and boat are generalised as follows:

$$\overline{\mathsf{House}} \sqsubseteq \forall \mathsf{usedBy.Resident} \sqcap \forall \mathsf{on.Medium}$$
$$\overline{\mathsf{Boat}} \sqsubseteq \forall \mathsf{usedBy.Person} \sqcap \forall \mathsf{on.Water}$$

The *house-boat* blend is obtained by conjoining the generalised mental spaces $\overline{\mathsf{House}}$ and $\overline{\mathsf{Boat}}$ (Figure 2.10). It is easy to see that $\overline{\mathsf{House}} \sqcap \overline{\mathsf{Boat}}$ is an amalgam according to Definition 2.5.

2.6.2.3 Conceptual Coherence in $\mathcal{AL}$

Thagard (2000) characterises conceptual coherence with these principles:

Symmetry: Conceptual coherence is a symmetric relation between pairs of concepts.

Association: A concept coheres with another concept if they are positively associated, i.e., if there are objects to which they both apply.

Given Concepts: The applicability of a concept to an object may be given perceptually or by some other reliable source.

Negative Association: A concept incoheres with another concept if they are negatively associated, i.e., if an object falling under one concept tends not to fall under the other concept.

Acceptance: The applicability of a concept to an object depends on the applicability of other concepts.

To provide a precise account of these principles we shall formalise *Association* and *Negative Association* between concepts expressed in $\mathscr{AL}$, since these are the principles defining coherence and incoherence. We shall assume coherence between two concept descriptions when we have explicitly stated that one subsumes the other ("there are objects to which both apply"); and we shall assume incoherence when we have explicitly stated that they are disjoint ("an object falling under one concept tends not to fall under the other concept").

Definition 2.26. Given a Tbox $\mathscr{T}$ in description logic $\mathscr{AL}$ and a pair of concept descriptions $C, D \notin \{\top, \bot\}$, we will say that:

- C coheres with D if $C \sqsubseteq D \in \mathscr{T}$, and that
- C incoheres with D if $C \sqsubseteq \neg D \in \mathscr{T}$ or $C \sqcap D \sqsubseteq \bot \in \mathscr{T}$.

In addition, coherence and incoherence between concept descriptions depend on the concept constructors used, and we will say that, for all atomic concepts A, atomic roles R, and concept descriptions $C, D \notin \{\top, \bot\}$:

- $\neg A$ incoheres with A;
- $C \sqcap D$ coheres both with C and with D;
- $\forall R.C$ coheres (or incoheres) with $\forall R.D$ if C coheres (or incoheres) with D.[8]

Symmetry follows from the definition above, and *Acceptance* is captured by the aim of maximising coherence in a coherence graph. For this we need to define how a TBox determines a coherence graph, and, in order to keep the graph finite, we express coherence and incoherence only between non-trivial concept descriptions (i.e., excluding $\top$ and $\bot$) that are explicitly stated in the TBox. This set can be computed based on Definition 2.21:

$$\mathsf{sub}'(\mathscr{T}) = \mathsf{sub}(\mathscr{T}) \backslash \{\bot, \top\}$$

Definition 2.27. The *coherence graph of a TBox* $\mathscr{T}$ is the edge-weighted, undirected graph $G = \langle V, E, w \rangle$ whose vertices are non-trivial subconcepts of $\mathscr{T}$ (i.e.,

[8] Note that since $\mathscr{AL}$ allows only for limited existential quantification we cannot provide a general rule for coherence between concept descriptions of the form $\exists R.\top$.

