A Process Model for Concept Invention

Roberto Confalonieri, Enric Plaza, Marco Schorlemmer
Artificial Intelligence Research Institute, IIIA-CSIC
Campus de la Universitat Autònoma de Barcelona (UAB)
E-08193 Bellaterra (Barcelona), Catalonia, Spain
{confalonieri,enric,marco}@iiia.csic.es

Abstract

In this paper, we propose a computational framework that models concept invention. The framework is based on conceptual blending, a cognitive theory that models human creativity and explains how new concepts are created. Apart from the blending mechanism modeling the creation of new concepts, the framework considers two extra dimensions such as 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. Throughout the paper, we exemplify the computational framework in the domain of computer icons.

Introduction

The cognitive theory of conceptual blending by Fauconnier and Turner (2002) models human creativity as a mental process according to which two input (mental) spaces are combined into a new mental space, called a blend. This theory, which was developed in the context of cognitive linguistics, posits that input mental spaces are somehow packaged by humans with the relevant information in the context in which the blend is created, and that blends are evaluated against some optimality principles (Fauconnier and Turner, 2002).

Existing computational models for concept invention — see the Related Work section 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 position paper, we claim that a computational model also need 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. These dimensions are justifiable if we think that there is no creation *ex nihilo* — thus, there is an origin — and there is usually a *purpose* in creating something new, and, consequently, there is a destination.

To this end, in this paper we propose to model concept invention by means of a process that consists of different sub-processes and components (Figure 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 a person's education, experience, and social circumstances. 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. 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.

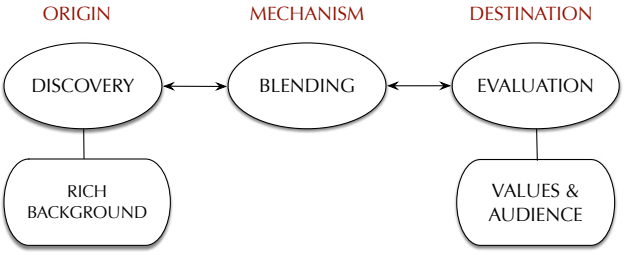


Figure 1: A process model for concept invention.

The above process model can be made more concrete in a domain such as *computer icon design*. In such a case, the Rich Background is what we can learn from, program about, specify of computer icons, such as a semiotic model of shapes, signs and relations between signs. This is understood as a finite and specific number of concepts given a particular set of icons (an icon library or a collection of libraries). Values, on the other hand, can be aesthetics such as simplicity or ambiguity, that matter for a specific type of

audience. These values serve to identify good icons that are created by the blending mechanism.

In the next section, we capture the process model above in terms of feature terms. This computational model is exemplified by means of a running example that shows the main processes that undergo the concept invention of new icons.

Related Work

Several approaches of formal and computational models for concept invention, inspired by the work of Fauconnier and Turner (2002), have been proposed.

Amalgam-based conceptual blending algorithms have been developed to blend CASL theories and $\mathcal{EL}^{++}$ concepts in (Confalonieri et al., 2015b; Eppe et al., 2015a,b). In these works, input spaces are assumed to be given. Good blends are selected by re-interpreting some optimality principles.

The Alloy algorithm for conceptual blending by Goguen and Harrell (2005) 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, 2005; Harrell, 2005, 2007). The input spaces of the Alloy algorithm are theories defined in the algebraic specification language OBJ (Malcolm, 2000). In the algorithm, input spaces are assumed to be given, hence there is no discovery. The optimality principles 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 atomic units, 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.

Divago, by Pereira (2007), is probably the first complete implementation of conceptual blending. The 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.

Finally, another work that relates to ours is (Confalonieri et al., 2015a). The authors use Lakatosian reasoning to model dialogues in which users engage to discuss the intended meaning of an invented concept. The main difference with the current work relies on 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., 2015a), an argument is a reason to refine the meaning of a blend and is provided by the user.

