Available online at www.sciencedirect.com
ScienceDirect
Knowledge-Based Systems 19 (2006) 459–470
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

Experiments with free concept generation in Divago

Francisco C. Pereira *, Amílcar Cardoso

Creative Systems Group, AILab, Centro de Informática e Sistemas da Universidade de Coimbra (CISUC), Departamento de Engenharia Informática, Polo II, Pinhal de Marrocos, Coimbra, Portugal

Received 23 January 2006; accepted 15 April 2006
Available online 21 June 2006

Abstract

This paper presents a set of experiments we carried out with, Divago, a system that is an attempt to implement our ideas towards a computational model of creativity. It is expected to be able to generate novel concepts out of previous knowledge. Here we show its behaviour with a large dataset constructed independently by other researchers consisting of over 170 nouns (for a project named C^3). Each noun is represented with a syntax that is equivalent to the one adopted for Divago. We apply a two step experimentation procedure, which starts by ''training'' the system with ''preferred outcomes'' and then allowing it to do free generation, constrained by the pragmatic goal of a given query. We evaluate the results and make a short discussion regarding well-defined criteria of novelty and usefulness. We also present a comparison with a similar experiment done with C^3.

1. Introduction

Divago aims to implement a set of principles suggested by our model of computational creativity (see [11]). It is expected to be able to generate new and useful concepts out of its knowledge base composed, among other things, of previously known concepts. Those newly generated concepts should not consist simply of the composition of previous concepts, instead it is desirable that they have an emergent structure of their own. In other publications, we have been focusing on several aspects of the model, by defining it formally [12,11], discussing its applicability to creativity [11,14], demonstrating its performance with specific pairs of domains [11,13,15]. In this paper, we focus the experimentation phase,[1] with particular validation concerns, namely by using a much larger dataset with concepts made by others, and evaluating the results according to novelty and usefulness. For both of these evaluating criteria, we propose fairly simple yes/no criteria. A concept is novel if it is not equal to any from the knowledge base and it is useful if it accomplishes a set of pragmatic conditions that may be specific to a situation (e.g. it should be a solid blue object that serves to make food) or a generic demand for an application (it should be an object with a single color and a single shape).

The dataset we use was borrowed from Fintan Costello's [1] validation work for the C^3 model of noun–noun conceptual combination and it contains over 170 different concepts. C^3 was capable of generating interpretations for noun–noun combinations, and we also had access to data from an experiment analogous to the one presented in this paper.

As we will discuss, our mapping mechanism based on structure alignment constrains the resulting concepts to the hybrid and property types of noun–noun combination. This means, for example, that it can generate ''vegetable person'' to be the concept of ''person'' (the head) modified by the concept of ''vegetable'' (the modifier), which could be an ''inanimate, static person''. A different interpretation (that our mapping is not allowing yet) could be that a ''vegetable person'' is a ''person that sells vegetables'', or

* Corresponding author.
E-mail addresses: camara@dei.uc.pt (F.C. Pereira), amilcar@dei.uc.pt (A. Cardoso).

[1] Due to publishing timings, we inform the author that the experimental material here presented is also part of the PhD thesis [11] from the same author.

0950-7051/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2006.04.008

"someone that really likes vegetables". These interpretations (classified as *relational* because constituent concepts do not merge or transfer properties, rather there is a specific relation between them) would demand a different mapping mechanism.

In the experiments, we seek to understand the behaviour of the system with regard to our parameters of novelty and usefulness as well as assessing aspects like predictability and consistency of the results. Moreover, we want to observe the role that a specific kind of structures, the frames, have in the achievement of the results. In general terms, we think that the applicability of Divago can take several forms, namely in situations where divergent solutions are welcome. Examples of such are applications in the domains of arts and games. In fact, we are currently developing a game environment that applies Divago as an object generator.

In the second section, the reader will hopefully become familiar with the aspects of Divago that are important for the experiments. In Section 3, we will give an overview of the dataset used. The experiments are the core of this paper and are presented in Section 4 and the paper is finished with some conclusions and further work.

We assume that the reader has some familiarity with Conceptual Blending. This does not mean that we do not explain the concepts involved, but it may become helpful to inspect some Conceptual Blending references [3,4,14].

2. The Divago system

The Divago system always works with two input domains (normally called input domains 1 and 2) and a generic space domain. Each of these are part of its knowledge base. It starts by establishing a mapping between the input domains (normally also recurring to knowledge of the generic domain) and then, with these domains and the mapping, it tries to generate a fourth domain, called "blend". Domains are our formal realization of Fauconnier and Turner's *mental spaces* [3]. The module that generates the blend is called Factory and it uses a parallel search method to explore the (usually large) search space. In order to guide itself, the Factory is supported by the Constraints module, which provides a set of evaluation criteria. A module dedicated to the Elaboration of blends, which is fundamental for the emergence of novel structure, will be present in next developments.

2.1. Knowledge base

The knowledge base is divided into domains. Each domain has several different types of knowledge: a Concept Map; a set of Instances; a set of Rules; a set of Integrity Constraints and a set of Frames. A concept map is a semantic network that declares the relationships that exist between specific concepts of the domain (*elements* of the mental space, in Fauconnier and Turner's literature). An instance corresponds to specific examples of the domain (e.g. in the domain of "house", a specific description of a house would be an instance). A rule represents procedural knowledge that is valid within the domain (e.g. If X has wings, then it can fly). Integrity constraints are rules with a *false* conclusion, i.e., if the premisses are satisfied, then we say that "an integrity constraint was violated". These are used to state facts that are not expected to happen simultaneously (e.g. an object X cannot be solid and liquid at the same time). Finally, frames are the more complex structures we use in the knowledge base. A frame corresponds to a set of conditions associated to a given *meta*-concept (this may recall the notion of Image Schemas [6], well known in the field of Cognitive Linguistics). A frame can be as concrete as a set of characteristics to identify a physical object (e.g. "something made of plastic, with a roller ball point, used to write" can be associated to the (meta-)concept of *pen*) or as abstract as a set of directives for self organization during the blend construction in the Factory module (e.g. "a new blend should maintain the same set of relations that exist in the first input concept"). Thus, these sets of conditions may describe specific patterns that the result should have (e.g. X should have a specific property), general patterns (e.g. every concept should have a color, a shape, a name) or abstract guidelines for the construction of the result (e.g. the result should have the structure of the first input concept and the elements of the second). When all conditions of a frame are satisfied in a concept map, then we say that it *accomplishes* the frame.

