Computational Creativity Infrastructure for Online Software Composition: A Conceptual Blending Use Case

Pedro Martins, Hugo Gonçalo Oliveira, João Carlos Gonçalves, António Cruz, F. Amílcar Cardoso, Martin Žnidaršič, Nada Lavrač, Simo Linkola, Hannu Toivonen, Raquel Hervás, Gonzalo Méndez, Pablo Gervás

Computational Creativity [CC] is a multidisciplinary research field, studying how to engineer software that exhibits behavior which would reasonably be deemed creative. This article shows how composition of software solutions in this field can effectively be supported through a CC infrastructure that supports user-friendly development of CC software components and workflows, their sharing, execution and reuse. The infrastructure allows CC researchers to build workflows that can be executed online and be easily reused by others through the workflow web address. Moreover, it enables the building of procedures composed of software developed by different researchers from different laboratories, leading to novel ways of software composition for computational purposes that were not expected in advance. This capability is illustrated on a workflow that implements a Concept Generator prototype based on the Conceptual Blending framework. The prototype consists of a composition of modules made available as web services, and is explored and tested through experiments involving blending of texts from different domains, blending of images, and poetry generation.

1 Introduction

Computational Creativity [CC] is a multidisciplinary research field mainly focused on the study and design of computational systems whose behavior can be deemed creative. Although originally seen as a subfield of Artificial Intelligence [AI], CC is now regarded as a multidisciplinary endeavor that draws on research from AI, Cognitive Science, Social Anthropology, Philosophy and Arts.

The current research in CC comprises not only various computational systems based on different cognitive theories related to creativity, such as bisociation and conceptual blending [1-9], but also evaluation methods to assess the quality of the aforementioned systems [10]. As for application domains, the research has been addressing different areas, including visual arts [11,12], music [13], poetry [14,15], and mathematics [16].

Collaborative frameworks aimed at the development, testing and sharing of creative systems are an ideal infrastructure to explore different ways of software composition and reuse, and to expand the range of application of a particular module.

We present a visual programming platform that was developed to allow the collaborative design, execution, adaptation and reuse of workflows for computational creativity applications. Such workflows are built by combining individual shared software components originally designed for performing specific tasks, or for providing access to specific resources. Building of procedures that are composed of software developed by different researchers from different laboratories is simplified, leading to novel ways of software composition for computational purposes that were not expected in advance. To illustrate the capabilities of the platform, we introduce *DivagoFlow*, a workflow that implements a Concept Generator prototype based on the Conceptual Blending framework. We propose and discuss several CC applications based on the integration of DivagoFlow with other modules. These applications include blending of text from different domains, blending of images, and poetry generation.

The final aim of our contribution is to provide CC researchers with an easy way to share and test their works, which will simplify the creation of collaborative projects. The workflows presented herein are already a result of combining various modules developed by researchers from different laboratories.

In Section 2 we present a brief description of a selection of easy-to-use workflow management systems that allow the user to compose complex computational pipelines in a modular visual programming manner, paying special attention to the *ConCreTeFlows* platform. In Section 3, we will make a concise introduction to the Conceptual Blending [CB] framework, which will provide context for the description, in Section 4, of DivagoFlow's architecture and its implementation using ConCreTeFlows. In Section 5 we describe experiments that illustrate the capabilities of the platform. Section 6 draws conclusions and discusses future work, which includes refinements and the design of new modules.

2 Infrastructures for Computational Creativity

Infrastructures supporting computational creativity and the generation of creative systems are scarce, although some recent research attempts have tried to fill this gap. One of the recent developments is FloWr [17], a system for implementing creative systems as scripts over processes, manipulated visually as flowcharts. Another example is the ConCreTeFlows infrastructure [18], which was developed to enable the construction, sharing and execution of CC workflows, composed of software ingredients of different partners of the

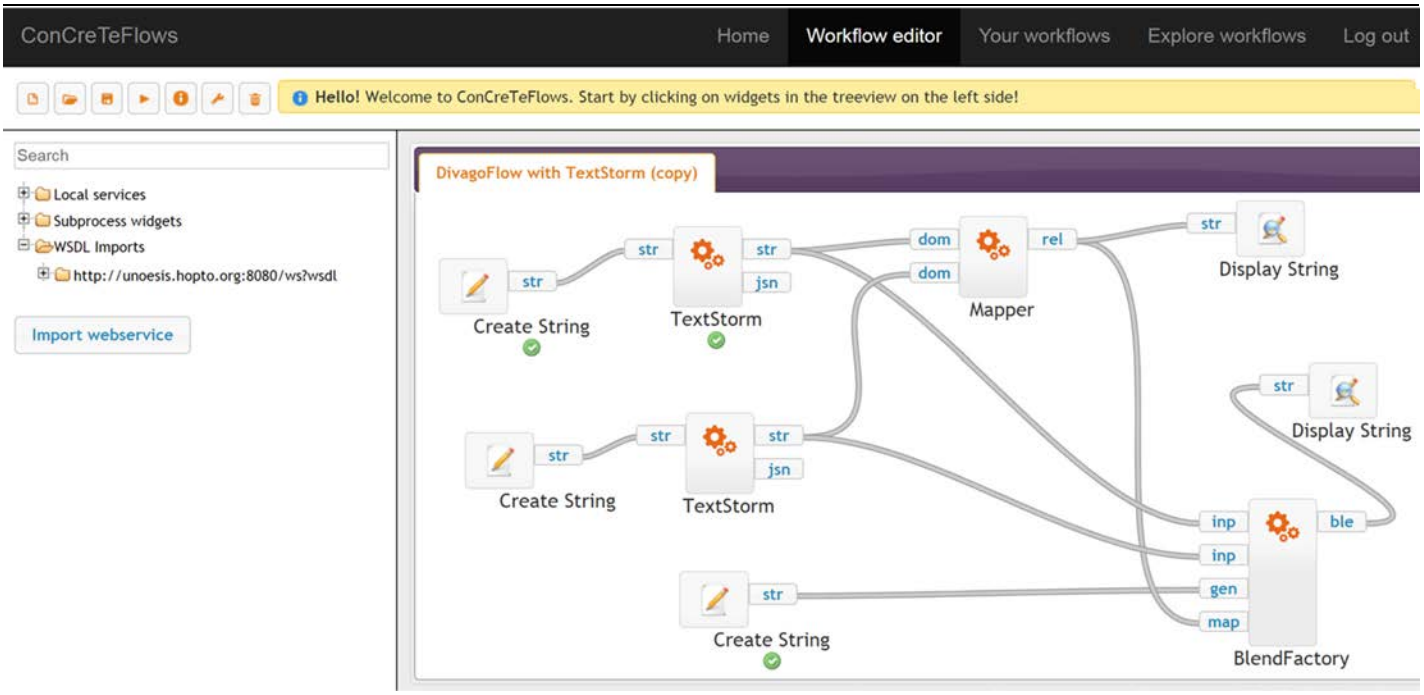

Figure 1 The user interface of ConCreTeFlows with an example of a workflow with TextStorm and DivagoFlow as central modules: a blend is produced from textual descriptions of concepts in natural language.

European project ConCreTe[1]. Both of these infrastructures use different types of resources (e.g., musical, pictorial and textual inputs) to support the development of some typical CC task such as poetry generation, metaphor creation, generation of narratives, creation of fictional ideas and conceptual blending.

ConCreTeFlows[2] is an online cloud-based platform. It was developed as a fork of the more generally data analysis oriented ClowdFlows [19], but unlike ClowdFlows, it offers various software components from the field of computational creativity and supports their specific resources and software requirements, such as widgets for accessing ConceptNet [20] and support for the Processing programming language.

ConCreTeFlows runs in any standards compliant Web browser and needs no client side installation. Software components in the workflows (denoted as *widgets*) can be either native ones, which are deployed on the ConCreTeFlows platform, or web-services, which can be added to the platform on the fly.

The user interface of ConCreTeFlows follows a visual programming paradigm and allows for creation of complex workflow processes by dragging, dropping and connecting the software building blocks. The basic software components in ConCreTeFlows are graphically represented as widgets. The connections among them represent data transfers from one component's output to another one's input. Every widget performs a task based on its inputs and user defined parameters and stores the results on its outputs. ConCreTeFlows can be extended by adding new workflow components that can offer graphical user interaction during run-time and visualization of results by implementing views in any format that can be rendered in a web browser.

The graphical user interface of ConCreTeFlows is shown in **Figure 1**. On the top of the interface there is a toolbar where workflows can be saved, deleted, and executed. Below the toolbar, on the left, is the widget repository, which is a list of available widgets grouped by their functionality. By clicking on them, the widgets can be added to the workflow construction canvas on the right, which is the main part of the user interface. At the bottom, there is a console for displaying success and error messages. For each widget successfully executed, there is a success message. In the case of a failed execution, an error message is displayed containing the widget's name and an identification of problem (e.g., absence of input data).

The platform aims to support and facilitate workflow sharing and reusability by allowing the authors to make the workflows public (they are private by default) and offer them for reuse and adaptation to others. Each public workflow is assigned a unique URL that can be shared or published by the author and then accessed by anyone to either replicate the experiment or use the workflow as a template to design new similar workflows.

When a user that is not its author accesses a public workflow, its copy is created and added to the user's workflow repository. The copy includes the structure, all the parameter settings and all the data, which ensures replicability of its results and allows the new workflow's user to change and adapt the workflow to their needs without causing any change to the original one.