Computational Model

Rich Background

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 2b(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 2b(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 $\mathcal{C}$ of feature terms (Carpenter, 1992; Smolka and Aït-Kaci, 1989), each representing a concept. In this paper, feature terms are defined over a signature $\Sigma = \langle \mathcal{S}, \mathcal{F}, \leq, \mathcal{X} \rangle$, where $\mathcal{S}$ is finite set of sort symbols, including $\top$ and $\bot$, which represent the most specific and the most general sort, respectively; $\mathcal{F}$ is a finite set of feature symbols; $\leq$ is an order relation inducing an inheritance hierarchy such that $\bot \leq s \leq \top$, for all $s \in \mathcal{S}$; and $\mathcal{X}$ is a denumerable set of variables. Then, a feature term ψ has the form:

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

with $n \geqslant 0$, and where $x \in \mathcal{X}$ is called the root variable of ψ (denoted as $\mathsf{root}(\psi)$), $s \in \mathcal{S}$ is the sort of x (denoted as $\mathsf{sort}(x)$), and, for all j with $1 \leqslant j \leqslant n$, $f_j \in \mathcal{F}$ are the features of x (denoted as $\mathsf{features}(x)$) and the values Ψ_j of the features are finite, non-empty sets of feature terms and/or variables (provided they are root variables of feature terms ocurring 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.

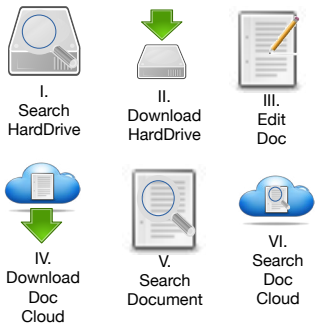
We choose to model icons as concepts represented by feature terms over the signature with the following sort hierarchy $\mathcal{S}$:[1]

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}

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

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

(a) Feature term representation of a computer icon.

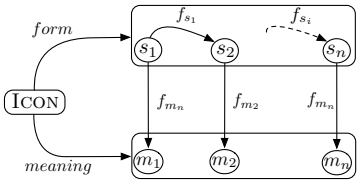


(b) Examples of computer icons.

Figure 2: Rich Background about computer icons.

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

In addition, feature terms representing icons need to be of the following form. A representation of the structure of an icon is presented below and its description follows.



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 $s \geq$ SIGN, and each meaning by means of feature terms whose root variable is of sort $s \geq$ 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 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 $s \geq$ ACTION, while those of the $objectType$ feature are of sort $s \geq$ OBJECTTYPE. Figure 2a shows the feature term representation of the icon in Figure 2b(I).

A fundamental relation between feature terms is that of subsumption ($\sqsubseteq$). Intuitively, a feature term ψ_1 subsumes another one ψ_2, or ψ_1 is more general than ψ_2, denoted as $\psi_1 \sqsubseteq \psi_2$, if all the information in ψ_1 is also in ψ_2.[2] We omit the formal definition of subsumption, which can be found in (Ontañón and Plaza, 2012) for feature terms as represented in this paper. The subsumption relation induces a partial order on the set of all features terms $\mathcal{L}$ over a given signature, that is, $\langle \mathcal{L}, \sqsubseteq \rangle$ is a poset.

[2]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.

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 computational model, an input space is a concept belonging to a library of concepts. The packaging of some relevant information corresponds to a discovery process that takes certain properties, which the blends need to satisfy, into account. In the creation of computer icons, we can imagine that an icon designer knows the meaning of an icon he wishes to create, but he ignores its form.

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:

$$\psi_q := x_1 : \text{ICON}\left[\ meaning = \left\{\begin{array}{l} x_2 : \text{SEARCH} \\ x_3 : \text{DOC} \end{array}\right\}\ \right] \quad (1)$$

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*.

The main idea behind the similarity-based search is that, for each icon concept ψ_i in the Rich Background, we measure how ψ_q and ψ_i are similar and, we use this measure to rank the results. The similarity between two feature terms can be defined by means of their *anti-unification* or *Least General Generalisation* (LGG) (Ontañón and Plaza, 2012).