Frames allow the use of a general purpose programming language (in our case, Prolog) as well as the language of the concept maps (binary predicates), projections (in ternary predicates) and special operators (e.g. op(exists(L)) checks if the relations or projections in list L are present in the concept map). Below, we show two simple frames. The first one, "haunted", says that something is haunted if it contains something that causes fear and is magic or mysterious, i.e., it identifies whether a concept X is haunted.

$$haunted : property(haunted, X) \leftarrow contain(X, Y) \wedge cause\_effect(Y, fear) \wedge attribute(X, [magic, unknown])$$

Differently, the frame for "shape transfer" is an abstract guideline that, if satisfied, transfers the shape of input domain 1 to input domain 2, in the result (e.g. in "necklace paper", the result will be a paper with the circular shape of the necklace).[2]

$$shape\_transfer : shape\_transfer(A, B, X) \leftarrow \{stats(domain1, A), stats(domain2, B), findall(shape/B/X, rel(A, A, shape, X), L)\} \wedge \{op(exists(L))\}$$

[2] The predicates *stats* identify the input domains, $rel(A, A, shape, X)$ identifies the shape relations in domain A, the *findall* serves to construct a list with the *shape* predicates expected to exist in the result.

Table 1
The frames used in the experiments

Frame	Description
bframe	The blend has the same relations of head noun (although the arguments may differ)
bcore	The blend has the same relations and arguments (except those related to function) of head noun
analogy_transfer	Transfer all neighbor elements and relations of an element of modifier to the mapping correspondent of head
function_substitution	A function from head is substituted by a function of modifier
single_differentiating_feature	Head and modifier differ only on one feature, which is transferred to head
function_transfer	The head gains a function that was part of the modifier
shape_transfer	The head gains the shape of the modifier
structure_transfer	The head gains the structure of the modifier
slot_set_completion	The slots in head that did not have a value are filled with modifier's corresponding values
feature_set_contrast	The feature-set in the head are replaced by the feature-set of the modifier

Frames end up being a meta-language for conceptual integration we may use for specifying our intentions in the results. As we will show below, if no such specification is given, the frames function as attractor structures that may compete or cooperate until reaching stability. Although frames are essential to the present experiments, there is no place to explain them in higher detail, so we provide the reader the superficial description for the used frames (in Table 1). Notice that these frames are generic in the sense that they may be (and actually have been) used in other completely different experiments (see [16,17]).

2.2. Mapping engine

The mapping engine establishes a correspondence between elements of the input domains. Currently, this correspondence is found by a structure alignment algorithm that is very similar (although simpler) to the one used by Tony Veale in his Sapper framework [18]. Sapper proposes metaphor interpretations through the establishment of cross-domain mappings (e.g. "If Surgeon is Butcher, then scalpel is cleaver, etc."). In [19], the reader can find a comparison of Sapper with other structure alignment algorithms. To understand the present paper, the reader should retain that we find a partial 1-to-1 mapping correspondence between elements of the input domains and that this mapping is based on identity of relational structure (i.e. if A, from input domain 1, is mapped to B, from input domain 2, and A has a relation R to C and B has a relation R to D, then C can be mapped to D).

In order to find these correspondences, the mapping engine may need to use knowledge from the generic domain, which contains an ontological "isa" hierarchy as well as many other knowledge structures (such as its own concept map).

The mappings are used later, in the concept generation, to constrain the possible projections a specific element can have in the result. More specifically, if x is mapped to y, it can be projected to x, y or *nil* in the result.[3] If x has no mapping counterpart, it can only be projected to x or *nil*. For example, if "necklace" is mapped to "paper", the result may contain a "paper" that is the projection of "necklace" and so it may bring all its surrounding elements (like being "circular") to the concept of "paper". On the other hand, if "paper" is projected to "paper", then the concept of "paper" will maintain its original features. In Fig. 1, we show a very simplified diagram that includes these projections.

The projections are selective, which means some elements get projected, some do not (i.e. their projection is *nil*). The ones that are projected can also have many choices (as shown in the preceding paragraph). The selection of the projection obeys very strict rules and methods, a task undertaken by the Factory and Constraints modules.

2.3. Factory

The factory is the module that takes most of the processing time of the system. Its goal is to explore the space of all possible combinations of projections of the input domains. We say elsewhere [15] that the size of this space may reach a maximum of $4^{2k} \times 2^{l-2k}$ different combinations,[4] with $l = m + n$ and $k = min(m,n)$ and m and n the number of concept map elements belonging to input domains 1 and 2, respectively. But, since we are allowing three projections instead of four (thus leaving out $x|y$), the maximum is reduced to $3^{2k} \times 2^{l-2k}$. This means that, for example if $m = 5$ and $n = 7$ (two very small concept maps), we may have 60466176 different solutions to evaluate.[5] This evaluation is achieved by a set of criteria provided by the Constraints module.

Given the size of the search space and the absence of any algorithm able to get the "best solution" through a sequence of deterministic steps, we decided to build a

[3] In [12], we allow a fourth alternative – the compound – represented as $x|y$, but currently we are not applying this possibility, since it raises problems of ambiguity in interpretation.

[4] This maximum is calculated for the largest possible 1-to-1 mapping. When there is no mapping (e.g. simple *copy* + *paste* of the inputs into the blend), the size of space will correspond to the minimum of 2^l different combinations.

[5] For very simple situations, like in Fig. 1, the space is considerably smaller, since we have only two mappings and $m = 5$ and $n = 4$, yielding $3^4 \times 2^5 = 2592$ different combinations. However, this is a very simplistic diagram.