[1] https://cordis.europa.eu/project/rcn/109703_en.html

[2] http://concreteflows.ijs.si

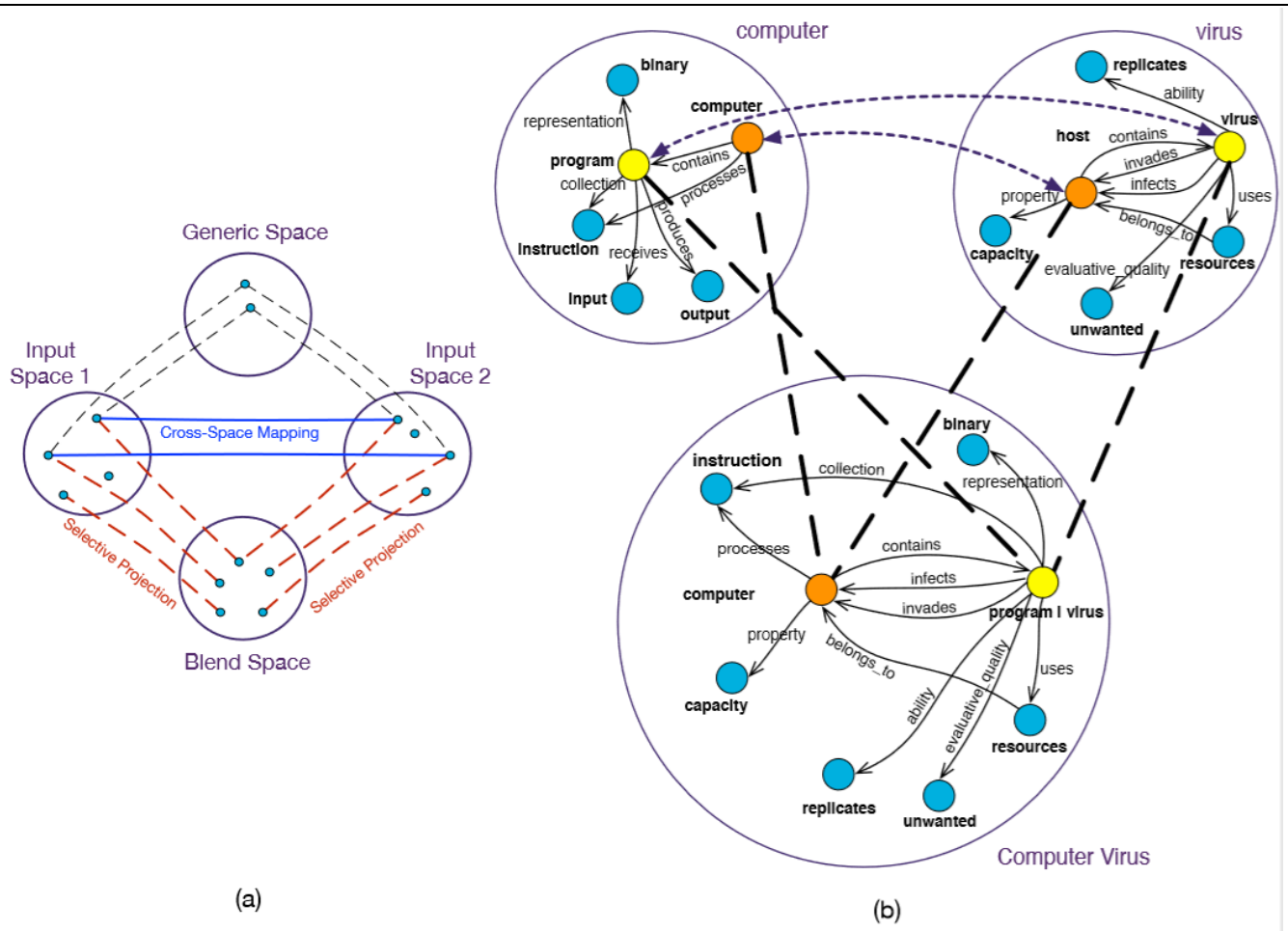

Figure 2 (a) The original four-space conceptual blending network [21] and (b) the "computer virus" blend example.

3 Brief Introduction to Conceptual Blending

The Conceptual Blending [CB] theory [21] was proposed to explain mechanisms involved in the creation of meaning and insight in the everyday mind. It provides a comprehensive description of the so-called conceptual integration process, which intends to explain how two distinct concepts like "horse" and "bird" can be *blended* into a different concept like "Pegasus". The theory also provides a set of consistent principles as well as a terminology that can be used in creativity modeling. As a result, the CB framework has been the basis for a few artificial creative systems [1-8].

A key element in CB theory is the *mental space*, a partial and temporary structure of knowledge built for the purpose of local understanding and action [22]. To describe the CB process, the theory makes use of a network of four mental spaces (**Figure 2(a)**). Two of these correspond to the *input spaces*, i.e., the content that will be blended (e.g., "horse" and "bird" are the input spaces for the "Pegasus" concept). The blending process starts by finding a partial mapping between elements of these two spaces that are perceived as similar or analogous in some respect (*cross-space mapping*). This mapping is reflected in a third mental space, called *generic space*, which contains what the initial spaces have in common, allowing it to encapsulate the conceptual structure shared by the given input spaces This third mental space provides guidance to the next step of the process, denoted as *selective projection*, where the matched elements as well as other surrounding elements are merged and *projected* into a new and final mental space, called the *blend(ed) Space*.

Figure 2(b) depicts an example of conceptual blending, where the concept "computer virus" results from the blending of two mental spaces: "computer" and "virus". An initial cross-space mapping, which is represented with dense dashed lines, maps "Computer" onto "Host" and "Program" onto "Virus". From those correspondences, selective projections are made into the blended space, which includes not only the initial matched elements but also some related (neighbor) elements. The outcome is a blended space that describes what we know as a "computer virus". Note that this is a simplified example and, for the purpose of simplicity, the generic space is omitted.

Further stages of the blending process elaborate and complete the blend. The *completion* stage corresponds to the use of existing knowledge in long-term memory to generate meaningful structures in the blend, whereas the *elaboration* stage involves cognitive work to perform a simulation of the blended space. There is not a pre-established order for these operations and several iterations may occur.

The possibilities for blending are seemingly unlimited. As such, the complexity and the quality of blends can be quite heterogeneous. The blending process is guided by *optimality principles* [21, 23], which are responsible for providing guidance towards highly integrated, coherent and easily interpreted blends. Fauconnier and Turner, who proposed the CB theory, provide a list of eight optimality principles [20]:

1. *Integration*: a blend must constitute a tightly integrated scene that can be manipulated and perceived as a unit.
2. *Topology*: For any input space and any element in that space projected into the blend, it is optimal for the relations of the element in the blend to match the relations of its counterpart.
3. *Web*: Manipulating the blend as a unit must maintain the web of appropriate connections to the input spaces easily and without additional surveillance or computation.
4. *Intensification of vital relations*: the blending process has the ability to *compress* a diffuse conceptual structure into more intelligible and manipulable human-scale situations in the blended space. That compression is likely to occur when mental spaces are connected by vital relations, such as time, space, cause-effect, analogy, or a part-whole relation. The principle of *intensification of vital relations* states that diffuse structures should be compressed by scaling a single vital relation (e.g., scale down an interval of time) or transforming vital relations into others.
5. *Maximization of vital relations*: the number of vital relations in the blended space should be maximized in order to create human scale.
6. *Pattern Completion*: Other things being equal, complete elements in the blend by using existing integrated patterns as additional inputs.
7. *Unpacking*: The blend alone must enable the understander to unpack the blend to reconstruct the inputs, the cross-space mapping, the generic space, and the network of connections between all these spaces.
8. *Relevance*; All things being equal, if an element appears in the blend, there will be pressure to find significance for this element. Significance will include relevant links to other space and relevant functions in running the blend.

In the following sections, we will describe how the ConCreTeFlows platform can be used to implement ideas from CB theory for novel concept creation.

4 The DivagoFlow Architecture

DivagoFlow is a workflow for creating novel concepts, inspired by the CB theory and implemented on top of ConCreTeFlows. It represents an evolution from a previous CB system, Divago [24]. It is relevant to the field of CC as a general-purpose concept generator designed to be used in a wide diversity of application domains and to be combined with other artificial creative systems.

In order to create novel concepts, DivagoFlow starts by selecting two concepts or domains from a given knowledge base, and then produces a blend from them.

We start from the latter task, as it forms the core of conceptual blending, which comprises selective projection and subsequent tasks aimed at creating a blend with an emergent structure on its own (the blend inherits a partial structure from the input spaces but also develops a structure that is independent of the two input spaces).

4.1 Blending Given Concepts

Humans blend concepts subconsciously, but for computers it is a non-trivial task. Given two concepts to be blended, what are the possible blends? How to compute them, and how to assess their feasibility computationally?

DivagoFlow aims to invent novel blends, and at its core is the ability to interpret what blending two concepts might mean. Equipped with this ability, it may then try to select pairs of concepts to blend (see next subsection).

The architecture of DivagoFlow for blending concepts is depicted in **Figure 3**. There are two central modules: the *Mapper* and the *BlendFactory*.

These modules interact with four mental spaces as in the CB theory. The spaces are represented as computational versions of concept maps [25]. A *concept map* is a semantic network that denotes the relationship between the concepts of a given domain, which correspond to the elements of a mental space in the CB framework. It corresponds to the factual part of the micro-theory of the domain. We often represent concept maps as graphs in which the relations are arcs and elements are nodes. **Figure 4** shows two concept maps for “bird” and “horse” connected through the common concept “cover”. Both domains are built with relations from a set of possibilities. The meaning of those relations is summarized in **Table 1**.

In addition to concept maps, the mental spaces in DivagoFlow may also include other types of knowledge: *instances*, *rules*, *integrity constraints*, and *frames* [26].

An instance is a particular example of a given domain. For example, a particular description of a bird is an instance of the domain *Bird*.

Rules are used to represent inherent causality. An example of a rule is “If *A* has wheels, then *A* can roll”.

Integrity constraints are rules that serve to assess the consistency of the concept by describing events or facts that cannot occur simultaneously.

Frames are a type of knowledge that includes a set of conditions and guidelines to define properties of the blend to be generated. They have the role of describing abstract prototypes of entities, actions, reasoning, situations or idiosyncrasies.

Let us now focus now on how the modules operate.

