Towards a Computational Theory of Epistemic Creativity

Jiří Wiedermann[1] and **Jan van Leeuwen**[2]

"The creative act is not an act of creation in the sense of the Old Testament. It does not create something out of nothing: it uncovers, selects, re-shuffles, combines, synthesizes already existing facts, idea, faculties, skills. The more familiar the parts, the more striking the new whole.
A. Koestler [9]

Abstract. We investigate the computational process of creativity from the viewpoint of our recent thesis stating that computation is a process of knowledge generation. Rather than considering the creativity process in its full generality, we restrict ourselves to so-called epistemic creativity which deals with the processes that create knowledge. Within this domain we mainly concentrate on elementary acts of creativity — viz. drawing analogies. In order to do so using the epistemic framework, we define analogies as certain relationships among linguistic expressions and we state what knowledge must be discovered in order to resolve a given incompletely specified analogy. We assume analogies are formed in a natural language and also require that a solution of each analogy must contain an explanation why the resulting analogy holds. Finally, the difference between non-creative and creative computational processes is discussed. Our approach differs from the majority of previous approaches in stressing the knowledge discovery aspects of computational creativity, in requiring explanations in analogy solving and, last but not least, in including theory-less domains serving as knowledge base for knowledge discovery process.

1 INTRODUCTION

Creativity is an activity producing knowledge in the form of ideas, artifacts or behavior that is new for its creator and in some way valuable or important for him or her. Without creativity, no artificial system can aspire to be on par with human intelligence. In its most developed form creativity permeates all human activities. It has been subject of studies in many academic disciplines, among them in psychology, cognitive science, education, philosophy (particularly philosophy of science), technology, theology, sociology, linguistics, economics, and in arts. While all of these disciplines have defined creativity according to their own paradigms and needs, hardly any of them made a serious effort to reveal the underlying mental mechanisms supporting and enabling the process of creativity. This is perhaps due to the fact that the anticipated nature of these mechanisms has been assumed to lay outside of the disciplines at hand. But there is one exception to this rule, and this is the field of artificial intelligence, and especially artificial general intelligence (AGI). Mechanisms of artificial creativity have been intensively studied in cognitive science as well. Due to its omnipresence in many fields of study, the literature concerning creativity is immensely rich and too extensive to be discussed, summarized or referenced fully here.

When inspecting definitions of creativity in whatever discipline, AGI included, two things strike the eye: first, the definitions are very informal, given in a natural language, and second, the definitions hardly ever mention the term knowledge. Especially the latter fact is quite surprising since, perhaps with the exception of artistic creativity, the ability to create new knowledge permeates all domains of creativity. In such domains the primary purpose of creativity is to generate or to demonstrate new knowledge in whatever form — be it conventional knowledge used in everyday life, or scientific knowledge, or a skill, behavior, or a "materialized knowledge" (i.e., knowledge embedded into objects, their functioning, shape or appearance). This kind of creativity is called *epistemic creativity*. Mokyr (cf. [11]) describes it as "actually creating new knowledge or combining existing fragments of knowledge in altogether new ways", as part of his more general view of productive creativity. How can the functioning of epistemic creativity effectively be understood?

It is true that the research field called "knowledge discovery" has become quite popular since the 1990s. Knowledge discovery describes the process of automatically searching large volumes of structured (databases, XML) and unstructured (text, documents, images, multimedia) data for patterns that can be considered knowledge about the data. When compared to what one expects from epistemic creativity, the field of knowledge discovery, despite its name, merely extracts knowledge about the data without having the ambition to create new knowledge other than that which can be straightforwardly extracted from data. This, by the way, can be illustrated by the fact that in the research papers in this field, the word "creativity" is used quite rarely.

It is also true that at the intersection of the fields of artificial intelligence, cognitive psychology, philosophy, and the arts there is a flourishing multidisciplinary endeavor called "computational creativity" (also known as artificial creativity or creative computation). Its roots go back to the nineteen sixties. The field is concerned with theoretical and practical issues in the study of creativity. Here the situation seems to be fairly opposite to the previous case: while the field teems with the word "creative" and all its derivatives, the notion of "knowledge" is much less frequent here. In part this could be due to the fact that the field very often seeks its inspiration in artistic creativity. The field is looking for its theoretical foundations.

In our opinion this frequent overlooking of the connection between (computational) creativity and knowledge generation - where the latter obviously is the main sense of epistemic creativity - may have been caused by an insufficient understanding of what compu-

[1] Institute of Computer Science of CAS and Czech Institute of Informatics, Robotics and Cybernetics of CTU, Prague, Czech Republic email: jiri.wiedermann@cs.cas.cz
[2] Center for Philosophy of Computer Science, Utrecht University, the Netherlands email: J.vanLeeuwen1@uu.nl

tation is. In our recent works [17],[18], [19], [15] we coined the idea that the classical view of computation, based on the ways information is processed by all sorts of machine models (typically by Turing machines), prevents us from clearly seeing the main purpose of computations. The classical view favors the view of HOW computations are performed, instead of WHAT they are doing, i.e. of what is their sense. We hold the view that computation is any process of *knowledge generation*, as we have demonstrated in our previous works. Note that the notion of knowledge generation is machine-independent: we are not interested how, by what means, knowledge is generated, be it in a serial, parallel, interactive, or any other way. What counts is what knowledge is generated.