Definition 1 (Least General Generalisation) *The least general generalisation of two feature terms ψ_1 and ψ_2, denoted as $\psi_1 \sqcap \psi_2$, is defined as the most specific term that subsumes both:* $\psi_1 \sqcap \psi_2 = \{\psi \mid \psi \sqsubseteq \psi_1 \wedge \psi \sqsubseteq \psi_2 \wedge \nexists \psi' : \psi \sqsubset \psi' \wedge \psi' \sqsubseteq \psi_1 \wedge \psi' \sqsubseteq \psi_2\}$.

The least general generalisation encapsulates all the information that is common to both ψ_1 and ψ_2 and, for this reason, is relevant for defining a similarity measure.

The least general generalisation can be characterised as an operation over a refinement graph of feature terms. The refinement graph is derived from the poset $\langle \mathcal{L}, \sqsubseteq \rangle$ by means of a generalisation refinement operator γ.

$$\gamma(\psi) = \{\psi' \in \mathcal{L} \mid \psi' \sqsubseteq \psi \text{ and } \nexists \psi'' \text{ s.t. } \psi' \sqsubset \psi'' \sqsubset \psi\}$$

The above definition essentially says that γ is an operation that generalises a feature term to a set of feature terms that is an anti-chain. The refinement graph, then, is a directed graph whose nodes are feature terms, and for which there is an edge form feature term ψ_1 to ψ_2, whenever $\psi_2 \in \gamma(\psi_1)$. We shall call *generalisation paths* all finite paths $\psi \xrightarrow{\gamma} \psi'$ in a refinement graph, and denote with $\lambda(\psi \xrightarrow{\gamma} \psi')$ its length.

Ontañón and Plaza (2012) describe a generalisation operator for feature terms that consist of:

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 variables of the term (a variable is removed only when the variable does not have any features);

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

We refer to (Ontañón and Plaza, 2012) for the formal details of the operator.

It is worthy noticing that, in case of variable equalities, it is not possible to define a generalisation operator that finds all possible generalisations of a feature term. However, for the purpose of defining a least general generalisation-based similarity, an operator which ensures that $\perp$ is reachable in a finite number of steps will suffice.

Example 1 (LGG example) *Let us consider the feature terms ψ_q in Eq. 1 and ψ_1 in Figure 2a. 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.*

As previously said, the least general generalisation of two feature terms $\psi_1 \sqcap \psi_2$ is a symbolic representation of the information shared by ψ_1 and ψ_2. It can be used to measure the similarity between feature terms in a quantitative way. The refinement graph allows us to estimate the quantity of information of any feature term ψ. It is the length of the (minimal) generalisation path that leads from ψ to the most general term $\perp$. Therefore, the length $\lambda(\psi_1 \sqcap \psi_2 \xrightarrow{\gamma} \perp)$ 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. Then a similarity measure can be defined as follows.

Definition 2 (LGG-based similarity) *The LGG-based similarity between two feature terms ψ_1 and ψ_2, denoted by $S_\lambda(\psi_1, \psi_2)$, is:*

$$\frac{\lambda(\psi_1 \sqcap \psi_2 \xrightarrow{\gamma} \perp)}{\lambda(\psi_1 \sqcap \psi_2 \xrightarrow{\gamma} \perp) + \lambda(\psi_1 \xrightarrow{\gamma} \psi_1 \sqcap \psi_2) + \lambda(\psi_2 \xrightarrow{\gamma} \psi_1 \sqcap \psi_2)}$$

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 algorithms for computing S_λ can be found in (Ontañón and Plaza, 2012).

Example 2 (Similarity example) *Let us consider the feature terms ψ_q in Eq. 1, ψ_1 in Figure 2a and their LGG in Example 1. Lengths $\lambda_1 = \lambda(\psi_1 \sqcap \psi_q \xrightarrow{\gamma} \perp) = 8$, $\lambda_2 = \lambda(\psi_1 \xrightarrow{\gamma} \psi_1 \sqcap \psi_q) = 12$, and $\lambda_3 = \lambda(\psi_q \xrightarrow{\gamma} \psi_1 \sqcap \psi_q) = 2$. Notice that λ_3 is very small (2 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 the 36% of the total information.