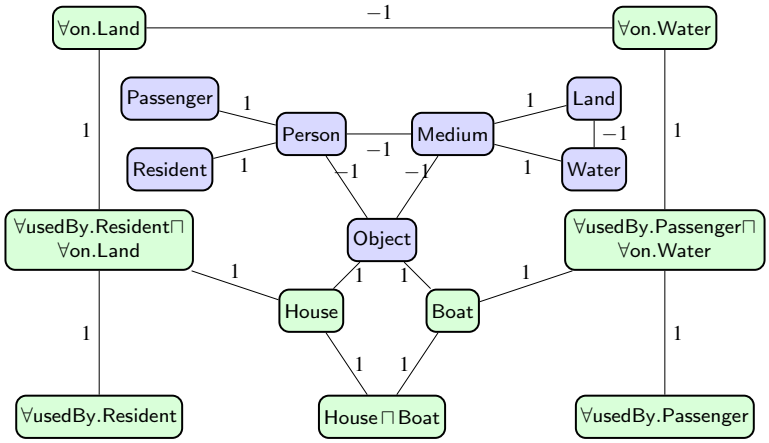


Fig. 2.11: The coherence graph of the House⊓Boat blend, showing the main concepts and their coherence relations. Blue and green coloured boxes represent concepts belonging to the background ontology and to the input mental spaces respectively

$V = \mathsf{sub}'(\mathscr{T})$), whose edges link subconcepts that either cohere or incohere according to Definition 2.26, and whose edge-weight function w is given as follows:

$$w(\{C,D\}) = \begin{cases} 1 & \text{if } C \text{ and } D \text{ cohere} \\ -1 & \text{if } C \text{ and } D \text{ incohere} \end{cases}$$

2.6.2.4 Evaluating the Coherence of Conceptual Blends

To exemplify how the coherence degree can be used to evaluate blends, we consider the *house-boat* example. According to the amalgam-based process of conceptual blending described in the previous section, several blends can be generated by blending the mental space of House and Boat. In particular, the concept House⊓Boat is a valid blend.

The coherence graph blending the House and Boat directly is shown in Figure 2.11. As expected the concepts House and Boat positively cohere with the axioms representing the mental spaces and with the concept House⊓Boat, which is representing the blend. The incoherence relation between ∀on.Land and ∀on.Water is due to the fact that the concepts Water and Land incohere, since the background ontology contains the disjointness axiom Water⊓Land ⊑ ⊥. The coherence graph of House and Boat has a maximal coherence value of 0.84.

For the sake of our example, we generate new blends by generalising the axioms modelling our mental spaces. For instance, by applying the generalisations seen in

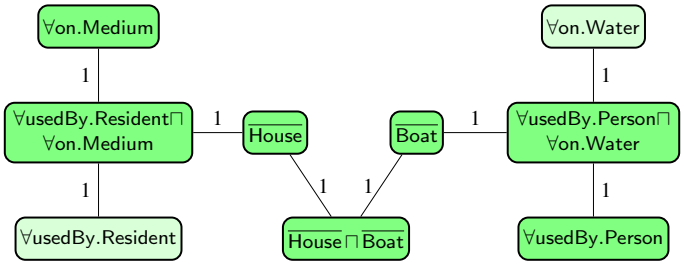


Fig. 2.12: The coherence graph of the $\overline{\mathsf{House}} \sqcap \overline{\mathsf{Boat}}$ blend, showing the main concepts and coherence relations. Generalised concepts are displayed in a darker tone

the previous section that lead to the creation of the *house-boat* blend, we obtain the coherence graph in Figure 2.12.[9] The coherence graph of blending $\overline{\mathsf{House}}$ and $\overline{\mathsf{Boat}}$ has a maximal coherence value of 0.9. This graph yields a higher coherence degree since generalising $\forall$on.Land to $\forall$on.Medium prevents the appearance of the incoherence relation between $\forall$on.Land and $\forall$on.Water.

By Definition 2.20, it is easy to see that the blend $\overline{\mathsf{House}} \sqcap \overline{\mathsf{Boat}}$ is preferred to $\mathsf{House} \sqcap \mathsf{Boat}$ since it has a maximal coherence degree that is higher.

2.7 Related Work

Several approaches of formal and computational models for concept invention have been proposed (Eppe et al., 2015a,b; Kutz et al., 2014; Goguen and Harrell, 2006; Veale and Donoghue, 2000; Pereira, 2005, 2007; Goguen and Harrell, 2010; Guhe et al., 2011). Many of these models are inspired by the work of Fauconnier and Turner (2002), but there are also other approaches emanating from analogical reasoning (Schwering et al., 2009) and neuroscience (Thagard, 2010).