Changing the view of what computation is may have dramatic consequences. For instance, in the past, various authors have argued that cognition is not computation (cf. [3], [14]), where they have viewed computation in its classical sense, through classical models and scenarios of computations. Under the new view, cognition becomes knowledge generation, and thus, computational, independently of the underlying machine model and computational scenarios. The previous problem vanishes thanks to a new apprehension of computation.

Seeing computations as knowledge generation processes does not automatically turn every computation into a creative process. Intuitively, epistemic creativity requires more than producing knowledge according to some rigid schema (program), counting with some fixed number of alternatives each of which corresponding to a certain pre-specified circumstance. For creativity, we require more: new, original alternatives (pieces of knowledge) satisfying as many required constraints as possible must be discovered within the existing knowledge and combined in a novel way under whatever circumstance that cannot be known beforehand. From the candidate alternatives, the one best fitting the constraints must get chosen. This leads to a computational view of epistemic creativity.

The ideas described in the last paragraph answer the often posed question why people have ideas and computers don't. The reason why computers are not creative can have two reasons. The first one is that in the majority of cases when an average person is using a computer, creativity is not required by the application (e.g., in looking for a train schedule). The second answer concerns the so-far quite rare cases where creativity is required — e.g., when consulting symptoms of a disease, or asking for a nice analogy. In such situations a computer will probably not be as creative as we would like to see because it is programmed without understanding how creativity works and what its prerequisites are. Nonetheless, the essence of epistemic creativity has been described in the last two sentences of the previous paragraph. Can we say more about the respective creative processes? Can we be more specific in describing which knowledge generating processes can be seen as creative processes? What are the prerequisites for computational creativity? (Note that we are using the term "computational creativity" in a new, broader sense than mostly used in the eponymous research field.)

In this paper we will answer the last three questions from the epistemic viewpoint of computations. As it turns out, answering the last three questions in their full generality is not easy. Therefore, in what follows we will first investigate but a specific case of creativity. We will concentrate on one of the simplest cases of creativity, and this is analogy solving. Solving an analogy can be seen as an elementary creativity act that calls for discovering and displaying new relations between known pieces of knowledge. Then we will extend our study to a general case of new knowledge discovery.

The structure of the paper is as follows. In Section 2 we present our view of computation as knowledge generation that will offer a unified framework for our further consideration of computations. Special attention is paid to computations in theory-less domains corresponding to natural languages. Section 3 contains the main contribution of the paper. After some preliminaries in Subsection 3.1. analogies and their formal definition in the epistemic framework is presented in Subsection 3.2. The "hard to vary" principle is described, enabling a "quality" judgment of explanatory analogies. In Subsection 3.3. metaphors and allegories as variants of analogies are considered. Subsection 3.4. deals with the efficiency issues in analogy solving. The entire Section 4 is devoted to the general problem of knowledge discovery. Finally, Section 5 contains a general discussion, also paying attention to the difference between creative and non-creative knowledge generation. Conclusions are given in Section 6.

The contribution of the paper to the present state of the art of the theory of computational creativity can be seen in several planes. First, the epistemological view of computations offers a natural unified framework for studying problems related to epistemic creativity. Second, this framework, being machine independent, allows the consideration of theory-less knowledge domains. Third, pertaining to analogy solving, the requirement for a computation to be accompanied by evidence that it works as expected is mirrored in the definition of analogy by a similar demand for analogy explanations. Fourth, explanations attached to each solution of explanatory analogies allows one to judge their explanatory power via the "hard to vary" principle. Finally, our considerations shed further light on the general problem when a computational process is a creative process.

2 COMPUTATION AS KNOWLEDGE GENERATION

Viewing computation as knowledge generation as described in [17], [18] and [19], requires certain ingredients that we first describe informally.

Knowledge in our framework is knowledge in the usual sense of this word. This, of course, does not look like a definition of knowledge, but we need not be very specific. For illustration purposes only, we cite the following definition from Wikipedia: *Knowledge is a familiarity with someone or something, which can include facts, information, descriptions, or skills acquired through experience or education. It can refer to the theoretical or practical understanding of a subject. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); it can be more or less formal or systematic.* Obviously, knowledge according to this definition is observer–dependent.

Any knowledge is a part of a so-called *epistemic domain,* or *domain of discourse,* corresponding to the kind of knowledge we are interested in. Such a domain can be given formally — as in mathematical or logical theories (e.g., theory of recursive functions) or entirely informally, in a natural language, as all sentences describing phenomena in a real world. Intermediate cases (like physical, chemical or biological theories) described in part formally and in part informally are also acceptable. In any case, we must have means to describe the so-called *pieces of knowledge* (e.g., axioms, sentences or formulae in formal theories, or words and linguistic expressions in informal theories described in a natural language).

The final ingredient we require are so-called *inference rules* applicable to the pieces of knowledge in a given domain allowing constructing, generating new pieces of knowledge that will still belong to the domain at hand. Again, in the case of formal theories these rules are also formal rules (like deductive rules in logic), but we also allow entirely informal ones, corresponding to "rational thinking" in the case of informal theories.