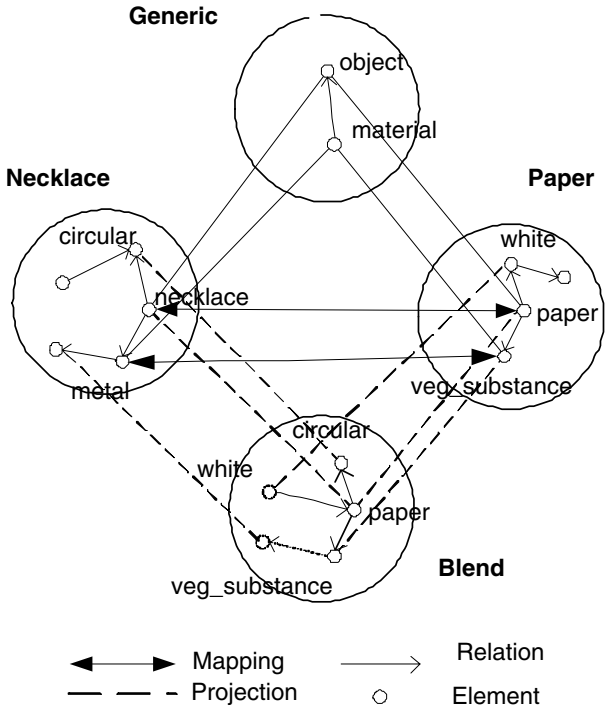

Fig. 1. Blending "necklace" and "paper".

genetic algorithm (GA). A GA makes a parallel search in the space and does not guarantee the achievement of *global maxima*, but, when its parameters are correctly chosen, it may be able to cover a wide area and obtain satisfactory results. Generally, a GA generates sets of *individuals* (the possible solutions to the problem being solved), each set being called a *generation*. From each generation, it selects some individuals (there are many methods to achieve this) and applies *operators* (e.g. *crossover*, *mutation* and others) to obtain the subsequent generation. It repeats these steps until achieving a *stopping condition* (e.g. a fixed number of generations, a threshold value). It is outside the scope of this paper to describe further details of GA's. For those interested, we advise the reading of a specialised book on the subject (e.g. [5]).

In our GA, we *evolve* projection combinations, thus each individual is described by a set of projections, one for each element of the input domain. With the projection of all elements of both input domains, it is straightforward to obtain the concept map of the blend.[6] The evaluation function we use (also called *fitness function*) consists of a weighted sum of the eight optimality principles enunciated in the Conceptual Blending framework [4] and is applied to the concept map of the blend. In [16], we formalize our implementation of these principles, which are the subject of the Constraints module. The algorithm stops after reaching one of three conditions: being at the nth generation (n is equal to 500, currently); achieving an individual with the value of 1 (the fitness function is normalized to fall within the 0..1 interval); not being able to improve results for more than m generations (m currently equal to 30). The latter means the algorithm has stabilized around a specific individual (or set of individuals with the same value). Thus, when we say that "the evolution stabilized at the ith generation", we mean that in generation $i + 30$ the value of the best individual remains the same since the ith generation.[7]

2.4. Constraints

In this module, we implemented the optimality principles (also called optimality constraints) according to our own formalization [16,11]. These constraints serve to evaluate a blend according to several aspects, from specific structural (like maximizing the number of specific relations) and pragmatic conditions (like respecting a query) to general guidelines (maintaining topology). Currently, our optimality constraints are applied solely to the concept map (of the blend) and we hope to extend them to evaluate other aspects of the blend (e.g. the instances), in future developments. Three of the eight optimality constraints suggested in [4] were not applied in these experiments: Web (because it is not an independent constraint); Pattern Completion (because we are not applying the completion and elaboration phases) and Intensification of Vital Relations (because we are only applying one kind of mapping – it would always yield the same value). We give now an informal description of the implementation of the optimality principles we use in the Constraints module:

- Integration. The Integration value of a blend depends on its accomplishment of frames. For example, if a single frame contains all relations and elements found in the blend, it will have the highest Integration value (1). The intuition behind this is that one recognizes a frame as a whole, meta-level concept. For example, a blend that has exactly the three relations of the frame "haunted" is considered as being totally integrated around the concept of "haunted thing" (or in other words, it fits the pattern of what a "haunted thing" should consist of). The same reasoning could be applied for the "shape_transfer" frame (a blend exactly integrating this frame would correspond to the redefinition of the "shape" of the target concept). On the other side, we penalize the multiplicity of frames, i.e., when a blend accomplishes several different frames (each one having different conditions), it will have less Integration value than if it were a single frame with the same conditions. Our Integration measure also penalizes "free" relations

[6] In GA literature, one could see the projections as the *genotype* of the individual and the corresponding concept map as the *phenotype*.

[7] When the evolution stabilizes in this way, and after a given set of non-changing generations (currently $m/2$) Divago increases the number of random individual generations. This process occasionally allows sudden jumps, although generally rare.

(relations that do not fit in any frame) and integrity constraint violations. Given its importance, Integration has a weight of 30% in the fitness function.

- Topology. Topology is the main principle that brings inertia to the process because it is centered on maintaining the original relationships between elements. Its measurement is fairly simple and consists on finding the ratio of elements in the blend that maintain the (exact) same neighborhood relations. Its weight in the experiments is 5%.
- Maximization of Vital Relations. Fauconnier and Turner propose a set of *vital relations*, which have a special role within the domains (called inner-space relations) and in establishing inter-connections (outer-space relations). In our case, we allow the customization of these relations. In the present experiments, these correspond to the relations *property*, *shape*, *pw*, *made_of*, *found*, *color* and *connects*, which we may find in the dataset configuration. The Maximization of Vital Relations constraint measures the ratio of vital relations that exist w.r.t. to the input domains. Its weight in the experiments is 5%.
- Unpacking. The ability to unpack the blend, i.e., to reconstruct its connections with the input domains and mappings, is measured in our model by finding subparts of the blend that are identical to subparts of one of the input domains. The intuition is that we are more able to make the reconstruction if we identify patterns that explicitly connect the blend with the input.[8] We are aware that this identification is not simply based on repetition of patterns, a fact we wish to consider in future developments. Unpacking takes 15% of the weight on the fitness function.
- Relevance. Relevance measures whether the blend respects a given query. This query is what specifies the usefulness of the results. A query consists of a set of relations the blend is expected to have and/or a set of frames it is expected to accomplish. As in the Integration measure, Relevance is penalized by integrity constraint violations. The Relevance constraint is the most important for our experiments (it has a weight of 45% on the fitness function) since it favours the accomplishment of the established goal.

As we could see, apart from the concept maps and mappings, there are two fundamental structures we apply in evaluating a blend: the frames and the integrity constraints. The former stimulates some interconnections of elements, the latter penalizes them.

It is important to add that the choice of the weights for the fitness function was decided taking into account two aspects: the intuitive importance of each measure, as described above; the empirical results we have been observing from many experiments (e.g. the ones presented in [15]).