Amalgam-based conceptual blending have been developed to blend $\mathscr{EL}^{++}$ concepts in (Confalonieri et al., 2016a). In this work, the generalisation of an $\mathscr{EL}^{++}$ concept is achieved by means of a generalisation refinement operator. The refinement operator is implemented in ASP as a step-wise transition process—similar to the one presented in this paper— that looks for a generic space between two (or more) concepts. The operator generalises a concept by taking the TBox knowledge into account. Good blends are selected by re-interpreting some optimality principles. Blending ontologies rather than concepts has been explored in the ontological blending framework of Kutz et al. (2014), where blends are computed as *colimits* of blending diagrams specified according to the Distributed Ontology Language (DOL) (Mossakowski et al., 2015), a recent OMG international ontology interoperability standard. In that framework, the blending process is not characterised

[9] Concepts belonging to the background ontology are omitted.

in terms of amalgams, nor are input concepts generalised syntactically. Rather, the generic space is assumed to be given and mapped to the input ontologies via theory interpretations.

The Alloy algorithm for conceptual blending by Goguen and Harrell (2006) is based on the theory of algebraic semiotics (Goguen, 1999). Alloy has been integrated in the Griot system for automated narrative generation (Goguen and Harrell, 2006; Harrell, 2007, 2005). The input spaces of the Alloy algorithm are theories defined in the algebraic specification language OBJ (Malcolm, 2000). In the Alloy algorithm, input spaces are assumed to be given, hence there is no discovery. The optimality principles proposed by Fauconnier and Turner (2002) are re-interpreted as *structural* optimality principles, and serve to prune the space of possible blends.

Sapper was originally developed by Veale and Keane (1997) as a computational model of metaphor and analogy. It computes a mapping between two separate domains—understood as graphs of concepts—that respects the relational structure between the concepts in each domain. Sapper can be seen as a computational model for conceptual blending, because the pairs of concepts that constitute its output can be manipulated as blended concepts (Veale and Donoghue, 2000). Strictly speaking, Sapper does not work with *a priori* given input spaces. It is the structure mapping algorithm itself which determines the set of concepts and relations between these concepts. In Sapper, most of the optimality principles are captured and serve to rank and filter the correspondences that comprise the mappings computed by the algorithm.

The research in (Pereira and Cardoso, 2002, 2003a,b) led to the development of Divago (Pereira, 2005; Pereira and Cardoso, 2006; Pereira, 2007), probably the first complete implementation of conceptual blending. Pereira draws the terminology and definitions for his formal and computational model from Wiggin's formalisation of creative systems (Wiggins, 2006). The implementation of Divago is realised in Prolog. Divago's architecture includes different modules. A knowledge base contains different micro-theories and their instantiations. Of these, two are selected for the blending by the user or randomly, thus, no discovery is taken into account. A mapper then generates the generic space between the inputs, and passes it to a blender module which generates the 'blendoid', i.e., a projection that defines the space of possible blends. A factory component is used to select the best blends among the blendoid by means of a genetic algorithm. A dedicated module implements the optimality principles. Given a blend, this module computes a measure for each principle. These measures yield a preference value of the blend that is taken as the fitness value of the genetic algorithm.