The epistemic domain together with the corresponding inference rules form the *epistemic theory*.

Each computation we will consider will generate knowledge from some epistemic domain with the help of the corresponding computational process. We will say that such a computation will be *rooted* in this domain. Starting from the so-called *initial knowledge* the computational process will generate *output knowledge* within the given epistemic domain. Depending on the epistemic domain, initial knowledge is given in the form of axioms, definitions, observations, facts, perceptions, etc. The output knowledge may take the form of propositions, theorems or proofs in the case of formal theories, and statements, hypotheses, scientific laws, or predictions in the case of natural sciences. In the case of informal theories (like theory of mind, arts, etc.) the generated knowledge takes the form of conceptualization, behavior, communication, utterances in a natural language, thinking, and knowledge about the world formed mostly in a natural language or in a form of scientific theories and other writings.

From what has been said above one can see that the epistemic domains range from so-called *theory-full* domains corresponding to formal, abstract theories to *theory-less* domains that admit no formal descriptions for capturing e.g. behavior in common life situations (cf. [13]).

In order for a computation to generate knowledge there must be evidence (e.g., a proof) that explains that the computational process works as expected. Such an evidence must ascertain two facts: (i) that the generated knowledge can be derived within the underlying epistemic theory, and (ii) that the computational process generates the desired knowledge.

The latter is the key to the following more formal definition (cf. [18]). In this definition we assume that the input to a computation is part of both the underlying epistemic domain (and thus of the theory) and the initial data of the computational process. Do not forget that although the notation used in the definition formally resembles the notation used in the formal theories, we will also be using it in the case of informal epistemic domains.

Definition 1 Let T be a theory, let ω be a piece of knowledge serving as the input to a computation, and let $\kappa \in T$ be a piece of knowledge from T denoting the output of a computation. Let Π be a computational process and let E be an explanation. Then we say that process Π acting on input ω generates the piece of knowledge κ if and only if the following two conditions hold:

- $(T, \omega) \vdash \kappa$, i.e., κ is provable within T from ω, and
- E is the (causal) explanation that Π generates κ on input ω.

We say that the 5-tuple $C = (T, \omega, \kappa, \Pi, E)$ is a computation rooted in theory T which on input ω generates knowledge κ using computational process Π with explanation E.

When considering epistemic creativity in the sense of human mental ability, one usually thinks of it in the context of a natural language. How could the corresponding computation (seen as knowledge generation) be captured by the above definitions?

First of all, one must bear in mind that the underlying knowledge domain is a domain comprising, in principle, all human knowledge. This knowledge can be seen as a union of various specific knowledge domains which vary from theory-full to theory-less domains. The respective knowledge is thus *heterogeneous knowledge* and natural language serves as an important, and in fact, the only one known mediator among the respective theories. The less formal the knowledge is the more it relies on the natural language. The "inference rules" for heterogeneous domains are a mix of informal and formal rules. That is, when one speaks within theory-less domains, the informal rules of "rational thinking" are used. Otherwise, speaking within theory-full domains one makes use of the rules corresponding to that domain. Natural language provides not only a tool for initial forming and describing a theory, it also provides a unified tool for understanding all theories and "moving" among them. Last but not least, natural language and its semantics provide a link between a theory and the physical world. Only due to natural language and only within a theory one can explicate meaning of the expressions of a natural language, i.e., their semantics. Namely, in our framework the meaning of any expression of a natural language is given by knowledge pertinent to this expression within a certain domain of discourse. This knowledge comes again in the form of a theory stating all contexts and relationships among them in which the expressions at hand can be used. That is, this theory captures the ways in which usage of an expression makes sense in various contexts. Semantics is knowledge and therefore it can be generated by a computation. From this viewpoint all computations, including the computations that generate knowledge based on understanding natural language, bear a homogeneous structure despite the fact that the underlying knowledge as a whole covers many epistemic domains.

The knowledge framework behind a computation over the domain of a natural language will normally be based on *cooperating theories.* This is an extremely complex system since in principle to each word a theory (in our general sense) is attached, controlling the proper use of this word. In general, such a theory depends not only on the word at hand, but also on the context in which the word is being used. In the case of embodied cognitive systems the context does not only refer to the grammatical context, but also to the entire perceptual situation (cf. [16]). All this leads to a complex intertwining of the respective theories working of the internal models of the world. If realized along the lines sketched above, the underlying cooperative theories should display understanding. The problem of understanding is the central problem of AGI and our approach to computation seems to offer a versatile tool for capturing the related issues. This is because it concentrates on the specification of WHAT the sense of understanding is, while postponing the questions HOW this can be realized. Nevertheless, it is fair to state that so far we do not know much about cooperating theories leading to computational understanding.

Second, what computational process is behind a natural language? It is the process running in our heads. Although we do not know the details of how it works, we do know that it generates knowledge that we can describe by natural language as indicated above. And finally, what corresponds to the explanation? Again, it is an explanation in a natural language.

To summarize, we see that natural language is used here as a means for describing the underlying theory-less domain and the inferences over such domain, as well as for explaining the respective computations as performed by the human brain. Note the analogous situation in classical computing where, for example, λ-calculus is used both as a programming language and as the underlying model of computation.