Given the above definitions, the discovery of concepts can be implemented by a discovery algorithm. The algorithm accepts a Rich Background of concepts $\mathcal{C}$, a query ψ_q, and the generalisation operator γ as input, and 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 rank them in a lexicographical order. The discovery returns a set of pairs of concepts $\langle (\psi_j, \lambda_j), (\psi_{j+1}, \lambda_{j+1}) \rangle$ in which $\lambda_j \geqslant \lambda_{j+1}$.

Blending

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 $\mathcal{L}$ for which a subsumption relation $\sqsubseteq$ between formulas (or descriptions) of $\mathcal{L}$ can be defined, together with an anti-unifier operation—playing the role of the generic space— and a unifier operation. Therefore, it can be defined for feature terms. We already defined the anti-unification of two feature term descriptions (Definition 1). Now, we proceed to define their unification.

Definition 3 (Unification) *The* unification *of two feature terms ψ_1 and ψ_2, denoted as $\psi_1 \sqcup \psi_2$, is defined as the most general term that is subsumed by both: $\psi_1 \sqcup \psi_2 = \{\psi \mid \psi_1 \sqsubseteq \psi \wedge \psi_2 \sqsubseteq \psi \ \wedge \nexists \psi' : \psi' \sqsubset \psi \wedge \psi_1 \sqsubseteq \psi' \wedge \psi_2 \sqsubseteq \psi'\}$.*

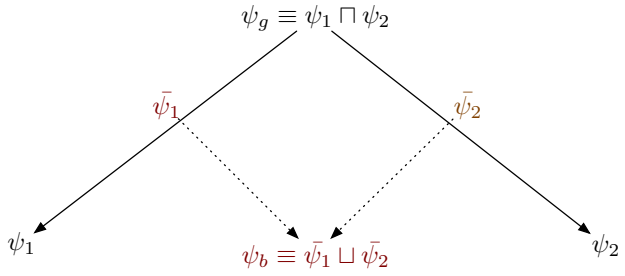


Figure 3: A diagram of a blend ψ_b from inputs ψ_1 and ψ_2.

Intuitively, a unifier is a description that has all the information in both the original descriptions. If joining this information leads to inconsistency, this is equivalent to say that $\psi_1 \sqcup \psi_2 = \top$, e.g., they have no common specialisation except 'none'.

An *amalgam* or *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 the purposes of this paper, we define an *amalgam* or *blend* of two input descriptions as follows:

Definition 4 (Blend) *A description $\psi_b \in \mathcal{L}$ 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) $\psi_b \equiv \bar{\psi}_1 \sqcup \bar{\psi}_2 \not\equiv \top$.*

The above definition is illustrated in Figure 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, e.g., 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) discuss optimality principles, however, these principles are difficult to capture in a computational way, and other selection strategies need to be explored.

We interpret blend evaluation in two steps. First, we discard those blends that do not satisfy a query ψ_q. Then, we order the blends by means of arguments, values and audiences in order to decide which blend is the best one.

Arguments, Values and Audiences

An argument is a central notion in several models for reasoning about defeasible information (Dung, 1995; Pollock, 1992), decision making (Amgoud and Prade, 2009; Bonet and Geffner, 1996), practical reasoning (Atkinson, Bench-Capon, and McBurney, 2004), and modeling 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 an abstract entity whose origin and structure are not defined, 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, aesthetic, etc.

Generally, there can be several values $\mathcal{V} = \{v_1, \ldots, v_k\}$. 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. For our purposes, we will consider values such as *simplicity* and *unambiguity*.

The main idea behind simplicity is that we want to estimate 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.

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

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:

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

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.

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

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 a preferences among these values. Given a set of values $\mathcal{V}$, there are potentially as many audiences as there are orderings on $\mathcal{V}$.