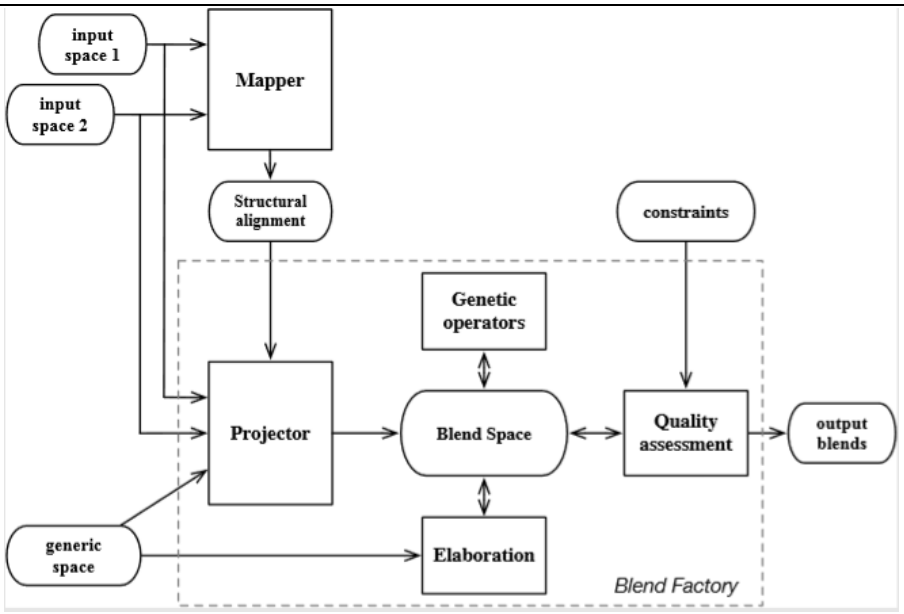

Figure 3 Concept blending architecture of DivagoFlow.

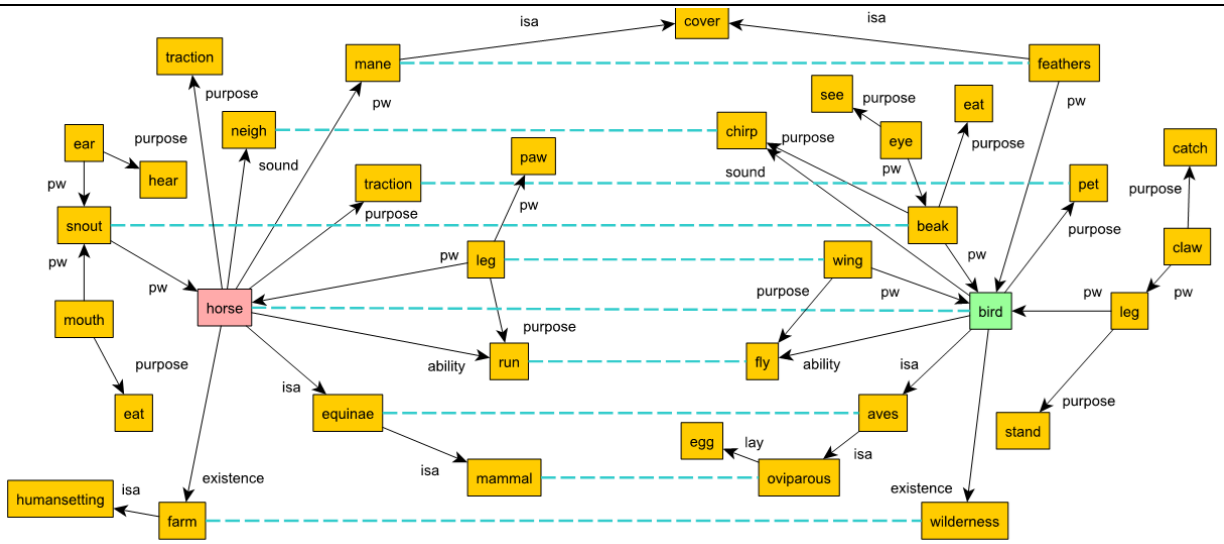

Figure 4 Conceptual maps of the two domains “horse” and “bird”. The horizontal dashed lines correspond to the pairs of concepts defining a mapping.

4.1.1 The Mapper

The *Mapper* performs the selection of elements from the input spaces for projection. Such selection is achieved by means of a partial mapping between the two concept maps using *structural alignment*. This operation looks for the largest *isomorphic* (structurally equivalent) pair of sub-graphs contained in the input spaces. Here, structural equivalence means that the graphs have the same edges (relations) regardless of the nodes. In terms of algorithmic structures, the mapping consists of a unique set of one-to-one associations of semantic concepts in the form of textual strings. The one-to-one associations are named *dormant bridges*.

The Mapper uses *a spreading activation* algorithm to look for the largest isomorphic pair of subgraphs, while the structure matching is performed through an algorithm inspired by the Sapper framework [27].

Table 1 Short description of the relations shown in Figure 4.

Relation	Meaning
ability(A,B)	*A* is able to *B*
existence(A,B)	*A* exists in *B*
isa(A,B)	*A* is a *B*
lay(A,B)	*A* lays *B*
purpose(A,B)	*A*’s purpose is *B*
pw(A,B)	*A* is part of the (whole) *B*
sound(A,B)	*A*’s sound is *B*

Starting at specific concepts, the spreading activation finds a set of concepts that were activated during the execution of the

algorithm. The concept's spreading (expansion) is done in a breadth first fashion. The concepts are activated according to a function of their relations and user specified thresholds, which together work as a filter. Hence, it is a type of graph search.

While the Sapper framework requires two cycles to obtain a mapping: one for laying down dormant bridges with the *triangulation rule* and another one for finding the mapping, the Mapper uses three cycles: one for laying down dormant bridges with the triangulation and *squaring rules*; another one for spreading activation; and a final one for finding the mapping [6]. Figure 4 also illustrates a mapping between two domains – *Horse* and *Bird* – via structural alignment.

The triangle rule creates a dormant bridge between two concepts when both share a common concept with the same relation:

$relation(A,C) \wedge relation(B,C) \rightarrow A \approx B$.

Likewise, the square rule handles the situation where the two concepts do not share a common concept as described above but do share an existing dormant bridge. In that case, the dormant bridge performs a similar function as the common concept and allows a new dormant bridge to be created if both concepts share the same relation with nearby concepts:

$A \approx B \wedge relation(A,C) \wedge relation(B,D) \rightarrow C \approx D$.

4.1.2 The BlendFactory

The second module, the *BlendFactory*, takes the output from the Mapper, the input spaces and a given generic space, and produces blends (see dashed block in Figure 3).

It starts by taking the mappings provided by the Mapper and performing a projection into the blend space. All the possible projections resulting from the mapping must be represented in the blend space at this stage, as we want to be as exhaustive as possible.

The mapping "horse" ↔ "bird" (see Figure 4) has four alternative projections: each node is projected as a separate concept (nodes "horse" and "bird' in the blend), no node is projected ("nil" node in the blend space) or both nodes are combined into a node for a new concept "horse | bird". Each of these four combinations may be a part of a possible blend.

Non-mapped nodes are projected as a copy of themselves and as a nil node (meaning that the node may appear or not in each of the possible blends). After the projection of nodes is concluded, the relations of the input spaces are also projected into the blend.

The whole set of selective projections summarize the set of all possible blends, which is called the *blendoid*. If the input spaces also contain other knowledge components, like rules, frames, instances and integrity constraints, they are also projected into the blend space.

The *blendoid* thus obtained constitutes the initial population of a genetic algorithm [GA] that explores the space of all possible blends resulting from the projection step. The GA interacts with two auxiliary components, the *Elaboration* and the *Constraints*. In each iteration, the GA sends each blend to the Elaboration component, which is responsible for applying context-dependent knowledge from the Generic Space and thus enriching the blend. Then it sends the result to the Constraints component, which implements the optimality principles of the CB theory as a set of constraints that applies to the blend to assess it. In other words, the evaluation of an individual is made by the direct application of the optimality principles. This component provides, therefore, the fitness function for the evolutionary process. The optimality constraints can be seen as *competing pressures* over the evolutionary process.

When the GA finds a solution with a satisfactory fitness value or a pre-defined number of iterations are reached, the BlendFactory stops the execution of the GA and returns the best blend. **Figure 5** illustrates a blend of Bird with Horse.

A key feature of the DivagoFlow concept generator, inherited from the original Divago framework [3], is the explicit use of the Optimality Principles in the blending process. Several other computational models of CB do not explicitly implement this component.

In particular, six principles have been modeled: Integration, Topology, Unpacking, Maximization/Intensification of Vital Relations, Web and Relevance. For each of them, a measure is provided (see [8] for details). The fitness of each blend is measured as a weighted sum of the individual measured values.

To assess the novelty of each blend, the BlendFactory makes use of the *edit distance* between the input spaces and the blend, i.e., the number of insert and delete operations required to transform one space into the other. The larger the distance to both input spaces, the higher is the novelty of the blend.

4.2 Choosing What to Blend

We have described above how blends are produced by DivagoFlows, given two concepts to blend. The overall aim of DivagoFlows is, however, to create novel blends, and selecting what to blend is a major component of the creative process. While concept blending is a relatively focused task (find an optimal blend for the given concepts), deciding what to blend is a much more open task (find a pair of concepts/domains/input spaces that give an interesting blend). Obviously, these two subtasks are tightly intertwined and selection should be informed of the blending process.

The overall DivagoFlow implementation, discussed in more detail in next subsection and represented in **Figure 6,** includes the *Domain Spotter* [DS], a module that tries to spot promising input spaces to blend within a large-scale concept map. More specifically, DS takes a semantic network and extracts two sub-graphs to be used as input by both the Mapper and the BlendFactory. To perform the task, the module uses exclusively structural information from the semantic network and therefore does not resort to any additional resources (e.g., to perform semantic analysis).