3 COMPUTATIONAL CREATIVITY

3.1 Preliminaries

Any computation as defined in the previous section generates knowledge. Nonetheless, as remarked in the Introduction, this does not necessarily mean that any computation should be seen as a creative process, as a process that generates something new, original, unexpected, surprising, deserving a special interest or having some worthy value as required in epistemic creativity. This "surprise effect" does not happen when an output of a computation can routinely be produced in a straightforward way, following pre-programmed paths corresponding to a priori envisaged circumstances. The majority of current computer programs works in this way. Typical examples include the computation of a function. Such a process can be seen as generating explicit knowledge (i.e., a function value corresponding to the input value) from implicitly described knowledge that is given in the form of an algorithm. There is no room for creativity in such a process. Note that, e.g., various editors and spreadsheets belong to the category of such computations. Operating systems can serve as an example of an interactive non-creative computational process. What they do can be subsumed as an iteration of the following activity: *"if so and so happens, do so and so"*. In computations of this kind no creativity is assumed, since it is not required by the applications at hand.

What about database searches? Here, pieces of knowledge are sought by searching a finite amount of data ("knowledge items") using a specified criterion. Is here some room for creativity? Now the answer is not so simple as in the previous case. In "old fashioned" databases as used in the early days of computing that used to seek an item satisfying a certain condition within the set of structured data, the situation was similar to the previous case. But think about the following case: a "database" (or rather: a knowledge base) containing all knowledge possessed by an average person (whatever it might mean), i.e., knowledge contained in the mind of that person. The query would be as follows: *"name me an animal living in a desert having the same relation to its living environment as has a shark to the ocean"* (the example taken from [16]). In this case, we can obtain an answer "I don't know" (e.g., from a child), or "a camel" (from an average educated person), or "a desert lion" (from an informed animal rights activist), or even "Cataglyphis bicolor" (a desert-dwelling ant also called "the Sahara desert ant"), from some joking entomologist. Now, were there some aspects of creativity in delivering any od these answers? Which of these answers is the best? And, last but not least, what was the mechanism enabling the answering of such a query?

3.2 Analogies

The last example has been an example of analogy solving. Discussions and studies of analogies go back to the ancient philosophers, since analogies have always played in important role in reasoning in logics, science, law and elsewhere. The role of analogy has been intensively studied for years in cognitive science (cf. [8], [10]). The notion of analogy is rarely formally defined. What one can find in the literature, vocabularies and on the web are informal definitions serving to the purpose of the underlying field. Thus, one can find definitions like *"analogy denotes a similarity between like features of two things, on which a comparison may be based;* or *"a comparison between one thing and another, typically for the purpose of explanation or clarification"*, or *"analogy is a figure of language that expresses a set of like relations among two sets of terms"*. In logic, *"analogy is a form of reasoning in which one thing is inferred to be similar to another thing in a certain respect, on the basis of the known similarity between the things in other respects"*.

There are many variants of analogies. For the purpose of knowledge generation we will be especially interested in so-call *explanatory analogies*. Such analogies create understanding between something unknown by relating it to something known. They provide insight or understanding by relating what one does not know with what one knows. Thus, these analogies may be seen as providing elementary creativity steps in deriving new knowledge. This approach where knowledge is not obtained by simply composing pieces of old knowledge has to be contrasted with the classical epistemological procedures of knowledge generation. Such procedures are usually described as extrapolations of repeated observations, or of known facts, as some variants of an induction process. In this process, there is no creativity aspect: knowledge is merely transformed from one form to an other. However, it is reasonable to expect that the ability to create new knowledge must also include the ability to create new explanations, not merely extrapolating or generalizing the past experience.

In order to better understand explanatory analogies, we will need a more formal definition of analogy that will enable us to see the finer details of the envisaged computational process of creating knowledge leading to analogy solution. Therefore, for our purposes the desired definition should fit into the framework of epistemic computations.

The starting point will be to choose a suitable theory in which the respective computations will be rooted. In this respect, note that all informal definitions of analogies involve direct or indirect reference to natural language. Moreover, they are using linguistic expressions like features, relations, similarity, comparison, or explanations. Therefore, a natural choice for such a theory would be a natural language $\mathcal{NL}$ possessing the richness of linguistic expressions needed to understand and resolve analogies. The (informal) rules corresponding to $\mathcal{NL}$ would be those of "rational thinking", and the corresponding computational process will be that produced by the human brain (cf. Definition 1) and the discussion thereafter.

In the following definition (taken from [16]) the adjective *linguistic* will mean that the corresponding expressions, predicates or relations are not described in any formal logical calculus or theory — rather they are described by expressions of a natural language $\mathcal{NL}$ corresponding to the respective pieces of knowledge. These pieces of knowledge form the *knowledge base* of $\mathcal{NL}$. Their validity usually cannot be proved formally but can be known from experience, empirically or from hearsay.