The combinatorial kind of creativity (Boden, 1996) that we are interested in has been investigated from a neurological perspective by Thagard and Stewart (2011). The major motivation of their approach is to explain and to model the *Aha!* or *Eureka!* effect that occurs when humans make serendipitous discoveries by means of creative thinking. The authors build their work on findings from neuroscience and approaches to realise human thinking with neural networks (Thagard, 2010). The key idea is to represent mental concepts as activity patterns of vectors of neurons and to perform a convolution operation to combine these patterns. Activity pat-

terns are mathematically represented as vectors of numbers that represent the firing rate of neurons. According to Thagard and Stewart (2011), a mental concept can then be represented as a huge but finite vector of such numbers. The blend is generated by mathematical convolution of vectors. The underlying mathematical model is based on the so-called LIF model of neuronal activity (see e.g., Thagard (2010)). It accounts for various details on the neuronal level, such as neuron voltage, input current, membrane time, direction vector of neuron patterns, and synaptic connection weights. Thagard and Stewart (2011) do not use Fauconnier and Turner's optimality principles to distinguish reasonable blends within the huge space of possible blends. Instead, they combine the blend of two input spaces with another space representing emotional reaction to assess blends. However, the authors do not provide a detailed description of how to model the emotional input spaces computationally.

Finally, works that relate to ours are (Confalonieri et al., 2015; Kaliakatsos-Papakostas et al., 2016). Confalonieri et al. (2015) use Lakatosian reasoning to model dialogues in which users engage to discuss the intended meaning of an invented concept. The main difference between that effort and the current work lies in the way in which arguments are generated and used. Here, an argument is a reason for choosing a blend and it is generated automatically, whereas in (Confalonieri et al., 2015) an argument is a reason to refine the meaning of a blend and is provided by the user. In (Kaliakatsos-Papakostas et al., 2016), arguments are specified by musicologists to drive the harmonic blending process.

2.8 Conclusion and Future Perspectives

In this chapter, we described a process model for concept invention that is based on and extends the conceptual blending theory of Fauconnier and Turner (2002). According to this process, concept invention is characterised by different subprocesses—discovery, blending, and evaluation—that together account for concept invention.

Apart from the blending mechanism modelling the creation of new concepts, we focused on two extra dimensions that are typically not addressed in computational approaches of concept blending. On the one hand, we described how a Rich Background supports the discovery of input concepts to be blended. On the other hand, we showed how arguments promoting or demoting the values of an audience (to which the invention is headed) can be used to evaluate candidate blends.

We also showed how the evaluation of new blended concepts can be achieved by taking the computational theory of conceptual coherence due to Thagard (2000) into account. In this setting, newly invented concepts are evaluated with respect to a Rich Background conceptual knowledge so as to decide which of them are to be accepted into a system of familiar concepts.

We described two instantiations of the process model using two structured languages, namely, feature terms and description logics. This allowed us to capture the concept invention process in terms of well-defined operators such as least general

generalisation—for computing a generic space—and most general specialisation—for computing a blend. Pairs of input concepts are retrieved from a Rich Background by means of a discovery process that takes a similarity measure into account. Blending is realised according to the notion of amalgam, and blend evaluation is achieved by means of arguments, values and audience and conceptual coherence. An implementation of conceptual coherence presented in this chapter using the OWL API and Answer Set Programming is available at: `https://rconfalonieri@bitbucket.org/rconfalonieri/coinvent-coherence.git`.

We exemplified the computational framework in these two languages but the framework is general enough to be instantiated in other representation languages in which a subsumption relation between formulas or descriptions holds.

We aim at extending the current work from different perspectives. First, here, we presented a discovery method based on a similarity measure based on the structure of the refinement space, but other similarity methods, considering more nuanced aspects of the domain, are envisioned to be needed and useful. Particularly, having a subset of the concepts in a Rich Background activated as salient but lacking a clear second concept that can be used to yield an interesting blend is an interesting avenue of research.

Then, generating other kinds of arguments than the ones seen in this chapter, opens also a wide area of research related not only to computational argumentation, but also to human level argumentation. For instance, social arguments applying to an invented concept could be considered as an open-ended process—that is to say a collection of arguments that can always increase, since the members of an audience may change and the values (and their social prevalence) may also change in time. This, for instance, has also been the approach of Confalonieri et al. (2015) when modeling blend evaluation using Lakatosian reasoning. In this way, a given invented concept may, for instance, first be divisive and at later times reach an overlapping consensus in an audience (be it positive or negative). This open-endedness also highlights the relationship between subjective and social values in a given domain, in the sense that a large disagreement between a traditional (consensued) set of values of an audience and the idiosyncratic values of a creative agent should be able to model disruptive or groundbreaking inventions.