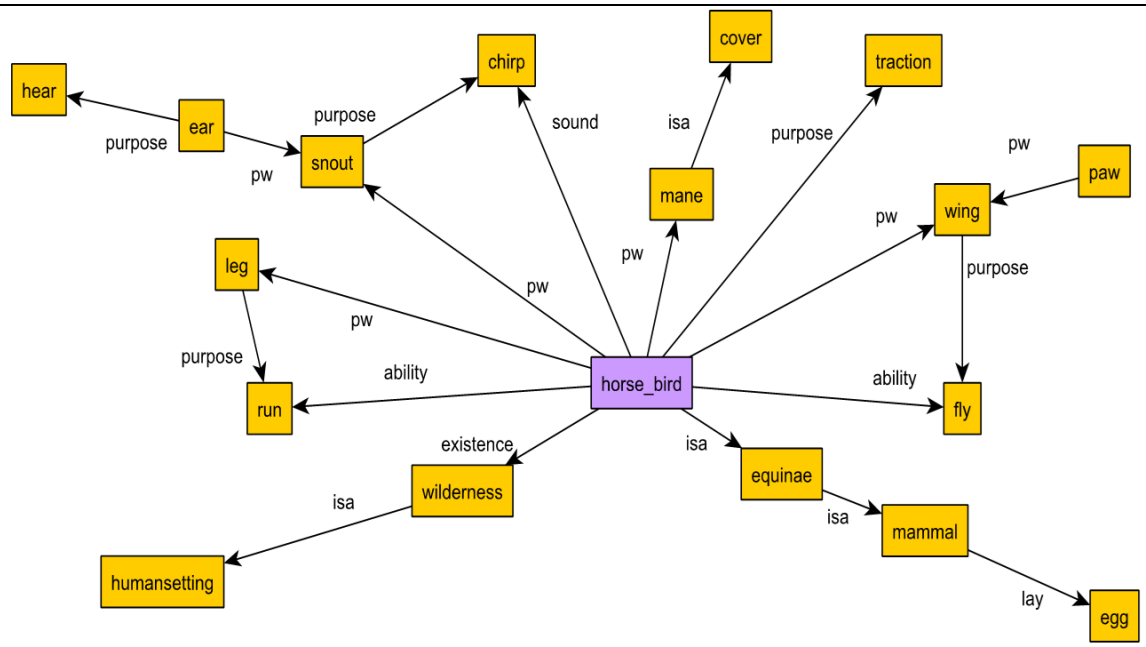

Figure 5 Blend of “bird” with “horse”.

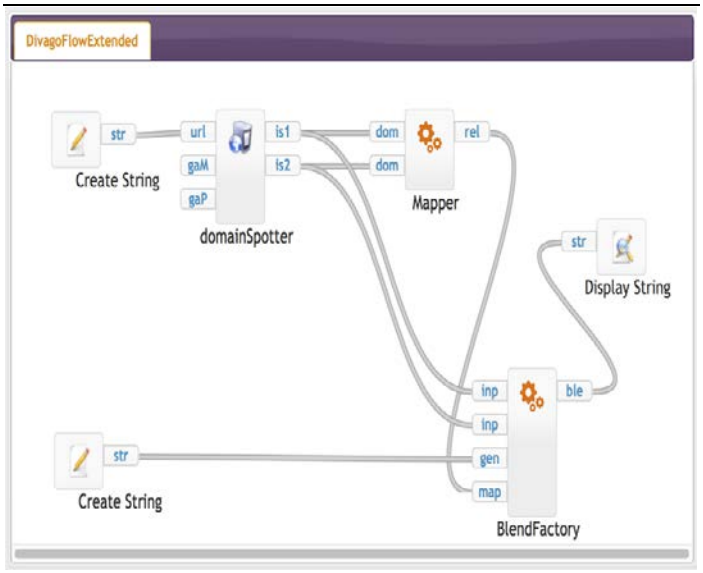

Figure 6 The extended DivagoFlow as an executable workflow in ConCreTeFlows.

The algorithm behind the DS is based on the formalized work by Nagel et al. [28]. The principle supporting both the DS and the work of Nagel et al. is the juxtaposition of two apparently unrelated domains (sets of concepts and relations) through a single term, the *bridging concept*. Intuitively, the ideal concept to select as a bridging concept is one that is present in at most two relations. As an example, we can think of the concept “life” connected (e.g., through relations “isa”) to a concept of the domain “animal” and to different concept of the domain “plant”. In this case, the bridging concept “life” is a vertex of the semantic graph, with a degree of two, connecting the “animal” and “plant” input spaces, and thus being present in the intersection of both spaces.

This notion of pairing two disjoint frames of reference using a single connection (relation or concept) was put forth by Koestler and named *bisociation* [29]. Our implementation allows the existence of more than one bridging node in the same pair of graphs, although the algorithm gives preference to a single bridging node solution. Additionally, the DS allows the manipulation of a highly interconnected semantic graph and an improved capability for handling real world blending examples.

Hence, this module executes a partitioning of the semantic graph in two sub-graphs, corresponding these to the two input spaces required. On the other hand, both input spaces should be maximized in cardinality (number of concepts or vertices) to maximize the information contained in the extracted graphs. The input spaces should also have a similar cardinality to correct for bias in either input spaces and, thus, give both an equal opportunity for influence in the succeeding modules of the concept generator.

Assuming that the semantic graph may represent a vast amount of knowledge, possibly from multiple domains and thus may be composed of a large amount of concepts and relations, the DS resorts to a fast GA capable of handling large output blends in real-time [30]. As an example, with a custom version of ConceptNet V5 (1791604 edges and 1229508 concepts) [20] and a population of 256 chromosomes we have an average execution time per epoch of 0.973±1.421 seconds (Intel X3470, 32 GB of RAM, Windows 7 x64 SP1).

The GA uses heuristics such as the degree of the vertex chosen as bridging concept, the cardinality of both the sub-graphs corresponding to the input spaces, including the ratio

between the two cardinalities. These heuristics are combined in the fitness function, which guides the optimization process behind the partitioning of the semantic graph.

One of the main differences between our work and Nagel's is that the latter requires two completely disjoint sets, while ours allows a user configured tolerance of the intersection between the two extracted domains. It is, thus, better prepared for handling real semantic data with partially overlapping regions.

This tolerance is calculated as a ratio between the cardinality of both the extracted input spaces (sub-graphs) and the number of vertices (concepts) in their intersection. However, this intersection should ideally only be composed of one vertex, the bridging concept. Therefore, the algorithm aims to find a minimum possible number of vertices in the intersection.

4.3 Integration in ConCreTeFlows

We have implemented DivagoFlow in the ConCreTeFlows platform to offer its main functionalities as web services and allow its integration with other modules being developed within ConCreTe. Each of the DivagoFlow modules (Mapper, BlendFactory and Domain Spotter) was implemented as a widget ("drop-in" graphical elements available in the ConCreTeFlows user interface).

4.4 Auxiliary tools

We developed two auxiliary tools to support the development and use of DivagoFlow: *Concept Blend Visualizer* and *TextStorm*, a concept map generator.

These tools will be described next. In addition, we have developed a tool for generation of visual metaphors, also applicable in conjunction with DivagoFlow (see Section 5).

4.4.1 Concept Blend Visualizer

The first tool, Concept Blend Visualizer [CBV], provides an interactive visualization where the different stages of the blending mechanism are represented in a four-space CB network. Each of the four mental spaces – the two input spaces, the blend and the generic space – is depicted as a large circle containing a node-link diagram, which represents all the concepts and relationships belonging to that space.

The visualization not only adapts to any user-provided datasets, but also sorts nodes inside each concept space through a force-based layout that attracts related nodes while repelling others, which creates self-organizing networks and prevents overlaps. Additionally, the initial position of nodes is calculated through a layout that minimizes edge intersections, thus reducing visual noise. **Figure 7** presents a visualization by the CBV of a Horse and Bird conceptual blend, where it depicts all four mental spaces.

The visualization is interactive and can be panned and zoomed with the mouse, while individual nodes can be selected or dragged. Selecting a node will highlight it and shows the names of the edges connected to that node, while the remaining edges will be faded out. If the user hovers the perimeter of the large circle that represents a space, it will highlight only the edges between that mental space and the others.

Edges within each space are directed and their direction is represented by the line's thickness that resembles an arrow, where the thinnest end points to the target node. If the same concepts have multiple types of relationships, this is represented with branching edges. Furthermore, there are non-directed colored edges that represent relationships between spaces when a node is selected. Nodes that exist both in the input space and in the blend will be connected through a blue line, similar nodes between the input spaces will be connected with a pink line, and relationships between the generic space are represented with a red line.

4.4.2 TextStorm

The second tool, TextStorm [31], is aimed at extracting concept maps from natural language texts. Given an input text, TextStorm applies Part-of-Speech tagging and looks for entries in WordNet [32]. Then, it builds predicates that map relations between two concepts from parsing sentences. The goal is to extract from utterances such as "Cows, as well as rabbits, eat only vegetables, while humans eat also meat", the predicates {*eat(cow,vegetables), eat(rabbit, vegetables), eat(human, vegetables), eat(human, meat)*}, which will form its concept map.

Since, in real world, concepts in text are not named every time the same way, TextStorm uses WordNet's synonymy semantic relationship [32] to identify the concepts that were already referred before with a different name, taking advantage of the fact that WordNet's organization is based on synonymy: words are grouped in sets of synonyms (*synsets*).

5 Experiments with DivagoFlow

We report and discuss a series of experiments on ConCreTeFlows in which DivagoFlow is integrated with other modules to build different CC applications including blending of texts from different domains, blending images, and poetry generation from texts.

Our main goal is to illustrate the capability of the infrastructure in allowing novel ways of software composition. However, it should be noted that this series of experiments does not include an evaluation of the different compositions in terms of the quality of the artifacts produced. For this type of evaluation, we refer the reader to previous works where comprehensive evaluations were performed [3,8,33,34].

5.1 Experiment with DivagoFlow + TextStorm

The combination of DivagoFlow and TextStorm modules is especially useful when text is used as a resource.

Figure 1 depicts a screenshot of a workflow implementation in ConCreTeFlows where TextStorm and DivagoFlow are combined to produce blends from a textual description of concepts in natural language. **Figure 8** contains a visualization of the blend produced by the workflow depicted in Figure 1.