Definition 5 (Audience) *An audience is a binary relation $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ which is irreflexive, asymmetric, and transitive.*

We say that v_i is preferred to v_j in the audience $\mathcal{R}$, denoted as $v_i \succ_{\mathcal{R}} v_j$, if $\langle v_i, v_j \rangle \in \mathcal{R}$. We say that a value v_j covers v_i in the audience $\mathcal{R}$, denoted as $v_i \succcurlyeq_{\mathcal{R}} v_j$, if $v_i \succ_{\mathcal{R}} v_j$ and $\nexists v_{i'}$ such that $v_i \succ_{\mathcal{R}} v_{i'} \succ_{\mathcal{R}} v_j$.

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 pros promote a blend whereas arguments cons demote it. Given a set of blends $\mathcal{B}$, the tuple $\langle \mathcal{B}, \mathcal{V}, \Delta \rangle$ will be called a theory.

Definition 6 (Argument) *Let $\langle \mathcal{B}, \mathcal{V}, \Delta \rangle$ be a theory.*

- *An* argument pro *a blend b is a tuple $\langle (v, \delta), b \rangle$ where $v \in \mathcal{V}$, $\delta \in \Delta$ and $0.5 \leqslant \delta \leqslant 1$*
- *An* argument con *b is a pair $\langle (v, \delta), b \rangle$ where $v \in \mathcal{V}$, $\delta \in \Delta$ 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 $\succeq_{\mathcal{B}}$ on the set of candidate blends $\mathcal{B}$. The definition of this relation is based on the set of arguments pros and cons 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 pros or arguments cons;
- **Bipolar decision criteria:** they take both arguments pros and cons into account;
- **Meta-criteria:** they aggregate arguments pros and cons into a meta-argument.

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

Definition 7 *Let $b_1, b_2 \in \mathcal{B}$. $b_1 \succeq_{\mathcal{B}} b_2$ if and only if $|\mathcal{M}_p(b_1)| \geqslant |\mathcal{M}_p(b_2)|$.*

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

In the above definition, $\succeq_{\mathcal{B}}$ depends on the relation $\succ_{\mathcal{R}}$. Since $\succ_{\mathcal{R}}$ is a preference relation, some of the values of the arguments can be incomparable. Consequently, b_1 and b_2 will not be comparable neither. This definition can be relaxed, for instance, by ignoring these arguments.

The counter-part decision criteria of Definitions 7-8 for the case of arguments cons can be defined in a similar way and we omit them.

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

Definition 9 *Let $b_1, b_2 \in \mathcal{B}$. $b_1 \succeq_{\mathcal{B}} b_2$ if and only if $|\mathcal{M}_p(b_1)| \geqslant |\mathcal{M}_p(b_2)|$ and $|\mathcal{M}_c(b_1)| \leqslant |\mathcal{M}_c(b_2)|$.*

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 pros and cons into a meta-argument. Then, comparing two blends amounts to compare the resulting meta-arguments. A simple criterion can be defined by aggregating the degrees of the arguments associated with a blend.

Definition 10 *Let $b_1, b_2 \in \mathcal{B}$. $b_1 \succeq_{\mathcal{B}} b_2$ if and only if*

$$\sum_{\alpha \in \mathcal{M}(b_1)} \mathsf{Deg}(\alpha) \geqslant \sum_{\alpha' \in \mathcal{M}(b_2)} \mathsf{Deg}(\alpha')$$

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

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

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 11 *Let $b_1, b_2 \in \mathcal{B}$. $b_1 \succeq_{\mathcal{B}} b_2$ if and only if*

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

This definition also guarantees that all the blends are comparable.

The Model at Work

Let us imagine an agent that has access to a Rich Background $\mathcal{C} = \{\psi_1, \psi_2, \psi_3, \psi_4\}$ consisting of four of the icons depicted in Figures 2b(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. 1), and an audience, that is, a preference order over the values. For the sake of this example, we assume that $\mathsf{Simplicity} \succ_{\mathcal{R}} \mathsf{Unambiguity}$.