Definition 2 *Let $S = (s_1, \ldots, s_k)$ and $T = (t_1, \ldots, t_k)$ be two sequences of linguistic expressions from $\mathcal{NL}$. If there exists a linguistic k-ary predicate $P \in \mathcal{NL}$ such that both $P(S)$ and $P(T)$ hold and linguistic relations $R_1, \ldots, R_k \in \mathcal{NL}$ such that $R_i(s_i, t_i)$, for $i = 1, \ldots, k$ holds, then we say that S* is analogous to *T w.r.t. predicate P and relations $R_1, \ldots, R_k$.*

Parameters $s_1, \ldots, s_k$ and $t_1, \ldots, t_k$ are called attributes *of S and T, respectively. Relations R_i's are called* similarity relations.

Note that the linguistic expressions, predicates and relations are all described as expressions of a chosen natural language $\mathcal{NL}$.

Definition 3 *Using the notation from Definition 1, given S and T,* analogy solving *is a knowledge generating process whose purpose is to find linguistic predicate P and linguistic relations $R_1, \ldots, R_k$ such that S is analogous to T w.r.t. predicate P and relations $R_1, \ldots, R_k$.*

We say that P is a conjecture *and $P(S)$, $P(T)$ and $R_i(s_i, t_i)$ are the* explanation *of this conjecture.*

To illustrate the use of the introduced formalism, consider again the example from Subsection 3.1. Excerpting from [16]: If $S =$

$(shark, ocean)$ and $T = (camel, desert)$, then we may define predicate $P(x, y)$ as"x lives in y" and R_1 as "both *shark* and *camel* are animals", R_2 as "both *ocean* and *desert* are living environments". Then the claim "x lives in y" is the conjecture and the facts that "*camel* lives in a *desert*" , "*shark* lives in *ocean*" , "both *shark* and *camel* are animals" and "both *ocean* and *desert* are living environments" are its explanation. The previous task is often described as "the relation of *shark* to *ocean* is like the relation of *camel* to *desert*" and abbreviated as *shark : ocean :: camel : desert.*

If all expressions in S are known and only some expressions from T are missing, then S is called the *source* and T is called a *target* of the analogy. Then the whole analogy inclusively of its explanation can be seen as an *explanatory analogy.* The task of finding both the conjecture and its explanation is an act of knowledge discovery. This is because in general the predicates corresponding to the conjecture and the explanations must be discovered among the pieces of knowledge that are at one's disposal.

We have already noted that an explanatory analogy might admit more than one solution. For instance, the solution of analogy *shark : ocean :: ? : desert* could have been either a camel, or a desert lion, or a Sahara desert ant. Under some circumstance, the answer "I don't know" could also be correct. In order to judge the quality and validity of an answer, we must also know the respective explanation. If all explanations are evaluated by an observer as valid, then what answer is the best? In such a case, the best answer would be the one which maximizes the number of relations between the source and the target (i.e., maximizes number k in Definition 2). For instance, in our case, the answer "desert lion" is to be preferred, because in addition to similarity relations R_1 and R_2 it also satisfies relation R_3 "both $shark$ and $desert\ lion$ are predators". The more similarity relations the candidate solution of the incomplete analogy satisfies, the harder it is to come with a different solution. We say that the solution at hand is "hard to vary". According to Deutsch [6], such a solution has a better "explanatory power" than the other competing solutions.

The multitude of answers points to the fact that the answer is observer dependent. The "less knowledgeable" observer might not know about the existence of desert lions and therefore the answer "camel" would sound more plausible to him or her. An observer not knowing any animal living in a desert obviously must answer "I don't know".

3.3 Variants of analogies

Analogies also occur in a number of different forms which can be seen as generalizations or specific cases of our definition of analogy. Let us mention but a few of such instances of analogy.

A more general case is the case of so-called *incomplete analogies*, in which one has to find an analogy between two (or even more) linguistic notions S and T but not all (possible none of the) attributes of neither notion are given. That is, a part of a solution must also be the discovery of the respective attributes of T and S whose pairs correspond to the similarity relations, and the maximization of the number of such pairs. Such problems occur, e.g., in taxonomy dealing with classification of things or concepts based on sharing similar features. In such cases the degree of creativity seems to be higher than in the cases described by Definition 2.

In the opposite direction, a metaphor is a special type of analogy. A *metaphor* is an expression of language that describes a subject by comparing it with another unrelated subject resembling the original subject only in some semantic aspects, on some points of comparison. Both subjects then share the same semantic property which is not immediately apparent from the names of both subjects (cf. the metaphor *"time is money"*) (cf. [10]).

An extended metaphor is *allegory,* in its most general sense. Allegory has been used widely throughout the histories of all forms of art, largely because it readily illustrates complex ideas and concepts in ways that are comprehensible to its viewers, readers, or listeners. Allegories are typically used as literary devices or rhetorical devices that convey hidden meanings through symbolic figures, actions, imagery, and/or events, which together create the moral, spiritual, or political meaning the author wishes to convey (cf. Wikipedia). Re-casting allegory into our framework, allegory usually establishes similarity relations between the narrative story and its possible interpretations in a real or imaginary world. Discovering such relations is a task for allegory creation as well as their projection into the solution of the allegory at hand. The idea is that this projection is not usually obvious at the first sight and its discovery is a task for the observer. In this sense, the similarity relations are "indirectly defined" and depend on the individual taste and knowledge of the observer. Aesthetics and emotions can play an important role in this process. In this way, both creating an allegory by its creator as well as its "deciphering" by an observer are creative acts.