5.2 Experiment with DivagoFlow + Vismantic

Vismantic [33] is a tool to generate visual metaphors, which can be used to visualize possible blends. Vismantic takes as input two concepts and outputs an image where properties relating to both concepts are present. The resulting visual blend is not a

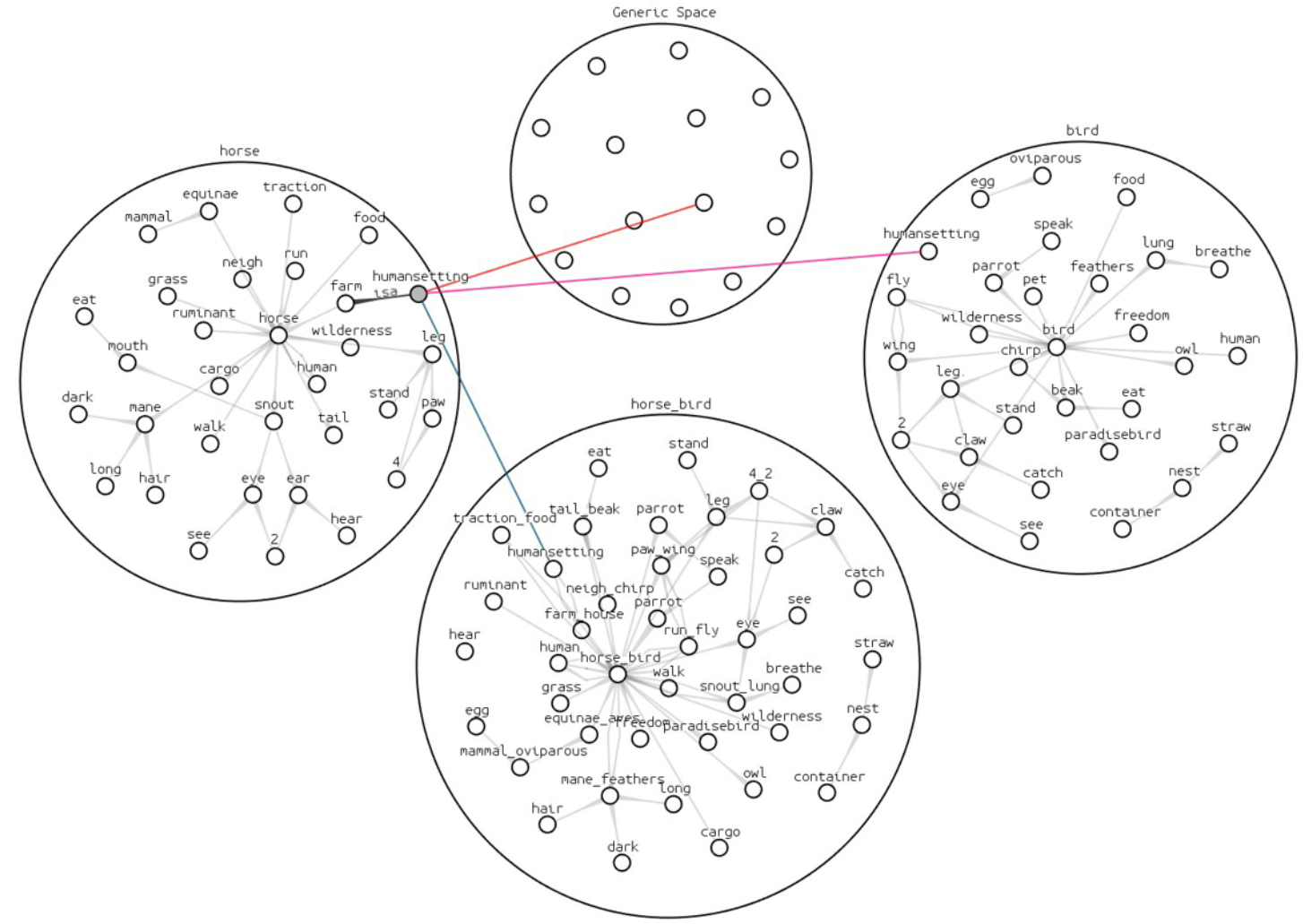

Figure 7 Visualization of the mental spaces of a “horse-bird” blend by the Concept Blend Visualizer. The input spaces for “horse” and “bird” are shown on the left and right respectively, the blend space is shown on the bottom and the generic space is represented on the top. The node “humansetting” is selected, highlighting the node’s relationship with the node “farm” as well as showing its relationships with other mental spaces through colored edges. Best viewed in color.

representation of a blend created by a concept blender, but an independent visual blend of the two concepts. As such, it can be used as a visual complement to the output of DivagoFlow.

In order to generate images, Vismantic first finds photos tagged with the two concepts in Flickr[3], respectively. Then, it analyzes each retrieved image keeping only the relevant images with high enough quality.

For each kept image, Vismantic separates the subject (the most salient object) and the background, and inpaints the subject mask in the background image aiming to hide any marks of the subject.

To create visual metaphors, Vismantic implements three visual operations: juxtaposition, replacement and fusion (see [33] for details). In this paper, we consider the latter two operations. In replacement, the subject of an image is replaced with the subject of another image, e.g., a bird replaced with a horse could show horse on a tree branch as is shown in **Figure 9 (a)**. In fusion, the texture of a subject is used to paint the other subject, e.g. a bird fused with a horse could show bird's silhouette where feathers resemble horse's fur as is shown in **Figure 9 (b)**.

5.3 Experiment with DivagoFlow + PoeTryMe

PoeTryMe [14] is a poetry generation platform that relies on a modular architecture, which enables the independent development of each module and provides a high level of customization, depending on the needs of the system and ideas of the user or developer.

[3] https://www.flickr.com/

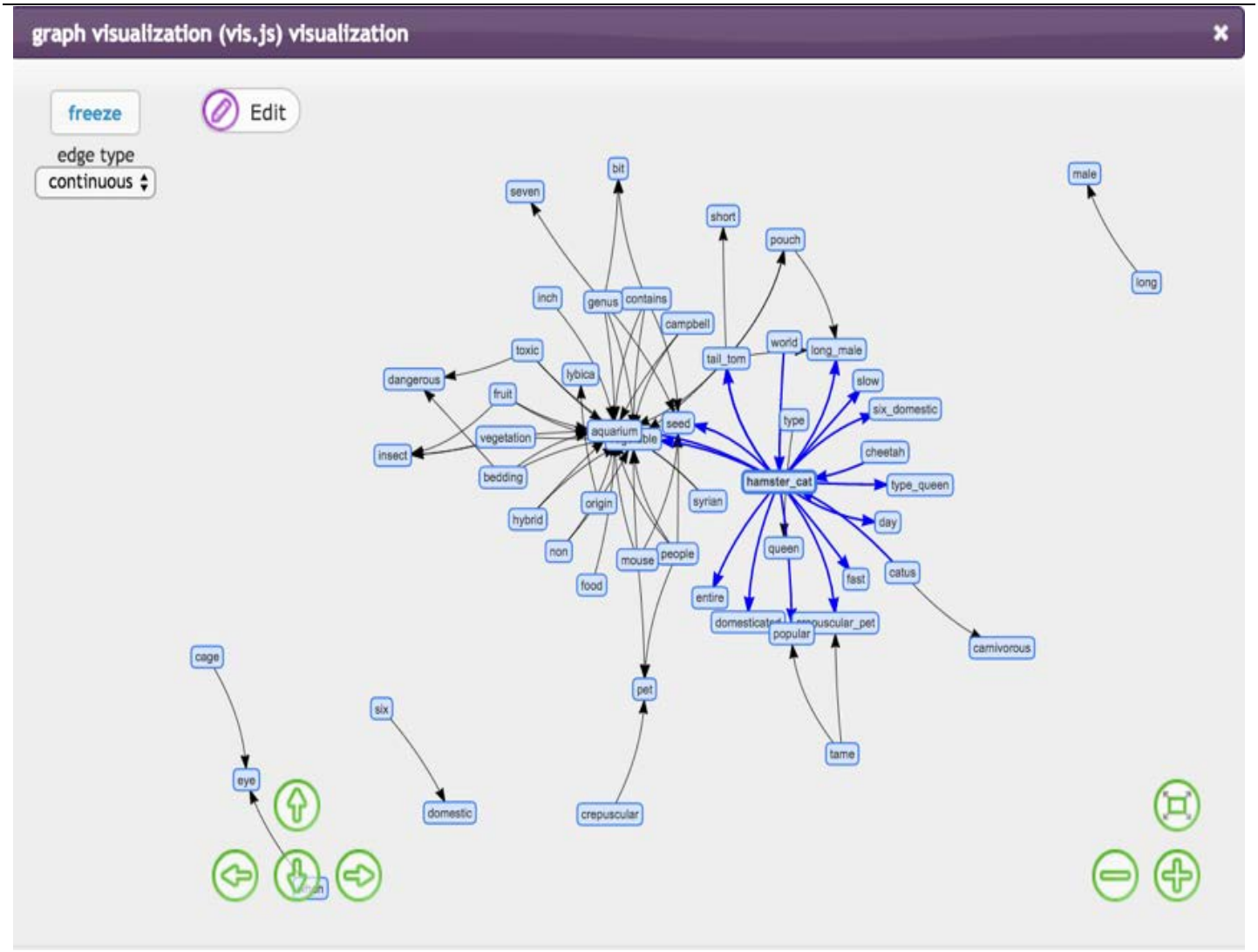

Figure 8 Visualization of the hamster-cat blend produced by the workflow depicted in Figure 1.

It was originally developed for Portuguese, but, among other instantiations, it was later adapted to Spanish and English [34]. Poetry is generated with the help of a given semantic network and a grammar with rules for generating lines based on the relations of the network. A generation strategy exploits the previous in order to generate new natural language fragments where a known semantic relation holds, and it organizes generated lines, such that they suit, as much as possible, the structure of a poetic form and exhibit certain features, such as rhymes.

Similarly to previous work [35], where it was used for generating poetry from TextStorm concept maps, here PoeTryMe was run using the graph of the Horse-Bird blend as its semantic network.

Yet, given that the current generation grammars did not have rules for most of the relations in this graph, a new grammar had to be created.

This was done automatically, as follows:

Figure 9 Examples of Vismantic's visual operations using horse and bird as input concepts: (a) replacement (bird replaced with horse); (b) fusion (horse fused with bird).