We aim at employing a richer DL, such as $\mathscr{SROIQ}$ (Horrocks et al., 2006), enacting the concept invention process, and allowing degrees of coherence and incoherence relations. Usually, coherence and incoherence are not treated only in binary terms, but it is also natural to take certain degrees of coherence or incoherence into account. This, for instance, has also been the approach of Joseph et al. (2010) when formalising deductive coherence.

Finally, we will need to discuss yet another important aspect of coherence theory, namely how to interpret the two parts of a coherence-maximising partition: the set of accepted and of rejected concepts. The information that a particular concept description falls in the set of accepted concepts or in the set of rejected concepts could also be taken into account to decide the acceptance or rejection of newly invented concepts; or even of already existing concepts in the Rich Background, in the light

of newly invented concepts. This aspect might become clearer as a wider range of concept representation languages is explored.

References

L. Amgoud and H. Prade. Using arguments for making and explaining decisions. *Artificial Intelligence*, 173:413–436, 2009.

K. Atkinson, T. Bench-Capon, and P. McBurney. Justifying practical reasoning. In *Proc. of the Fourth Workshop on Computational Models of Natural Argument (CMNA'04)*, pages 87–90, 2004.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

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.

M. A. Boden. Creativity. In M. A. Boden, editor, *Artificial Intelligence (Handbook of Perception and Cognition)*, pages 267–291. Academic Press, 1996.

B. Bonet and H. Geffner. Arguing for decisions: A qualitative model of decision making. In *Proc. of the 12th Conf. on Uncertainty in Artificial Intelligence (UAI'96)*, pages 98–105, 1996.

B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge University Press, New York, NY, USA, 1992. ISBN 0-521-41932-8.

R. Confalonieri, J. Corneli, A. Pease, E. Plaza, and M. Schorlemmer. Using argumentation to evaluate concept blends in combinatorial creativity. In *Proc. of the 6th International Conference on Computational Creativity, ICCC15*, pages 174–181, 2015.

R. Confalonieri, M. Eppe, M. Schorlemmer, O. Kutz, R. Peñaloza, and E. Plaza. Upward refinement operators for conceptual blending in $\mathscr{EL}^{++}$. *Annals of Mathematics and Artificial Intelligence*, 2016a. DOI: 10.1007/s10472-016-9524-8.

R. Confalonieri, E. Plaza, and M. Schorlemmer. A process model for concept invention. In *Proc. of the 7th International Conference on Computational Creativity, ICCC16*, pages 338–345, 2016b.

P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence Journal*, 77:321–357, 1995.

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, 2015a.

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, *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2015*, Lexington, KY, USA, September 27-30, 2015, pages 309–316, 2015b.

G. Fauconnier and M. Turner. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books, 2002. ISBN 978-0-465-08785-3.

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

J. Goguen and D. F. Harrell. Style: A computational and conceptual blending-based approach. In S. Argamon, K. Burns, and S. Dubnov, editors, *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*, pages 291–316. Springer, 2010.

J. A. Goguen and D. F. Harrell. Foundations for active multimedia narrative: Semiotic spaces and structural blending. *Available at* `https://cseweb.ucsd.edu/~goguen/pps/narr.pdf`, 2005. Last accessed, 2016.

M. Guhe, A. Pease, A. Smaill, M. Martínez, M. Schmidt, H. Gust, K.-U. Kühnberger, and U. Krumnack. A computational account of conceptual blending in basic mathematics. *Cognitive Systems Research*, 12(3-4):249–265, 2011.

D. F. Harrell. Shades of computational evocation and meaning: The GRIOT system and improvisational poetry generation. *6th Digital Arts and Culture Conference*, 2005. URL `http://groups.csail.mit.edu/icelab/sites/default/files/pdf/Harrell-DAC2005.pdf`.