Finally, let us mention the most general and the most important case that plays a crucial role in scientific discovery, and this is the case of the resolution of a "flaw" in a theory. In our framework (cf. Definition 1), the scenario of such a situation is as follows: consider theory T working well over some epistemic domain until one day an input ω to T is found delivering output κ_1. This output is different from output κ_2 which for some reason was expected (e.g. κ_1 disagrees with observations or with experiments). Now the question is, what is the minimal adjustment of theory T such that it would predict output κ_2 on input ω while retaining its ability to work correctly for all other inputs? Clearly, this is another variation on the theme of analogy. This time however, the epistemic theory T itself has become the source and the new theory T' the target of the analogy, and one has to invent new attributes of the target theory preserving as much of the old theory as possible while repairing its flow w.r.t. input ω. Of course, it may happen that theory T is "irreparable" and T' will be completely different from T. History of science knows a lot of such examples (cf. the clash of Darwinism and creationism).

3.4 Efficiency issues in analogy solving

In the framework of epistemic computations one cannot speak about complexity of computations in the classical sense. This is because, in this case, no concrete computational model is used. What can be done for a computation generating complex knowledge, is to describe what partial knowledge or pieces of information are needed in order to generate the knowledge.

Consider the case of solving an explanatory analogy in the form as described by Definition 2 and 3.

The input knowledge for our computation consists of two linguistic predicates $S, T \in \mathcal{NL}$ from a natural language $\mathcal{NL}$, respectively, with $S = (s_1, \dots, s_k)$ and $T = (t_1, \dots, t_k)$. Since we are dealing with explanatory analogy we will assume that some (but not all) attributes t_i's in T are left unspecified. Let $\mathbb{K}$ be the knowledge base that is at the disposal of our computation.

In order to resolve such analogy, we need to discover the following knowledge:

1. we have to check whether an object corresponding to predicate S does exist in $\mathbb{K}$. If not, the answer would be *"I don't know"* and we are done.

2. for each object $T' \in \mathbb{K}$ satisfying predicate T in specified attributes, we check whether in $\mathbb{K}$ there exists
   (a) a k-ary predicate P satisfying $P(S) = P(T')$ (i.e., we are looking for a conjecture). If there is no such P the answer would again be *"I don't know"* and we are done.
   (b) next, we look for linguistic relations of similarity $R_1, \ldots, R_k \in \mathbb{K}$ such that $R_i(s_i, t_i)$, for $i = 1, \ldots, k$ holds. If such relations are found then the answer would return object T', conjecture P and explanations R_i's. Otherwise the answer would again be *"I don't know"* and in either case, we are done.

   If no object T' is found then the answer is *"I don't know"* and we are done.

A more involved procedure would be needed in case the necessary knowledge is not found and we don't want to "give it up". If this happens then it is possible to consult "external sources" such as the web, encyclopedias, monographs, experts, etc. In any case, one can see that resolving explanatory analogies is a quite demanding task, requiring in the worst case knowledge of all items in the underlying knowledge base.

Can we say at least something about the computational complexity of solving explanatory analogies? Well, in any case, when we are dealing with analogies over finite knowledge bases, the previous "algorithm" of finding a solution (if it exists) solves in fact a combinatorial problem over a finite domain and therefore can be solved in finite time.

Obviously, the solution of an analogy problem, and in general, of any creativity task depends on all items in the underlying knowledge base. In order to address the essence of the problem of knowledge discovery in terms of the size of the underlying knowledge base we also use a metaphor, viz. the *metaphor of a mosaic.* Namely, a simplistic view of knowledge discovery is that we seek a piece or pieces of knowledge that fit into a certain unfinished mosaic composed from pieces of knowledge possibly from various domains. Here, "fit" means that the new pieces of knowledge are related to the existing pieces by a certain set of known eligible relations that can be either of a syntactic or a semantic nature. (Note that this was also the case of analogies and metaphors.) Then the *creativity problem* is the task of composing a solution of a problem from finitely many pieces of knowledge that have to be related in a logical way in order to come up with the desired solution. It is interesting to observe that a mosaic where only few pieces are missing can be seen as a hypothesis, or a conjecture. In the case of explanatory analogy solving the size of the mosaic is bounded by the number of attributes of both source and target predicate (parameter k in Definition 2). If n is the size of the knowledge base then solving the mosaic problem requires inspection of at most $\binom{n}{k} = O(n^k)$ subsets of the knowledge base. This means that for sufficiently large n and a fixed k the mosaic problem is of polynomial complexity and thus fixed parameter tractable in k.

A problem similar to the creativity problem — the so-called *domino problem* — has been studied in classical complexity theory (based on Turing machines). In 1966, Berger [2] proved that the domino problem is (classically) undecidable if the pieces of knowledge can be used an arbitrarily number of times. The basic idea of the proof is to have a mosaic to encode a halting computation of a Turing machine.

On the one hand, this explains the difficulty of finding new knowledge in general: there is no (Turing machine) algorithm solving such a task. On the other hand, solutions with a small number of pieces are relatively easy to find by a combinatorial search. It is interesting to note that the unrestricted creativity problem seems to be one of the few known undecidable problems of practical significance.