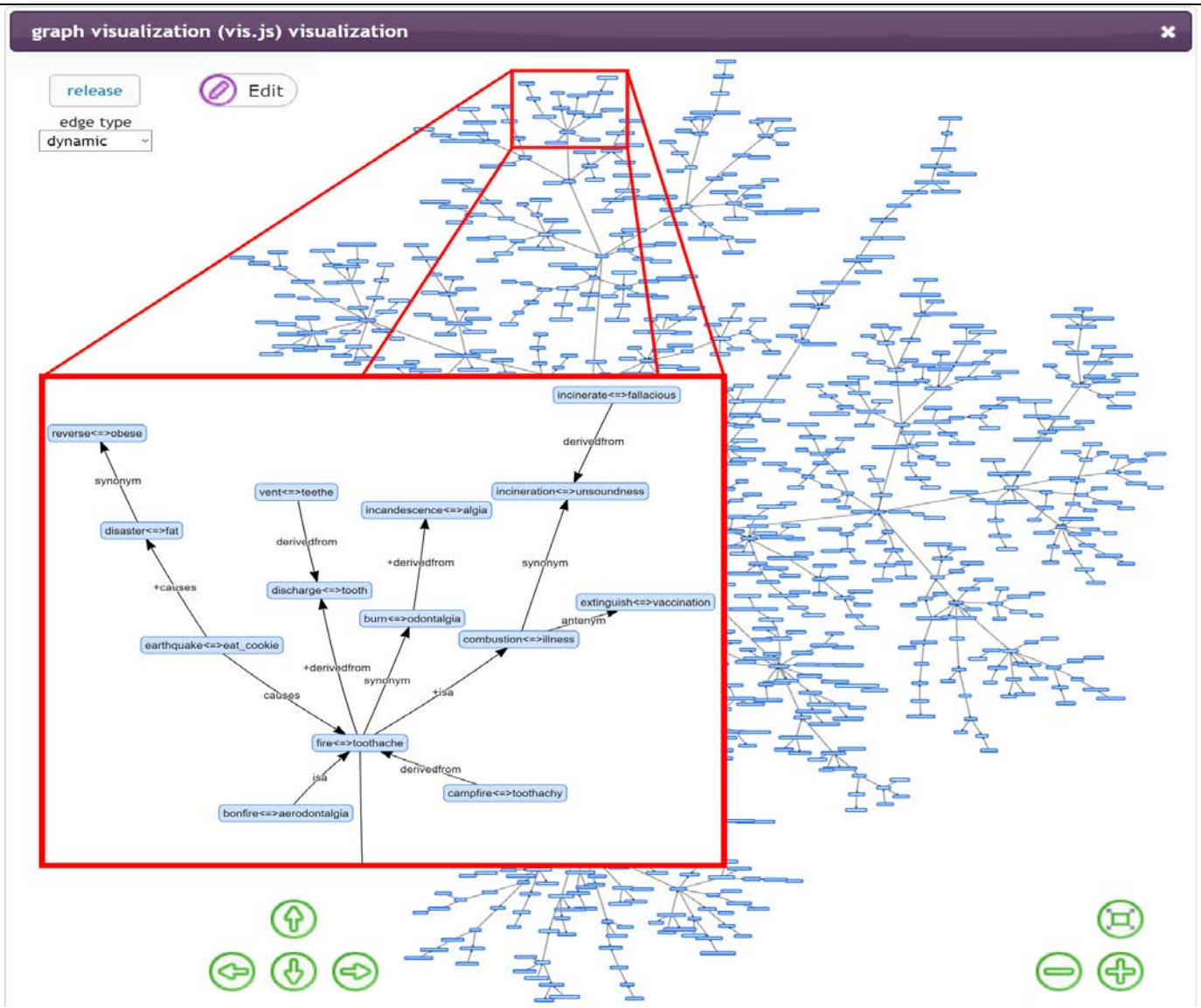

Figure 10: One of the possible mappings - drawn as a semantic network - generated by the EEmapper module. This mapping contains 1,195 pairs of concepts. A partial zoom of this mapping is shown inside the highlighted red rectangle. In that magnification we can evaluate the same structure of edges (isomorphism) which maps concepts from two subsets of the input graphs.

1. Every line in 55000+ Song Lyrics dataset[4] was scanned for the presence of two words that were related in a small collection of concept maps previously used by DivagoFlow. For instance:
   1.1. (wing, pw, bird) → *as a bird with a broken wing*
   1.2. (eye, purpose, see) → *the eye can see*
   1.3. (green, isa, color) → *but green is the color of spring*
2. For each of the previous 397 lines found, the name of the related words is replaced by a placeholder and the resulting pattern is added to a grammar as a possible way to render the relation type in text. For the previous examples:
   2.1. pw → as a <arg2> with a broken <arg1>
   2.2. purpose → the <arg1> can <arg2>
   2.3. isa → but <arg1> is the <arg2> of spring

We present poems for the horse and bird graphs, as well as for the horse-bird blend. All of them are blocks-of-four lines with the same length, some with eight syllables (first three), others with ten (last three). They were all generated with a generate-and-test strategy. More precisely, for each line to fill, up to n=2,000 textual fragments are produced sequentially and tested against the target size and rhyme, while keeping the best one.

Blocks of four lines generated for the Horse graph:

[4] https://www.kaggle.com/mousehead/songlyrics

horse is the equinae of my hell
horse is the equinae of farewell
walk away, my sweet horse, over the land
and, sure, the reverent leg must stand

Blocks of four lines generated for the Bird graph:

every bird that sings is born to fly
mommy and daddy don't see eye to eye
but this parrot was meant to speak away
where a bird with a broken leg lay

as a bird with a broken wing
but nest is the container of spring
she run her beak, but can't chirp right
bird is the aves of the night

Blocks of four lines generated for the Horse-Bird blend:

like a horse-bird with clipped paw-wing
but owl is the horse-bird of spring
don't traction food horse-bird to eye
mine paw-wing at times can run-fly

every leg has a paw-wing
but nest is the container of spring
she run her ear, but can't hear right
day and mane-feathers, dark and white

a little music from the nest next straw
a little music from the leg next claw
when we neigh-chirp not thro' the tail-beak
every parrot that sings deserves to speak

parrot must be the horse-bird of the blues
nest must be the container angels choose
you're like a leg with the broken claw
walked to the next nest directly next straw

The presented blocks sketch how the integration of a poetry generation system with DivagoFlow may result in text constrained by the input spaces. The produced text may be used as an alternative way of invoking the original concepts or the blends, creatively, due to the presence of features that are typical of poems, including the organization in four lines with the same number of syllables, always ending in rhyme. This happens while words related to the input space are used in semantically-coherent lines.

The evaluation of these examples is out of the scope of the paper. Though, an evaluation of poems produced by PoeTryMe, in different languages, is presented elsewhere [34], focusing on poetic features (metrics and rhyme); variation of text in different poems produced with the same seed words; and topicality, which is related to the semantic connection between the seed words and the words used.

5.4 Dealing with more realistic graphs

We have recently added a new component to ConCreTeFlows, the mapping module EEmapper [36]. It is an evolution from the work presented in [37] and executes a similar task to the Mapper module – the extraction of a concept map (a mapping) from semantic networks.

The EEmapper was developed in order to have an algorithm fast enough to extract mappings, in real-time, from large non-trivial conceptual graphs on the Internet, such as ConceptNet [20]. The speed requirement is not only due to the requirement of handling large semantic structures, but to support future developments and increased complexity in the fitness function of the optimization algorithm. We previously implemented an optimum mapper capable of finding the largest mappings [38] but, by being optimal, that algorithm is not able to work in real-time with conceptual graphs containing more than a hundred concepts. This happens because finding isomorphic sub-graphs is a complex task with a computational time complexity between polynomial and exponential [39].

As with Mapper, the EEmapper extracts mappings of concepts by finding sub-isomorphisms in the input conceptual graphs. Each isomorphism is built according to the structure of edges and their labels between two sub-graphs contained within the larger conceptual graph. The EEmapper is built on a GA, which stochastically evolves a large number of individuals through many generations, each individual representing a mapping of concepts. As any GA, the search for an optimal mapping is done according to the search space defined by the fitness function. Since the EEmapper is still a proof of concept, the fitness of any mapping is simply the number of concept pairs contained in the mapping. Hence, the EEmapper aims to find the mapping with the greatest amount of concept associations.

The mapper evolves isomorphisms heading towards the optimum guided by the fitness function. Each chromosome is built by first randomly choosing a pair of distinct concepts (the root pair – similar to Sapper's dormant bridge), and then randomly expanding the mapping in an isomorphic way. The isomorphism is structured according to the same sequence of nearby relations connected to either the left or the right concepts contained in the pair in a breadth first expansion. Each chromosome is thus composed by two sub-graphs connected either to the left or right root pair concept. Both sub-graphs have exactly the same edge structure (the direction and label of every edge). A mapping is extracted from the set of concepts contained in those sub-graphs. During evolution, a mutation operator is continuously applied to both the root pair of every chromosome and the structural pattern of relations defining the isomorphism in order to roam the search space. In each epoch, the mapper applies a tournament selection to exert selection pressure on the population towards the optimum solution – the largest mapping – which is returned when the evolution reaches a time limit.

The EEmapper in its ConCreTeFlows version is invoked by

the workflow as a remote Web Service[5]. To demonstrate the mapper web service, we have a publicly available workflow titled EEmapper[6]. The workflow is comprised of various format converters, the EEmapper itself and a conceptual graph drawing module. In this workflow, as input spaces, we have two semantic graphs given as triples defining the concepts and edges. These triplets are stated as Comma Separated Values [CSV] in the form *source,relation,target* with the source and target concepts corresponding to the vertices in the semantic graph and their relation name to the label of the directed edge connecting the two vertices.

To be compatible with other modules present in ConCreTe-Flows, the EEmapper component works with inputs and outputs in Divago format (e.g., facts and rules in Prolog) and hence this workflow uses two converters to adapt each input in the CSV format to the DT format required by the EEmapper module. The two remaining EEmapper inputs are *pop* and *tim*. The former specifies the size of the population being evolved by the GA internal to the mapping module and the latter specifies the maximum time (in seconds) that the GA can execute. As in any GA and in general, the larger these two values are, the higher the probability of obtaining better results.

After its execution, the EEmapper module generates two outputs. The first is the best mapping found by the algorithm and the second output is the same analogy but projected back in the original conceptual graph. As an example, a mapping containing 1,195 concept pairs is shown in **Figure 10**.