The discovery retrieves the following pairs of concepts:

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

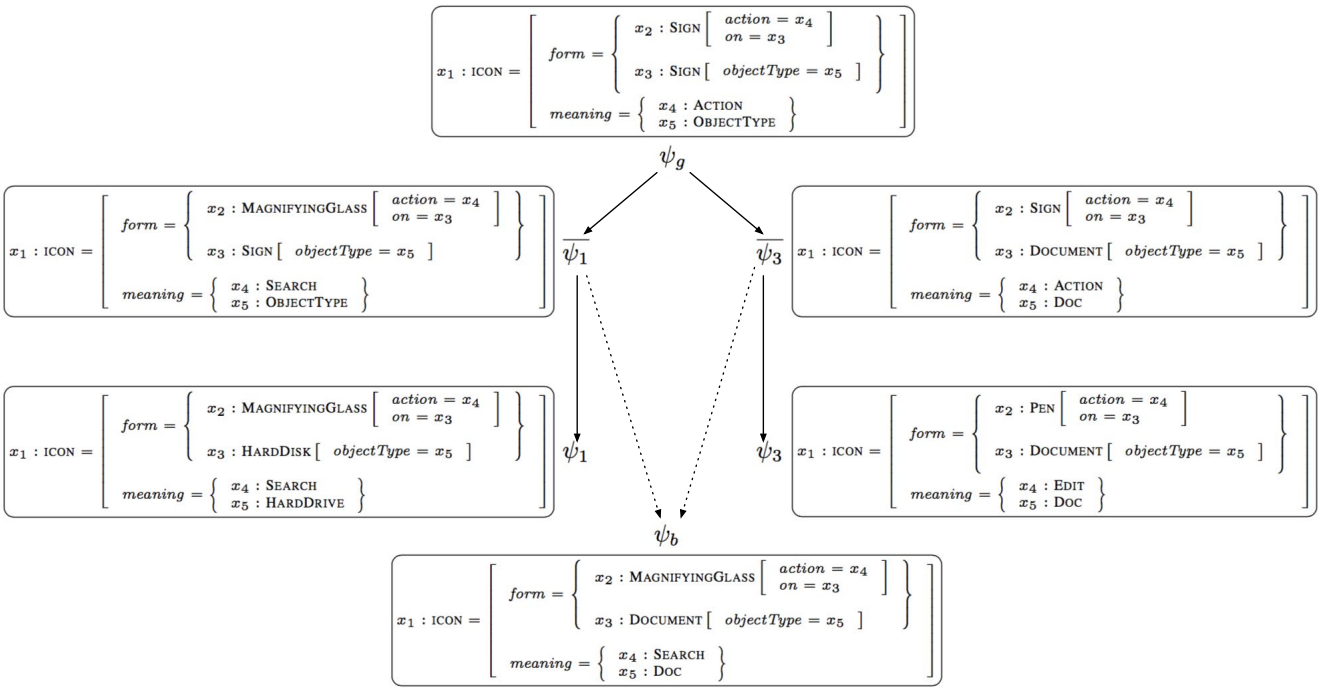


Figure 4: Amalgam-based blending of feature terms ψ_1 and ψ_3.

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

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

The agent proceeds to blend the first pair in the list. To this end, it applies the amalgam-based blending. 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 agents 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, is discarded, and only ψ_{b_1} is kept. The creation of ψ_{b_1} is illustrated in Figure 4.

The agents 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. We denote this blend as ψ_{b_2}. The set of blends is $\mathcal{B} = \{\psi_{b_1}, \psi_{b_2}\}$. A representation of ψ_{b_1} and ψ_{b_2} is given in Figures 2b(V-VI).

The agent evaluates these blends by means of the arguments and values described in the previous section. 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 pros (or cons) (Definition 7), and when considering both arguments pros and cons (Definition 9). Instead, ψ_{b_1} is preferred to ψ_{b_2} when using the criteria that take the audience into account (Definitions 8 and 11).