4 Discovering Knowledge

In [5] Barry Cooper asked, whether information can increase in a computation. Indeed, how could a computation produce information which has not already been somehow encoded in the initial data? This does not seem to be possible. An exhaustive answer to this problem has been given by S. Abramsky in [1]. He concludes that, while information is presumably conserved in a total (closed) system, there can be information flow between, and information increase in, subsystems. Note that in our definition of computation we have considered computational processes rooted in the underlying epistemic domains. This can be viewed as though computations are "observing" their "environments" as captured in their knowledge bases, and indeed, some of them even update the underlying knowledge or gain information from cooperating theories under an interactive scenario. In this case it is possible for such a computation to discover new knowledge.

More precisely, it is possible to go beyond the current knowledge explicitly represented in a knowledge base. This can be done by discovering new relationships among the elements of knowledge, or to discover an element or elements of knowledge that satisfy a required relationship to the existing pieces of knowledge, or to gain a new piece of knowledge from "external sources". By "discover" we readily mean to make something explicitly known, i.e., to obtain explicit knowledge of something for the first time. As an example of new knowledge one can take the resolution of a given analogy.

When speaking about creativity in the sense of knowledge generation one must take into account that knowledge can only be generated from knowledge — this is in fact the essence of our definition of computation. Thus, there exist two opposite processes related to knowledge processing: knowledge acquisition, and knowledge generation.

There are many ways of knowledge acquisition: by reason and logic, by scientific method, by trial and error, by algorithm, by experience, by intuition, from authority, by listening to testimony and witness, by observation, by reading, from language, culture, tradition, conversation, etc.

The purpose of *acquisition processes* is to let the information enter into a system and to order it — via computation — into the existing theory or theories over the pertinent knowledge domains (and represent it in a knowledge base). Such domains take various forms of conceptualizations which are part of the respective theory. A *conceptualization* is a simplified, abstract view of the world representing the given knowledge domain. It captures the objects, concepts and other entities and their relationships existing within the knowledge domain at hand (cf. Wikipedia). Obviously, any knowledge acquisition process builds and updates the existing theories.

The purpose of the *knowledge generation process* is to produce knowledge in reaction to the external or internal requests. One can distinguish two basic principles of knowledge generation: syntactic and semantic knowledge mining. Both methods make use of specific inference mechanisms whose purpose is to discover hidden patterns in the data.

Syntactic knowledge mining works solely over the data representing knowledge. It takes into account only the syntax of the respective data, not their meaning, and also the syntactic inference mechanisms of the underlying theory. Syntactic knowledge mining is the computational process of discovering patterns mainly in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. Finding a pattern corresponding to a certain relationships among data not previously known certainly counts as knowledge discovery.

Semantic knowledge mining is the main engine of creativity. It also looks for hidden patterns in data (knowledge representations) which are semantically, rather than syntactically, related. Usually, based on the semantics of one pattern (the base) and a semantic relation an other pattern (the target), possibly in a different knowledge domain, with a similar semantic structure as the base pattern, is sought satisfying the required relation.

Very often the task of semantic knowledge mining is formed in a natural language. That is, the items to be sought and the relations to be satisfied are described in linguistic terms (as was the case of analogy solving). This complicates the searches since the meaning of linguistic terms must be known. The meaning of each term is, in fact, a piece of knowledge — a theory describing (the properties of) the notion at hand in various contexts. Thus, semantic knowledge mining calls for detecting similarities between theories usually related to different knowledge domains.

Discovering a, in a sense, "parallel" theory to a given one contributes to a better understanding of either theory since it enables to expect relations holding in one theory to also hold in its pendant theory. This is an important element of insight, explanation and understanding. Insight, understanding and explanation make only sense within a theory. They must follow from known facts and rational thoughts.

Unfortunately, general mechanisms of semantic knowledge discovery, explanations and understanding are largely unknown. What we have described here are but the first steps along the respective road.

5 DISCUSSION

We have already remarked at several occasions that, although we see every computation as a knowledge generating process, we cannot automatically consider any computational process to be a creative process. Stating this, when will a computation become a creative process? This seems to be a "million dollar question" of the entire field.

As an example, consider computing x^2, given x. Is this computation a creative process? Compare this with solving the incomplete analogy *shark : ocean :: ? : desert*. Where is the difference? We even know for both cases an algorithm leading to an answer. So where is a difference? Why is finding a solution of the former task considered to be a "non-creative" task whereas finding a solution of the latter is considered to be a creative task?

Well, there seem to be two important differences. In solving the first task one can compute directly with x and manipulate it as the computation requires. In the second case, the computation requires to discover other notions contained in the knowledge base, and the answer depends on what knowledge is stored in the knowledge database at that time. As we have seen, the solution of the second task need not be unique. And the second difference is in the complexity of explanations. While in the first case we must offer an explanation as required by Definition 1, in the second case, in principle, we must offer two explanations: one as asked by Definition 1 related to the correctness of the computation, and the second one required by Definition 2 concerning the correctness of analogy drawing. In general, is there a clear cut between non-creative and creative computational knowledge generation? Nevertheless, the extreme cases can be distinguished.