6 Conclusion

In this paper we presented ConCreTeFlows - a visual programming platform for development, execution and sharing of workflows for computational creativity applications. The platform was showcased through a conceptual blending solution named DivagoFlow, which is based on the Conceptual Blending framework.

The core concept blending part of DivagoFlow is composed of two modules, the Mapper and the BlendFactory.

The Mapper performs the selection of elements from the input spaces for projection by searching for partial mappings between the two concept maps using structural alignment.

The BlendFactory takes such mappings, the input spaces and a given generic space, and produces blends. This module is based on a genetic algorithm that explores the space of all possible blends resulting from the projection step. An implementation of the optimality principles of the CB Theory provides the fitness function for the evolutionary process.

The module that selects what concepts to blend is Domain Spotter. It introduces an autonomous and pro-active search for seemingly unrelated input spaces from a wide concept space. This module takes a semantic graph and extracts two sub-graphs to be used as input by both the Mapper and the BlendFactory. To perform the task, the module uses exclusively structural information from the semantic graph and therefore does not resort to any additional resources (e.g., to perform semantic analysis). We have also described experiments where DivagoFlow was combined with other systems. This included TextStorm, used for acquiring concept maps from text, Vismantic, for generating visual blends, and PoeTryMe, for producing poetry inspired by given concepts, including blends. All these experiments confirm that DivagoFlow can be used as a piece in other workflows, towards the generation of different creative artifacts.

Some interesting topics for future work can be identified. First, DivagoFlow can only deal with two input spaces. However, the original CB framework, as proposed by Fauconnier and Turner [22], allows the existence of more than two input spaces. Although this is a limitation of our method, it cannot be regarded as a significant drawback or a simplistic modelling of the CB mechanism, as the use of two input spaces is sufficient to ensure advanced and complex forms of blending [40].

Also, the Domain Spotter identifies input spaces linked with (at least) a bridging concept. This is a common strategy to identify seemingly unrelated pieces of information, as the bridging concept often corresponds to metaphors or ambiguous concepts [41]. We do not expect further improvements in the Mapper module but we do believe that the ensemble Domain Spotter + EEmapper can still be substantially improved in at least two aspects: additional changes in the genetic mutation to traverse more of the search space and support for multi-objective optimization allowing multiple angles of semantic evaluation in the evolving mappings.

Finally, the CB Visualizer has not been integrated in the platform so far, but we envisage to do so in a near future. It can either be integrated as a part of the BlendFactory or as an independent widget that communicates with the BlendFactory.

Acknowledgment

This research was partly funded by the Slovene Research Agency and supported through EC funding for the project ConCreTe (grant number 611733) that acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission.

References

1. M. Eppe, E. Maclean, E., R. Confalonieri, et al., “A computational framework for conceptual blending,” *Artificial Intelligence*, vol. *256*, 105-129, 2018. (journal)

2. T. Veale and D. O’Donoghue. “Computation and blending,” *Cognitive Linguistics, Special Issue on Conceptual Blending*, pp. 253–282, 2000. (journal)

[5] http://unoesis.hopto.org:8080/ws?wsdl

[6] http://concreteflows.ijs.si/workflows/copy-workflow/469

3. F. C. Pereira, “Creativity and AI: A Conceptual Blending approach,” PhD thesis, University of Coimbra, Jan 2005. (thesis)

4. P. Thagard and T. C. Stewart, “The AHA! experience: Creativity through emergent binding in neural networks,” *Cognitive Science*, vol. 35, no. 1, pp. 1–33, Oct 2010. (journal)

5. M. Schorlemmer, A. Smaill, K.-U. Kühnberger, et al.,“COINVENT: Towards a computational concept invention theory,” *Proceedings of the 5th Int. Conference on Computational Creativity, ICCC-14*, Ljubljana, Slovenia, 2014. (conference proceedings)

6. T. R. Besold and E. Plaza, “Generalize and Blend: Concept Blending Based on Generalization, Analogy, and Amalgams,” *Proceedings of the 6th Int. Conference on Computational Creativity*, ICCC-15, 2015. (conference proceedings)

7. T. Veale, “From Conceptual Mash-ups to Bad-ass Blends: A Robust Computational Model of Conceptual Blending”. In T. Veale and F. A. Cardoso, editors, Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems, pages 71–89. Springer, 2018. (book chapter)

8. P. Martins, F. C. Pereira, and F. A. Cardoso, “The nuts and bolts of conceptual blending: Multidomain concept creation with Divago,” In T. Veale and F. A. Cardoso, editors, *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, pp 91–118. Springer, 2018. (book chapter)

9. W. Dubitzky, T. Kötter, O. Schmidt, and M. R, Berthold, “Towards creative information exploration based on Koestler’s concept of bisociation,” In *Bisociative Knowledge Discovery*, pp. 11-32. Springer, Berlin, Heidelberg, 2012 (book chapter)

10. A. K. Jordanous, “Evaluating computational creativity: a standardised procedure for evaluating creative systems and its application,” Doctoral dissertation, University of Sussex, 2013 (thesis).

11. S. Colton, “The painting fool: Stories from building an automated painter,” *Computers and creativity*, pp. 3-38. Springer, Berlin, Heidelberg, 2012. (journal)

12. D. Norton, D. Heath, and D. Ventura, “Finding creativity in an artificial artist,” *The Journal of Creative Behavior*, vol. *47*, no. 2, pp. 106-124, 2013. (journal)

13. A. Zacharakis, M. Kaliakatsos-Papakostas, C. Tsougras, and E. Cambouropoulos, “Creating musical cadences via conceptual blending: empirical evaluation and enhancement of a formal model,” *Music Perception: An Interdisciplinary Journal*, vol. *35*, no. 2, pp. 211-234, 2017. (journal)

14. H. Gonçalo Oliveira, “PoeTryMe: a versatile platform for poetry generation,” *Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence (C3GI at ECAI 2012)*, PICS, Montpellier, France, August 2012. (conference proceedings)

15. J. Toivanen, H. Toivonen, A. Valitutti, and O. Gross, “Corpus-based generation of content and form in poetry,” *Proceedings of the Third International Conference on Computational Creativity*, ICCC-12, 2012. (conference proceedings)

16. M. Martinez, A. M. Abdel-Fattah, U. Krumnack, et al., “Theory blending: extended algorithmic aspects and examples,” *Annals of Mathematics and Artificial Intelligence*, vol. 80, no. 1, pp. 65-89, 2017. (journal)

17. J. Charnley and S. Colton and M. T. Llano and J. Corneli. “The FloWr Online Platform: Automated Programming and Computational Creativity as a Service”, *Proceedings of the Seventh International Conference on Computational Creativity*, pp. 363-370, 2016. (conference proceedings).

18. M. Žnidaršic, A. Cardoso, P. Gervás, et al., “Computational creativity infrastructure for online software composition: A conceptual blending use case,” In F. Pachet, A. Cardoso, V. Corruble, and F. Ghedini, editors, *Proceedings of the 7th International Conference on Computational Creativity*, ICCC-16, pages 371–379, Paris, 2016. Association for Computational Creativity, Sony CSL Paris. (conference proceedings)

19. J. Kranjc, V. Podpecan, and N. Lavrac, “Clowdflows: A cloud based scientific workflow platform”. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *ECML/PKDD* (2), volume 7524 of Lecture Notes in Computer Science, pages 816–819. Springer, 2012. (conference proceedings)

20. R. Speer and C. Havasi. “Representing general relational knowledge in ConceptNet 5,” *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*, pages 3679–3686, 2012. (conference proceedings)

21. G. Fauconnier and M. Turner, *The Way We Think*, New York: Basic Books, 2002. (book)

22. G. Fauconnier, *Mental Spaces: Aspects of Meaning Construction in Natural Language*, New York: Cambridge University Press, 1994. (book)

23. G. Fauconnier and M. Turner, “Conceptual integration networks, ” *Cognitive Science*, vol. 22, no. 2, pp. 133–187, 1998. (journal)

24. Pereira, F. C. *Creativity and artificial intelligence: a conceptual blending approach.* Walter de Gruyter, 2007. (book)

25. J. Novak, *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations*, Lawrence Erlbaum, Mahwah, NJ, 1 edition, 1998. 2nd edition published in 2010. (book)

26. F. C. Pereira and A. Cardoso, “Experiments with free concept generation in Divago,” *Knowledge Based Systems*, vol. 19, no. 7, pp. 459–471, 2006. (journal)

27. T. Veale, “Metaphor, Memory and Meaning: Symbolic and Connectionist Issues in Metaphor Interpretation,” PhD Thesis, Dublin City University, 1995. (thesis)

28. U. Nagel, K. Thiel, T. Kötter, et al., “Towards discovery of subgraph bisociations,” In Michael R. Berthold, editor, *Bisociative Knowledge Discovery*, volume 7250 of Lecture Notes in Computer Science, pp. 263–284. Springer Berlin Heidelberg, 2012. (book chapter)

29. A. Koestler, *The Act of Creation*, New York:Macmillan, 1964. (book)

30. J. Gonçalves, P. Martins, A. Cruz, et al., “Seeking divisions of domains on semantic networks by evolutionary bridging,” *Proceedings of the International Conference on Case-Based Reasoning* (ICCBR), Frankfurt, Germany, 2015. CEUR, CEUR. (conference proceedings)

31. A. Oliveira, F. C. Pereira, and A. Cardoso, “Automatic reading and learning from text,”.*Proceedings of the International Symposium on Artificial Intelligence*, pp. 69–72, 2001. (conference proceedings)

32. G. A. Miller, “Wordnet: A lexical database for English,” Commun. ACM, vol. 38, no. 11, pp. 39–41, November 1995. (journal)

33. P. Xiao and S. Linkola, “Vismantic: Meaning-making with images,” *Proceedings of the Sixth International Conference on Computational Creativity*, ICCC-15), pp. 158–165, Park City, Utah,

June - July 2015. Brigham Young University, Brigham Young University. (conference proceedings)

34. H. Gonçalo Oliveira, R. Hervás, A. Díaz, et al.,"Multilingual extension and evaluation of a poetry generator," *Natural Language Engineering*, vol. 23, no. 6, pp. 929–967, 2017. (journal)