[8] This is particularly important in situations like advertising or humor, in which one must find the connections to the subject to understand the message correctly.

Finally, the reader should retain the strong connection between Relevance and Usefulness. The Relevance constraint allows the system to be configured towards specific goals and, from our point of view, the accomplishment of these goals will determine whether a blend is useful or not.

3. The dataset

The correct functioning of Divago highly depends on Knowledge richness of its inputs and generic space. Our first direction was towards the freely available ontologies such as WordNet [10] or CYC [8], however the former is rich in quantity of nouns, but extremely poor in terms of their interrelations (there are essentially taxonomical relationships) while the latter demands great effort to process. With the obvious relationships between C^3 and Divago, we found its own database as a perfect choice, as it has many different kinds of relations for each noun, and the representation is very simple.

The dataset comprises 179 noun descriptions borrowed from Fintan Costello's PhD thesis [1] on noun–noun conceptual combination. In this thesis (and in subsequent publications [2]), the author describes each noun by a set of attribute-value pairs, as shown below (for ''necklace'').

Necklace

name:	(necklace)
feature-set:	(solid inanimate static)
color:	(silver gold)
shape:	(small circular)
structure:	
made_of:	metal
parts:	(pendant)
found:	
function:	((wears person3 necklace neck) (decorates necklace person3))

The conversion to our concept maps is straightforward: each of the ''features'' becomes a property relation; the attributes ''color'', ''shape'' and ''made_of'' become relations with the respective name; each of the ''parts'' is converted into a ''pw'' (part whole) relation; each ''function'' is converted into a set of ''actor'' and ''actee'' (with third arguments, such as ''place'' or ''instrument''). The ''actor'' is expected to be the first argument of the function, while ''actee'' is the second. Therefore, our concept map representation for ''necklace'' is as follows:

property(necklace, solid)	made_of(necklace, metal)
property(necklace, inanimate)	pw(pendant, necklace)
property(necklace, static)	actor(wears, person3)
color(necklace, silver)	actee(wears, necklace)
color(necklace, gold)	place(wears, neck)
shape(necklace, small)	actor(decorate, necklace)
shape(necklace, circular)	actee(decorate, person3)

In the original dataset, there are interrelationships between nouns. For example, there is also a representation for "pendant", "person3" and "neck" so, along with "necklace", these nouns can be seen as a small graph representing the knowledge about people and necklaces. Within this small graph, there is normally no repetition of function specifications (e.g. in "neck" or "person3" representation, there is no "wears" function, although it exists implicitly). We converted directly and separately each noun to a concept map, and there is no communication between our concept maps, which means that many nouns in our knowledge bases lose their original implicit data. Another aspect of the dataset is that some concepts have several different instantiations (e.g. "person3" is the third representation of the noun "person"). We also converted these directly and separately to our knowledge base, without merging them.

4. Experiments with free-concept generation

The main goal of these experiments was to observe how Divago behaves with respect to criteria of novelty and usefulness. An important condition we put for these tests was that data should not be constructed by anyone involved in the project, so that the results were not in any way influenced by unconscious biasing or tailoring. The C^3 dataset seemed to us a perfect choice, given that we could not find easily a relatively big database with knowledge coded in such a compatible representation.

The noun–noun interpretations we consider in the experiments are either *hybrid interpretations*, in which the resulting concept is a blend of both concepts being combined (as in "a *drill screwdriver* is a two-in-one tool with features of both a drill and a screwdriver" [20][9]) or *property interpretations*, in which the head gains one (or more) property of the modifier (as in "an elephant pencil is a very large pencil"). In some tests we made prior to the ones presented here, the mapping we use (based on structure alignment) was clearly unable to allow other types of interpretations such as relational and known-concept interpretations, which corroborates the thesis that conceptual combination is *not* structure alignment [7].

In order to provide a pragmatic background for the experiments, we invite the reader to consider a game, in which its elements (objects and characters) are defined by *scripts* with the same syntax of the nouns described above. In this context, a *useful* concept must have specific values for the slots of the script and respect a set of integrity constraints. The slots and values required can thus be grouped together in a query. In all experiments (except in the training set), this query consisted of:

property(A,[animate, inanimate]),
property(A,[liquid, solid]),
property(A,[static, mobile]),
made_of(A,_),
shape(A,_), color(A,_),
actor(F,_), actee(F,_)

Square brackets mean disjunction (e.g. the concept A must be "animate" or "inanimate") and underscores mean "anything" (e.g. the concept A must have a "color"). The presence of "actor" and "actee" relations means that the concept should have a *function*.

Integrity constraints specify impossibilities within the game (e.g. a concept cannot have two "shapes" at the same time). A *novel* concept is a concept that is not present in the knowledge base.

Beforehand, we could not know exactly what kinds of frames were needed to build "good" combinations, leading to the need of a supervised training phase that helped us find a set of appropriate frames. Only after this training, we were able to test the system, leaving it to construct its own concepts.

About the configuration of the GA, we must say that each population has 100 individuals and we are applying the crossover (for 80% of the new individuals) and mutation (14%) operators. The (5%) top best individuals are also kept for the next generation and the rest is stochastically selected favoring fitness value and diversity. One (1%) random new individual is generated for each population.

The weights are as described above and were chosen after an extensive set of tests, some of which described in [15].

4.1. Training

The training set we used consisted of 30 pairs of randomly selected nouns from the list. For each one, we constructed a solution (called the *training target*) correspondent to our own interpretation of the noun–noun combination. This hybrid interpretation considered exclusively the knowledge contained within the selected noun representations and was centered on the head noun, which means that, in any pair A–B of nouns, the interpretation was that "an A–B is a B with such and such A characteristics". In other words, the concept B is always the *focal concept* in our interpretations.

Each experiment consisted of making 30 runs for each pair (each run with the exact same starting conditions), having in the query the set of frames we expect could achieve the target. When the results differed from the target more than acceptable, we either selected other frames or designed new ones and made the 30 runs again. More specifically, this happened when there was

[9] Cited in [2].