D. F. Harrell. *Theory and technology for computational narrative: an approach to generative and interactive narrative with bases in algebraic semiotics and cognitive linguistics*. Ph.D. thesis, University of California, San Diego, 2007.

I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathscr{SROIQ}$. 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.

S. Joseph, C. Sierra, M. Schorlemmer, and P. Dellunde. Deductive coherence and norm adoption. *Logic Journal of the IGPL*, 18(1):118–156, 2010.

O. Kutz, J. Bateman, F. Neuhaus, T. Mossakowski, and M. Bhatt. E pluribus unum: Formalisation, use-Cases, and computational support for conceptual blending. In T. R. Besold, M. Schorlemmer, and A. Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, Thinking Machines. Atlantis/Springer, 2014.

G. Malcolm. *Software Engineering with OBJ: Algebraic specification in action*. Kluwer, 2000.

M. A. Kaliakatsos-Papakostas, R. Confalonieri, J. Corneli, A. I. Zacharakis, and E. Cambouropoulos. An argument-based creative assistant for harmonic blending. In *Proc. of the 7th International Conference on Computational Creativity, ICCC16*, pages 330–337, 2016.

T. Mossakowski, M. Codescu, F. Neuhaus, and O. Kutz. *The Road to Universal Logic—Festschrift for 50th birthday of Jean-Yves Beziau, Volume II*. Studies in Universal Logic. Birkhäuser, 2015.

S. Ontañón and E. Plaza. Similarity measures over refinement graphs. *Machine Learning*, 87(1):57–92, Apr. 2012.

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. ISBN 978-3-642-14273-4.

F. C. Pereira. *A Computational Model of Creativity*. Ph.D. thesis, Universidade de Coimbra, 2005.

F. C. Pereira. *Creativity and Artificial Intelligence: A Conceptual Blending Approach*. Mouton de Gruyter, 2007.

F. C. Pereira and A. Cardoso. The boat-house visual blending experiment. In *Proceedings of the 2nd Workshop on Creative Systems: Approaches to Creativity in AI and Cognitive Science. ECAI 2002, Lyon, France*, 2002.

F. C. Pereira and A. Cardoso. Optimality principles for conceptual blending: A first computational approach. *AISB Journal*, 1(4):351–370, 2003a.

F. C. Pereira and A. Cardoso. The horse-bird creature generation experiment. *AISB Journal*, 1(3):257–280, 2003b.

F. C. Pereira and A. Cardoso. Experiments with free concept generation in Divago. *Knowledge-Based Systems*, 19(7):459–470, 2006.

J. Pollock. How to reason defeasibly. *Artificial Intelligence Journal*, 57:1–42, 1992.

M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, Feb. 1991. ISSN 0004-3702.

M. Schorlemmer, R. Confalonieri, and E. Plaza. Coherent concept invention. In T. R. Besold, O. Kutz, and C. Leon, editors, *Proceedings of the Workshop on Computational Creativity, Concept Invention, and General Intelligence (C3GI 2016)*, Bozen-Bolzano, Italy, August 20-22, 2016, volume 1767 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

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.

G. Smolka and H. Aït-Kaci. Inheritance hierarchies: Semantics and unification. *Journal of Symbolic Computation*, 7(3–4):343—370, 1989.

P. Thagard. *Coherence in Thought and Action*. The MIT Press, 2000. ISBN 978-0-262-20131-5.

P. Thagard. *The Brain and the Meaning of Life*. Princeton University Press, 2010.

P. Thagard and T. C. Stewart. The AHA! experience: Creativity through emergent binding in neural networks. *Cognitive Science*, 35(1):1–33, 2011.

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. ISSN 0743-1066.

T. Veale and D. O. Donoghue. Computation and blending. *Cognitive Linguistics*, 11(3-4):253–282, 2000. DOI: 10.1515/cogl.2001.016.

T. Veale and M. Keane. The competence of sub-optimal theories of structure mapping on hard analogies. In *International Joint Conference in Artificial Intelligence*, pages 232–237, 1997.

G. A. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458, 2006.