35. H. Gonçalo Oliveira and A. O. Alves, "Poetry from concept maps – yet another adaptation of PoeTryMe's flexible architecture," *Proceedings of 7th International Conference on Computational Creativity*, ICCC-16, Paris, France, 2016. (conference proceedings)

36. J. Gonçalves, P. Martins, and A. Cardoso, "A fast mapper as a foundation for forthcoming conceptual blending experiments," *Special Track Analogy - Proceedings from The Twenty-Sixth International Conference on Case-Based Reasoning (ICCBR 2018)*, 2018. (conference proceedings)

37. J. Gonçalves, P. Martins, and A. Cardoso, "Blend City, Blendville," *Proceedings of the Eighth International Conference on Computational Creativity*, ICCC-17, 2017. (conference proceedings)

38 J. M. Cunha, J. Gonçalves, P. Martins, et al., "A pig, an angel and a cactus walk into a blender: A descriptive approach to visual blending,". *Proceedomgs of the Eighth International Conference on Computational Creativity*, ICCC-2017, 2017. (conference proceedings)

39. L. Babai, W. M. Kantor, E. M. Luks. "Computational complexity and the classification of finite simple groups," *Foundations of Computer Science*, 1983. (conference proceedings)

40. M. Turner, *The Origin of Ideas*, Oxford University Press, 2014. (book)

41. T. Koetter, K. Thiel, and M. Berthold "Domain bridging associations support creativity," *Proceedings of the First International Conference on Computational Creativity*, ICCC-2010, pp. 200–204, 2010. (conference proceedings)

Pedro Martins *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal (pjmm@dei.uc.pt).* Pedro Martins holds a PhD degree in Computer Science, an MSc degree in Informatics and Systems, and a BSc degree in Mathematics, with a major in Computer Science, from the University of Coimbra. His main research interests are in the fields of Computational Creativity, Computer Vision, Computer Graphics, and Computational Art. Currently, he is a researcher at the Centre for Informatics and Systems of the University of Coimbra (CISUC) and an Assistant Professor at the Department of Informatics Engineering of the University of Coimbra.

Hugo Gonçalo Oliveira *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal (hroliv@dei.uc.pt).* Received a BSc degree (2006), a MSc degree (2007) and a PhD (2013), all by the University of Coimbra. Has worked as a hired researcher in the node of Coimbra of the project Linguateca (2007-2008); as a NLP Engineer in Spotdata, Coimbra / Montpelier (2013); and is currently an Assistant Professor at the Department of Informatics Engineering of the University of Coimbra. Has had a PhD grant awarded by the Portuguese Foundation for Science and Technology, FCT (2009-2012) and won the best PhD thesis in the Computational Processing of the Portuguese Language (2012-2014). Has (co-)authored 12 articles in peer-reviewed journals and more than 60 papers in peer-reviewed proceedings of scientific conferences. His main research interests are Natural Language Processing and Computational Creativity.

João Carlos Gonçalves *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal (jcgonc@dei.uc.pt).* João Gonçalves is a Ph.D. student and a member of the Cognitive and Media Systems (CMS) group of the Centre for Informatics and Systems of the University of Coimbra (CISUC). He finished his MSc in Informatics Engineering in 2012 with a thesis implementing high performance SVMs running on the GPU. His current research work is concerned with the acquisition and development of new insights in the fields of computational creativity, cognitive sciences and artificial consciousness as well as high performance computing to make all of his creations work swiftly and scalably.

António Cruz *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal (antonioc@dei.uc.pt).* A. Cruz received a B.S. degree in 2011 and a M.S. degree in 2014 at the University of Coimbra, and he is currently a PhD student at the Faculty of Sciences and Technology of the University of Coimbra enrolled in the Doctoral Program for Information Science and Technology. His interests center around graphic design and programming applied to Information Visualization, particularly the development of dynamic data visualization methods and tools. Currently he is exploring the visualization of multivariate big data within the field of Computational Biology, including gene expression time-series, biological pathways, and protein-protein interaction networks.

F. Amílcar Cardoso *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal (amilcar@dei.uc.pt).* F. Amílcar Cardoso is a Lecturer at the Department of Informatics Engineering of the University of Coimbra, where he teaches Artificial Intelligence, Computational Creativity, Programming for Design and other topics. He is currently Vice-President of Instituto Pedro Nunes, the technology transfer and incubator of University of Coimbra. He developed pioneering work on Computational Creativity in the 90s, and assumed since then an active role in the area. In the last years, his research has been focused mostly on computational models of Conceptual Blending. He has been involved in two EU projects on Computational Creativity: the FET CA PROSECCO - Promoting the Scientific Exploration of Computational Creativity (FP7-ICT-FET-600653) and the FET/ICT ConCreTe - Concept Creation Technology (FP7-ICT-FET-611733). He was the General Chair of the International Conference on Computational Creativity, held in Paris, France, June 2016. He is co-editor of the forthcoming book "Computational Creativity - The Philosophy and Engineering of Autonomously Creative Systems", to be published by Springer in 2019. He is founder of the Association for Computational Creativity, where he currently serves as Treasurer.

Martin Žnidaršič *Jozef Stefan Institute, 1000 Ljubljana, Slovenia (martin.znidarsic@ijs.si).* Martin Znidarsic is a researcher at the Department of Knowledge Technologies of JSI, and Assistant Professor at the JSI Postgraduate School. His main research interests are in data mining and machine learning with a focus on probabilistic modelling, evaluation modelling, sentiment analysis, and computational creativity. He applies his work mostly in domains of ecology and finance. He was involved in several EU projects from FP5 to H2020 programme (ECOGEN, SIGMEA, Co-Extra, ConCreTe, WHIM, SAAM, etc.), and is the coordinator of the CF-Web project (started in June 2017).

Nada Lavrač *Jozef Stefan Institute, 1000 Ljubljana, Slovenia (nada.lavrac@ijs.si).* Nada Lavrac is Head of Department of Knowledge Technologies of Jozef Stefan Institute, Ljubljana, Slovenia. She is Professor at Jozef Stefan International Postgraduate School in Ljubljana and at University of Nova Gorica. Her main research interests are in Knowledge Technologies, an area of Information and Communication Technologies. Her particular research interests are machine learning, data mining, text mining, knowledge management and computational creativity. Areas of applied research include data mining applications in medicine, health care and bioinformatics, media analysis and virtual enterprises. She is Ambassador of Science of Slovenia and is an elected ECCAI Fellow. She was founding member

of the International Machine Learning Society board, and is member of the Artificial Intelligence in Medicine board. She was scientific and/or organizational chair of numerous international conferences (including AIME 2011, ILP 2012, ICCC 2014), she was invited speaker at conferences (including MedInfo 2010, ECCB 2014, ISWC 2017). She wrote 3 books and co-edited tens of conference proceedings. She coordinated and participated in several EU projects.

Simo Linkola *University of Helsinki, Helsinki, Finland (simo.linkola@helsinki.fi).* Simo Linkola obtained his M.Sc. degree on computer science in 2016 from University of Helsinki, and is currently pursuing his Ph.D. there. His research considers the intersection of computational creativity, autonomous agents and multi-agent systems. From a single agent perspective he is interested in how autonomous and self-adaptive agents can exhibit creativity both in their outputs and in their internal processes. In multi-agent settings his main focus is on how a group of creative agents can work together in novel ways to accomplish tasks that are not easily fulfilled by any single agent alone.

Hannu Toivonen *University of Helsinki, Helsinki, Finland (hannu.toivonen@helsinki.fi).* Hannu Toivonen is Professor of Computer Science at the University of Helsinki since 2002. He works in the areas of artificial intelligence and data science, more specifically in computational creativity and data mining. His current research focus is on using data science for computational creativity, on self-aware and creative systems, and on analysis and generation of natural language. He has over 100 international refereed publications. According to Google Scholar, he has been cited over 20,000 times and he has an h-index of 51. Six of his publications have over 1,000 citations. He served as Programme Chair of IEEE ICDM 2014, a leading data mining conference, and of ICCC 2015, the International Conference on Computational Creativity. He is Editorial Board member of the leading journals (Data Mining and Knowledge Discovery; Machine Learning) and a regular Senior Programme Committee member of the leading conferences (SIGKDD, ICDM, ECML-PKDD, SDM, IJCAI, AAAI, ICCC, ...) in his research areas.

Raquel Hervás *Facultad de Informática, Universidad Complutense de Madrid, Spain (raquelhb@fdi.ucm.es).* Dr. Hervás received her PhD in Computer Science in 2009, and she is Associate Professor in the Computer Science School at the Universidad Complutense de Madrid (UCM) since 2010. Dr. Hervás has been part of the research team in several national and international research projects, including the coordination of the IDiLyCo project, funded by the Spanish Ministry of Economy. Her research focuses on applying natural language technologies to different fields like accessibility, computational creativity and narratology, and she is author of more than 70 international publications in these topics.

Gonzalo Méndez *Facultad de Informática, Universidad Complutense de Madrid, Spain (gmendez@fdi.ucm.es).* Dr. Méndez received his PhD in Computer Science from the Universidad Politécnica de Madrid (UPM) in 2008. He is currently an Assistant Professor in the Department of Software Engineering and Artificial Intelligence of the Complutense University of Madrid. Dr. Méndez is the director of the Instituto de Tecnología del Conocimiento (www.ucm.es/itc), where he has been a researcher since 2008. He has been part of several national and international research projects, being his current research interests on narrative generation, natural language generation for figurative language, accessibility systems and agent-based simulations. He is author of more than 50 international publications on these topics.

Pablo Gervás *Facultad de Informática, Universidad Complutense de Madrid, Spain (pgervas@sip.ucm.es).* Pablo Gervás is Associate Professor at the Department of Software Engineering and Artificial Intelligence of the School of Informatics at the Universidad Complutense de Madrid. He is the director of the NIL research group (Natural Interaction based on Language) and of the Instituto de Tecnología del Conocimiento. Over the years, his research interests have shifted towards studying the role of narrative in human communication, with a view to applying it in human-computer interaction. Besides his generic interest in understanding and modelling language, he tries to tackle computational models of human creativity. His interests include the following lines of research: Natural language generation, Natural language analysis, NLP for accessibility, and NLP and literary artifacts.