Table 2
Excerpt of the training set (average error to target = 0, 67)

Combination	Target interpretation	Error	Frames
bullet potato	Small and cylindrical potato	2	bcore, shape_transfer, slot_set_completion
cow vehicle_body	Black and white vehicle that eats grass	0	bcore, function_transfer, slot_set_completion
eagle shirt	Brown, bird-shaped shirt	1	bcore, structure_transfer, shape_transfer
engine ball	Self-mobile ball	0	bcore, feature_set_contrast
head hammer_handle1	Mobile and animate (living) hammer handle	2	bframe, feature_set_contrast
flower_bloom plant	Spherical plant	0	bframe, shape_transfer
fruit1 paper1	Paper with fruit-seeds that humans eat	0	bcore, structure_transfer, function_transfer
neck instrument	A small and straight instrument	0	bcore, shape_transfer
necklace paper	Circular paper that people use in the neck for decoration	1	bcore, function_transfer
patient paper1	Paper that has illness	0	bcore, function_transfer
pen person	Thin, long person, that is used (by others) to write on paper	2	bcore, function_transfer
pencil pendant	Thin, long pendant, used to write on paper	0	bcore, shape_transfer, function_transfer
potato acorn	Brown, spherical acorn that	1	bframe, function_trasnfer
potato herring tail	Spherical herring tail	2	bcore, shape_transfer
pottery spoon	Spoon made of clay	1	analogy_transfer
skin stem	Thin stem	0	bcore, shape_transfer
spoon1 frame	Brown, spoon-shaped frame	1	bframe, single_differentiating_feature
spoon1_handle lens	Straight and long lens	0	bcore, shape_transfer
thorns hammer1	Small and sharp hammer	0	bcore, shape_transfer
tool boxcar	A boxcar used to make other objects	2	bcore, function_transfer
torso pencil1	Small, animate and mobile (living) pencil	0	bcore, feature_set_contrast
utensil web	A metal web, used to make food	0	feature_set_contrast, function_transfer
vegetable person3	Static, inanimate person	0	feature_set_contrast
vegetable spoon	A thing with spoon shape that grows on earth	1	bcore, function_transfer
vehicle_body vessel1	Vessel made of metal	1	bcore, slot_set_completion
vessel1 food	Concave-shaped food in which one can put something	0	bcore, function_transfer, shape_transfer
victim projectionist	Projectionist that was damaged by a gun	0	bcore, function_transfer
wheel sitting_room	Circular sitting room	0	bcore, shape_transfer

an error of more than 2 relations[10] to the target or when this error was due to fundamental relations (i.e. without them, the result would not be novel or satisfy the usefulness criteria). In Table 2, we show a sample with the training combinations, target descriptions, resulting error to the target and frames used in the query. It is important to remember that the target interpretations are obtained using only the existing knowledge representation of both nouns, which justifies the appearance of awkward interpretations (e.g. ''head hammer_handle'', ''pen person''). We can also see that the frames were initially tailored to fit the target interpretations and reused later when effective (e.g. the ''shape_transfer'' was created for ''bullet potato'', and used often in the succeeding experiments).

It is clear though that both the target interpretations and the frames were made by us, so introducing a subjectiveness component in these experiments. Since there does not seem to be any simple automatic frame generation mechanism and given that the language itself demands some expertise, the frames had to be constructed with the method described (constructing the frame that intuitively would lead to the target). On the other hand, it would be possible to use other people's interpretations of the random generated pairs, requiring a reasonably large set of participants with some expertise to understand the constraints (interpretations are confined to the specific representation). Not having done this, we tried to follow our intuition and imagination in each case. At the worst, the experiments will reflect our specific ways of noun combination on the training set applied to the other set.

The mappings used in all the experiments were automatically generated by our structure alignment algorithm, starting the alignment with the individual identifier symbol of the nouns and then finding systematic correspondences along both structures (for example, in ''necklace paper'', it establishes a mapping between ''necklace'' and ''paper'', then goes to the ''made_of'' relations, establishing a mapping between ''metal'' and ''paper'' and so on). It typically established mappings between elements with the same role in both nouns (color value with color value, made_of value with made_of value, etc.)

4.2. Results and discussion

The free generation of noun–noun combinations consisted of selecting randomly a set of 33 pairs of nouns and using the above described query (with the slots for the game script) to generate new blended concepts. Every frame shown in Table 1 was available to the system so that it could find itself the selection of frames that suited the highest scores of the fitness function. We also added an integrity constraint for having at least two frames being accomplished so to stimulate knowledge transfer. Apart

[10] This *error* is calculated by the sum of relations (from the target) that do not exist in the result with the ones that are present in the result although absent in the target.

from these, parameters were equal to those used for training.

Below, we show examples of the generation of the ''fish_tail1 desk'' and ''fish_spider'' blends, with input domains (''fish_tail1'', ''desk'', ''fish'' and ''spider'') and the frames that were applied.

In Table 3, we show the results achieved. For each pair of nouns, we show the best result (in terms of the fitness function) of the 30 runs and describe textually the result by comparison to the head and the modifier. The Usefulness value corresponds exactly to the resulting Relevance value. Therefore, a 100% means that every condition of the query was satisfied and no integrity constraints were violated. Other values indicate that either some condition was not satisfied or integrity constraints were violated (or both).

For example, in Fig. 2, we can observe that both blends satisfy all requirements of the query (therefore scoring 100%). If, say, there were no values for ''made_of'' and ''color'', then the usefulness would be 75% since two (in eight) conditions were not satisfied. Another situation could be an integrity constraint violation (e.g. ''Something cannot be *black* and made of *flesh* at the same time''), which would lead to a penalty (e.g. supposing *integrity constraint violation penalty* was 20%, ''fish spider'' usefulness value would be 80%). For the novelty, we decided for a simple yes/no decision (''no'' meaning the result is a copy of other nouns in the knowledge base). The frames listed correspond to the frames found in the construction of the best result for each combination.