Conclusion and Future Work

In this paper, 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 sub-processes—discovery, blending, and evaluation—that together account for concept invention. We proposed its computational model in terms of feature terms, a formal knowledge representation language. This allowed us to capture the concept invention process in terms of well-defined operators such as anti-unification—for computing a generic space—and unification—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.

We exemplified the computational framework in the domain of computer icon design, but the framework is general enough to be used in other domains such as music or poetry generation. We plan to explore the use of arguments, values and audiences as a means to evaluate concept blends in such domains as future work.

We also aim at extending the process model by including the notion of coherence by Thagard (2000). Coherence theory, when used to explain human reasoning, proposes that humans accept or reject a cognition depending on how much it contributes to maximising the constraints imposed by situations or other cognitions. In the case of concept invention, coherence can be defined and used, for instance, to measure to what extent a blend coheres or incoheres with the Rich background and other blends.

Acknowledgments

This work is partially supported by the COINVENT project (FET-Open grant number: 611553).

References

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

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

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

Besold, T. R., and Plaza, E. 2015. Generalize and Blend: Concept Blending Based on Generalization, Analogy, and Amalgams. In *Proc. of the 6th Int. Conf. on Computational Creativity, ICCC15*.

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

Carpenter, B. 1992. *The Logic of Typed Feature Structures*. New York, NY, USA: Cambridge University Press.

Confalonieri, R.; Corneli, J.; Pease, A.; Plaza, E.; and Schorlemmer, M. 2015a. Using Argumentation to Evaluate Concept Blends in Combinatorial Creativity. In *Proc. of the 6th Int. Conf. on Computational Creativity, ICCC15*.

Confalonieri, R.; Eppe, M.; Schorlemmer, M.; Kutz, O.; Peñaloza, R.; and Plaza, E. 2015b. Upward Refinement for Conceptual Blending in Description Logic —An ASP-based Approach and Case Study in $\mathcal{EL}^{++}$. In *Proc. of 1st Int. Workshop of Ontologies and Logic Programming for Query Answering.* Co-located with IJCAI-2015.

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

Eppe, M.; Confalonieri, R.; Maclean, E.; Kaliakatsos-Papakostas, M. A.; Cambouropoulos, E.; Schorlemmer, W. M.; Codescu, M.; and Kühnberger, K. 2015a. Computational Invention of Cadences and Chord Progressions by Conceptual Chord-Blending. In *IJCAI 2015*, 2445–2451.

Eppe, M.; Maclean, E.; Confalonieri, R.; Kutz, O.; Schorlemmer, W. M.; and Plaza, E. 2015b. ASP, Amalgamation, and the Conceptual Blending Workflow. In *Logic Programming and Nonmonotonic Reasoning - 13th Int. Conf., LPNMR 2015, Lexington, KY, USA, September 27-30, 2015.*, 309–316.

Fauconnier, G., and Turner, M. 2002. *The Way We Think: Conceptual Blending And The Mind's Hidden Complexities*. Basic Books.

Goguen, J. A., and Harrell, D. F. 2005. Foundations for active multimedia narrative: Semiotic spaces and structural blending. Manuscript to appear published in the journal "Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems".

Goguen, J. 1999. An introduction to algebraic semiotics, with application to user interface design. In Nehaniv, C. L., ed., *Computation for Metaphors, Analogy, and Agents*, volume 1562 of *LNCS*. 242–291.

Harrell, D. F. 2005. Shades of computational evocation and meaning: The GRIOT system and improvisational poetry generation. *6th Digital Arts and Culture Conference*.

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

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

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

Ontañón, S., and Plaza, E. 2010. Amalgams: A Formal Approach for Combining Multiple Case Solutions. In *Proc. of the Int. Conf. on Case Base Reasoning*, volume 6176 of *Lecture Notes in Computer Science*, 257–271. Springer.

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

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

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

Thagard, P. 2000. *Coherence in thought and action*. The MIT Press.

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

Veale, T., and Keane, M. 1997. The competence of sub-optimal theories of structure mapping on hard analogies. In *IJCAI*, 232–237.