Table 3
Results (average usefulness = 78%)

Combination	Interpretation	Usefulness (%)	Novelty	Frames
barrel spoon1	Spoon	100	No	bcore
bed pips	Oblong pips	53	Yes	bcore, shape_transfer
bird1 sea	A bird-shaped sea, with wings, and head and made of flesh	100	Yes	bcore, shape_transfer structure_transfer, slot_set_completion
bird_head clothes	Curve-shaped clothes	81	Yes	bcore, bframe, slot_set_completion, shape_transfer
cow_head torso	Conical torso	60	Yes	bcore, shape_transfer
desk ornament	Brown, wooden, ornament one can put paper on	100	Yes	bcore, function_transfer slot_set_completion
desk1 spoon_bowl	Spoon bowl besides which one puts a chair (and is not used to put food in)	100	Yes	bcore, function_substitution
engine apple_tree	Oblong, long and large apple tree	43	Yes	bcore, shape_transfer
fish spider	Spider with fish tail that lives in sea, but does not make webs	100	Yes	bcore, function_substitution, structure_transfer
fish_tail1 desk	Thin, triangular desk	100	Yes	bcore, shape_transfer
flower_bloom hammer	A spherical hammer	100	Yes	bcore, shape_transfer
food body_part	A body part that serves to be eaten	35	Yes	bcore, function_transfer
herring instrument	A silver, fish-shaped (with fin and tail) instrument that lives on sea and is not used to play music	100	Yes	bcore, shape_transfer, function_substitution, structure_transfer
horse_head insect	Insect	4	No	bcore
insect rodent	A small rodent	100	Yes	bcore, shape_transfer
mattress knife1	A long knife that is on a frame	0	Yes	bcore, shape_transfer, function_substitution
oak horse	A horse that grows on earth, it has a trunk and a crown, but keeps its horse shape	100	Yes	bcore, structure_transfer, function_transfer
paper1 chair_seat	White chair seat	57	Yes	bcore, slot_set_completion
patient fruit	Human-shaped, skin-colored fruit that is ill	100	Yes	bcore, shape_transfer, function_substitution
patient plant	Human-shaped, skin-colored plant	100	Yes	bcore, bframe, shape_transfer, slot_set_completion
person5 paper	Paper that sleeps in bed	100	Yes	bcore, function_substitution
person5 stem	Stem that sleeps in bed	100	Yes	bcore, function_substitution
person2 paper	Paper with a torso	60	Yes	bcore, structure_transfer
potters_wheel desk	A flat and circular desk	100	Yes	bcore, shape_transfer
pottery neck	A neck made by a human	100	Yes	bcore, function_transfer
rose_bloom desk1	Desk	100	No	bcore
sole bird1	A black bird	100	Yes	bcore, slot_set_completion
spider_legs carriage	Carriage	67	No	bcore, bframe
stem vehicle	A straight, green vehicle	100	Yes	bcore, shape_transfer, slot_set_completion
train building1	A building with the shape and structure of a train, and which serves to transport people	100	Yes	bcore, function_transfer, structure_transfer, shape_transfer
utensil pottery	Pottery	3,50	No	bcore
victim potters_wheel	Potters wheel	5	No	bcore
wheel machine	Black and circular machine	100	Yes	bcore, shape_transfer, slot_set_completion

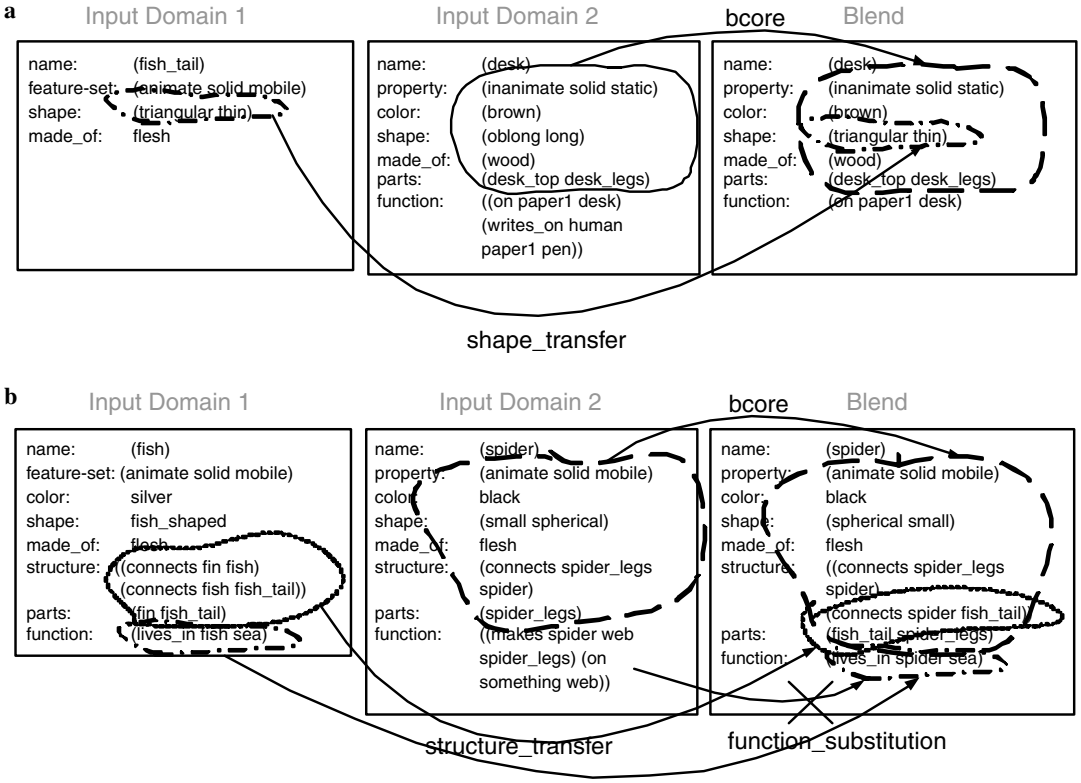

Fig. 2. Frames used in the construction of (a) "fish_tail1 desk" and (b) "fish_spider".

A first remark to notice is that every experiment ended satisfying a "bcore" frame. This is not surprising considering the query we used, which comprises a set of relations that coincides almost with the "bcore" frame relations. Still with regard to frames, we can also see that the results used essentially 6 different frames ("bcore", "slot_set_completion", "shape_transfer", "structure_transfer", "function_transfer" and "function_substitution"). A possible hypothesis is that the other 4 were either too specific (e.g. "single_modifying_feature") or too generic (e.g. "bframe") to achieve stability in the runs.

Results show that there is no correlation between the values of Novelty and Usefulness, which seems intuitively plausible. Yet, Usefulness may contrast with our intuition in some examples (e.g. there is no apparent reason why an "horse_head insect" is so less useful than a "rodent insect") and its explanation is simply that, for the context we are dealing with (a game with objects defined by a script), the new object may lack some fundamental conditions.

Another aspect we would like to invite the reader to observe is that the influence of modifiers seems predictable in some degree. Using the same modifier with two different heads tends to add the same features, with slight variations (see "person5 paper" and "person5 stem"[11]) and using different instances of the same modifier (e.g. desk in "desk ornament" and "desk1 spoon_bowl") leads to different features. This indicates that to some extent the outputs are predictable, a fact that may go against Divago as a creativity model. A balance must be found that still maintains some level of surprise and expectation but still generates useful results (for more on measuring surprise and expectation, see [9]).

Probably because the query is too much centered on the "core" of the object (every aspect except its function), it may loose its function during the blend generation, even when it is vital. For example, in blending "herring" and "instrument", the result *says* it is an instrument, but it lost its musical function, so leading to an empty concept. We also point out to the blends "train building1" and "bird1 sea". Both reveal inconsistencies ("a train building1 is a building that serves to transport people" and "a bird1 sea is a sea with wings...it is made of flesh"). On one side, these inconsistencies may reveal creative if explored from a metaphoric perspective, a very complex computational challenge. But literally, they lead to a void concept in our game environment. Preventing the existence of these extreme examples depends on adding integrity constraints (e.g. "something that serves for transport cannot be made of bricks") but will once again go against the creative potential of the system. In this case, it seems, improving Usefulness would go against Novelty.

A final remark on the results regards the observation of the Usefulness values. It should be clear that the average value of 78% obtained is highly dependent on the specific query and on the specific knowledge contained in the dataset. If the query was less constrained

[11] person5 specified a person with a relation to the concept of bed.

(e.g. having just half of the conditions), the value would certainly improve, whereas if we added conditions that could not be satisfied within the dataset, the values would never achieve 100%. These numbers show that the system is able to satisfy the query when it is (the knowledge base, query and the factory) properly configured, thus providing useful outcomes to the pragmatic context in question.

5. Comparison to C^3

We had access to a set of analogous experiments that Costello and Keane did with C^3. In these experiments, the authors randomly generated 10 pairs of nouns (e.g. "eagle" and "tulip") and, for each pair, generated interpretations for the two possible combinations (e.g. "eagle tulip" and "tulip eagle"). This gives 20 combinations, for which C^3 provided interpretations (e.g. "An eagle tulip is a tulip that grows on an eagle"). These experiments intended to model the creativity of concept combination and therefore it makes sense to compare with Divago. However, we cannot do a comparison that survives subjectivity because the values both referred here (of novelty and usefulness) and by C^3 output (plausibility, informativeness and diagnosticity) are not aligned in the same perspectives. Surely, C^3 interpretations would many times fail in the *use* measure suggested by our script, and Divago would not necessarily do well with C^3 constraints, and any of these conclusions would lead nowhere in terms of saying which one is more creative. We can do a different, perhaps more interesting, experiment: check if Divago can arrive to the same results of C^3 (thus proving possible to achieve the *same* creativity, whatever it is); and determine which frames would be needed (would they have to be different?). First of all, to level both systems in terms of representation, we had to allow Divago access to the implicit relations with other concepts.

In order to check if Divago could find the same results as C^3, we applied the process described above as *tuning phase* and found that Divago is able to achieve the same results with an average error of 2.4 and median 1. This means that the *normal* error was either 0 or 1 and so the average was strongly affected by two outliers, of errors 8 and 10. These latter cases, in which Divago failed, were interpretations that included knowledge from third nouns, i.e. when there are attributes that do not belong to any of the inputs and come from other elements in the knowledge base. In the rest, it normally achieved the same results of C^3. Another remark is that it tended to include knowledge (e.g. that "an eagle tulip is solid") that C^3 had excluded via the informativeness constraint. Whatever which one is more correct in this issue, it was also clear that, by declaring the *diagnostic* features of each noun, the features that differentiate the noun in relation to other nouns (an information that is actually available in C^3), Divago could reduce drastically this extra knowledge.

Perhaps the more striking conclusion from this experiment was that Divago could achieve the same results of C^3 (with the error just described) with a very small set of frames. Indeed, only two frames were needed in about 85% of the times: `acore` (or `bcore`, depending on whether the focus was the modifier or the head) and `analogy_transfer`. This means that, essentially, C^3 picked one of the nouns (head or modifier), built the combination centered on it – which means it has the same structure and the same "core" attributes –, and transferred also the attributes directly related to the other noun. By *directly related* we mean attributes with distance *1* in its graph representation. This seems to indicate that combinations generated in C^3 were essentially of the *property* type. The other 15% of the results used also the `bframe` (or `aframe`, depending on the focus). The results, representations and C^3 results are listed in [11].

To sum up, Divago is able to achieve the same results of C^3 by using a proper set of frames (`aframe`, `bframe`, `acore`, `bcore` and `analogy_transfer`) as goals in the search. This means that, if wanting to configure it as a noun–noun combination interpretation system, only a smaller set of frame combinations should be considered, at least for hybrid and property types, and attention to other factors should be payed, namely to diagnostic features. On the other hand, considering the other experiments in this book, we conclude that Divago offers a much larger set of possibilities, without focusing specifically on the linguistics of combinations. In other words, C^3 models noun combinations[12] and Divago deals with concept combinations, being more open to other problem solving situations. We cannot answer the doubt about the limits of C^3 (could it achieve also the same results of Divago, with a proper configuration?), but it is clear that these are internally very different systems that tackle the same cognitive problem from different perspectives.

6. Conclusions and further work

The experiments we showed in this paper present the behaviour of our system, Divago, while making blends of pairs of nouns. A straightforward conclusion is its ability to comply with the parameters of novelty and usefulness used. These parameters, while being too simplistic in terms of *human* creativity, are easily applicable for a *computational* creativity, the latter demanding a more formal, explicit and systematic methodology. We argue that Divago provides a set of mechanisms and structures (that generally follow those of Conceptual Blending [3]) that allow problems that involve creativity. One of those is the integration of pairs of nouns into a new noun, which shares a selected set of features and functionalities from both its inputs. In

[12] We point out that the combinations, as modelled in C^3, are specific to a set of human languages (English, German, Dutch...). Others, like Portuguese and French, are less ambiguous because of the obligatory use of prepositions.

this paper, experiments were made towards this problem, which was approached before [2], although integrated in a more specific model (the C^3 model of noun–noun combination interpretation).

Using the same dataset of C^3, we made a relatively large set of experiments, which we tried to show in a clear fashion.

As far as our experience and complexity analysis goes, the search space for blends grows worse than exponential with the size of input domains (see [15] for a more detailed analysis) so Divago would hardly be more efficient than C^3 in retrieving the "best" results.[13] This is understandable since Divago is not focussed on specific kinds of conceptual combinations and therefore considers many possibilities that would be deemed *wrong* for C^3. On the other hand, it was empirically demonstrated that Divago is able to achieve approximately the same results achieved by C^3 in similar experiments, in which randomly selected noun–noun pairs were given to the system. We also suggest that, with a proper configuration and the inclusion of diagnosticity knowledge, Divago would achieve the exact same results. Of course, we must remember that the mappings automatically generated by Divago only allow hybrid and property interpretations, thus a further study would be necessary for other types of interpretations.

Another aspect we could compare Divago and C^3 is their internal search mechanisms. While in C^3 the authors use a step-by-step algorithm (applying each constraint in order), Divago makes use of a parallel search method, a genetic algorithm. By its own, this difference does not say anything fundamental about the two systems, but the size of the space and the complexity brought by the presence of the frames seem to provide Divago with a much more flexible and varied behaviour: Divago looks for alternatives that are not considered by C^3.

A final remark on C^3 and Divago regards the creativity evaluation. Although Costel-lo and Keane provide arguments for C^3 as a model that considers the full creativity of conceptual combination [2], they do not make any specific validation for its outcome as being creative. Essentially, it seems that the notion of creativity in question is reduced to the diversity of possible conceptual combinations considered and so, in the sense that C^3 considers a large range of solutions in a computationally efficient way, they argue it simulates human efficient creativity in noun–noun combination interpretation. From our point of view, deeper analysis should be taken, namely with regard to the results. In the present paper, we propose an assessment of the creativity of Divago exclusively based on the two criteria of novelty and usefulness. While the choices for these may be considered too simplistic, we give an idea for the evaluation of the creative potential of such a system.

[13] In the context of C^3, a "best" result would be the one that coincides with human interpretation.

Frames provide a generic mechanism that may be used systematically to generate combinations, each one with its own underlying rationale. A challenging development will be to find frames that produce results closer to human intuitions. These would possibly be very complex frames and would necessarily demand other mapping algorithms. More specific and immediate further evolutions in this work (in noun–noun combination) may be the inclusion of a mechanism of diagnostic features and interrelation among nouns. More generally and with respect to the Divago system, a more powerful mapping algorithm and an elaboration mechanism would certainly improve considerably its creativity potential. The latter is fundamental to the Conceptual Blending framework, since it brings the possibility of *running the blend*, i.e., to give rise to novel emergent structure. Possible ideas to follow are the direct writing of domain-dependent rules and, a more challenging possibility, the mining of the knowledge base in search for patterns (e.g. when A and B exist, there is a big chance that C also happens). This emergent structure would give an independent existence to the blend, not "just" attached to the original input domains.

Acknowledgements

We thank Fintan Costello for providing us the dataset for these experiments and some of the results obtained by C^3. We also like to thank the many suggestions given by the reviewers and by Miguel Ferrand, which were fundamental to improve the quality of the paper.

References

[1] Fintan J. Costello, Noun–noun conceptual combination: the polysemy of compound phrases, PhD thesis: Trinity College, Dublin, 1997.
[2] F.J. Costello, Mark T. Keane, Efficient creativity: constraint-guided conceptual combination, Cognitive Science 24 (2) (2000) 299–349.
[3] Gilles Fauconnier, Mark Turner, Conceptual integration networks, Cognitive Science 22 (2) (1998) 133–187.
[4] G. Fauconnier, M. Turner, The Way We Think, Basic Books, 2002.
[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
[6] M. Johnson, The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason, University of Chicago Press, Chicago, 1987.
[7] Mark T. Keane, Fintan J. Costello, Setting limits on analogy: why conceptual combination is not structural alignment, in: D. Gentner, K. Holyoak, B. Kokinov (Eds.), The Analogical Mind: A Cognitive Science Perspective, MIT Press, Cambridge, MASS, 2001.
[8] D.B. Lenat, Cyc: a large-scale investment in knowledge infrastructure, Communications of the ACM 38 (11) (1995).
[9] L. Macedo, Amílcar Cardoso, Modelling forms of surprise in an artificial agent, in: Proceedings of the 23rd Annual Conference of the Cognitive Science Society, Erlbaum, Mahwah, NJ, 2001.
[10] G.A. Miller, Wordnet: a lexical database for english, Communications of the ACM 38 (11) (1995).
[11] Francisco C. Pereira, A Computational Model of Creativity, PhD Thesis, Departamento de Eng, Informática da Universidade de Coimbra, 2005.

[12] Francisco C. Pereira, Amílcar Cardoso, Knowledge integration with conceptual blending, in: Proceedings of the Irish Conference on Artificial Intelligence, (AICS 2001), 2001.
[13] Francisco C. Pereira, Amílcar Cardoso, The boat-house visual blending experience, in: Proceedings to the Second Workshop on Creative Systems, European Conference of Artificial Intelligence, ECAI'02, 2002.
[14] Francisco C. Pereira, Amílcar Cardoso, Conceptual blending and the quest for the holy creative process, in: Proceedings to the Second Workshop on Creative Systems. European Conference of Artificial Intelligence, ECAI'02, 2002.
[15] Francisco C. Pereira, Amílcar Cardoso, The horse–bird creature generation experiment, AISB Journal 1 (3) (2003).
[16] Francisco C. Pereira, Amílcar Cardoso, Optimality principles for conceptual blending: a first computational approach, in: AISB'03 Symposium on AI and Creativity in Arts and Science, SSAISB, 2003.
[17] Paulo Ribeiro, Francisco C. Pereira, Bruno Marques, Bruno Leit ao, and Amílcar Cardoso, A model for creativity in creature generation, in: Proceedings of the 4th Conference on Games Development (GAME ON'03), EuroSIS/University of Wolverhampton, 2003.
[18] Tony Veale, Mark Keane, A connectionist model of semantic memory for metaphor interpretation, in: Neural Architectures and Distributed AI Workshop, Centre for Neural Engineering, U.S.C. California, 1993.
[19] Tony Veale, Mark Keane, The competence of sub-optimal structure mapping on hard analogies, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'97, 1997.
[20] E.J. Wisniewski, Conceptual combination: possibilities and esthetics, in: Creative thought: an investigation of conceptual structures and processes, 1997.