One might think that there is one more difference. Namely, that in the first case we do not need to know the semantics of x whereas in the second case it is necessary to know the semantics of the "parameters" of the analogy. But this is not true — both computations proceed without knowing the semantics of the respective notions.

Thus, as it appears, in creative knowledge generation (i.e., in computational creativity) the resulting knowledge depends, in addition to the discovery algorithm, on the contents of the underlying knowledge base. The result need not be uniquely defined and in some case need not be defined at all. The respective "creative computation" must work over whatever complete or incomplete knowledge base over the domain of the natural language at hand.

An aspect that seems implicit in "epistemic creativity" is that it isn't driven by the search for a pre-determined answer. In other words, creativity seems be synonymous with "unanticipated solution". In this context the underlying computational process is divergent since it leads to many answers, solutions, knowledge items even from domains that are not internal already but may be imported from elsewhere. Thus creativity seems to involve the generation of options that do not follow by mere deterministic reasoning. (If an artist has found a style that he can repeat, the question is whether it remains a creative process after the first time.)

The bottom line seems that a creative process is not a special type of computation to begin with but a whole collection of computations as also seen from the schema of resolving explanatory analogies in Subsection 3.4. This process is guided at best by some overall triggering process. This latter process may also hold a criterion for judging the computations or rather, the knowledge they come up with, a kind of objective function (which may not be well determined nor a "function" either). In the case of analogies we opted for the "hard to vary" criterion. In any case, this would mean that the question "when is a computation creative" is perhaps not the proper one to ask, if we accept that it is rather a complex process of "divergent computations".

Interestingly, in [12] the author attributes the difference between creative and non-creative *mental processes* not to the underlying computational/functional mechanisms, but rather to the way in which the mental process is experienced. This, however, throws no light on the nature of the underlying mechanisms.

In the context of computational creativity our analysis of analogy solving has revealed that the larger the knowledge base the greater the potential for discovering new knowledge. In order to have the knowledge base as large as possible it must potentially involve all the existing human knowledge and the creative agent must have a command of natural language in order to be able to navigate among various knowledge domains. Along these lines it appears that among the main obstacles of the progress in AGI is our insufficient knowledge of natural language processes concerned with the interactions between computers and human (natural) languages and representation of knowledge accessible to natural languages. Automatic procedures building the respective knowledge bases must be sought (cf. [16]).

Finally, one remark regarding the series of recent writings and interviews of one of the world top thinkers, the prominent British physicist David Deutsch (cf. [7]). At these occasions he has repeatedly stressed that *"The field of artificial (general) intelligence has made no progress because there is an unsolved philosophical problem at its heart: we do not understand how creativity works."*

In spite of what is known about computational creativity (cf. [4]) and despite of the enormous activities in this field, there is something in Deutsch's statement. What is still missing in all known approaches, are the phenomenal issues related to creativity. The "phenomenal component" of creativity seems to be required for a genuine understanding and realization of creative acts. In our approach we have covered up this problem by the requirement of a full mastering of the natural language. This appears to be impossible without engagement with issues around consciousness and free will, and this is why we have stressed the central role of natural language in epistemic creativity processes.

6 CONCLUSION

Our approach is consistent with the modern philosophical view accepted since ancient times that creativity is a form of discovery of new knowledge rather than some kind of inspired guessing. In this discovery process the role of natural language is indispensable since it serves as a universal language bridging various theory-less knowledge domains serving as knowledge base for a knowledge discovery process. Our approach to the problems of computational creativity via the epistemic view of computations offers a natural and uniform framework for the investigation of such problems. Under this view, computational creativity is simply seen as a specific kind of computational knowledge discovery in the underlying knowledge base. The richer the knowledge base the higher the potential for creativity is possessed by the corresponding computations. From this viewpoint, the classical, "non-creative" computational processes are but a special, in a sense "degenerated" kind of computations that do not make use of epistemic theories corresponding to knowledge domains described by explicit knowledge. The epistemic view of computations points to the full capability of computations by revealing their creative potential already in their very definition.

ACKNOWLEDGEMENTS

The research of the first author was partially supported by ICS AS CR fund RVO 67985807 and the Czech National Foundation Grant No. 15-04960S. We thank the anonymous reviewers for their insightful comments that have helped to improve the final version of our paper.

REFERENCES

[1] S. Abramsky. Information, processes and games. In J. van Benthem and P. Adriaans, editors, Handbook of the Philosophy of Information, pages 483549. Elsevier Science Publishers, Amsterdam, 2008

[2] Berger, R.: The undecidability of the Domino Problem, Memoirs of the American Mathematical Society 66, 1966.

[3] Bishop, J.M., (2009), A Cognitive Computing fallacy? Cognition, computations and panpsychism, Cognitive Computing 1:3, pp. 221-233

[4] Boden, M.: The Creative Mind: Myths and Mechanisms (Weidenfeld/Abacus & Basic Books, 1990; 2nd edn. Routldge, 2004)

[5] Cooper, B.: Turing centenary: The incomputable reality Nature 482,465, 23 February 2012

[6] D. Deutsch, *The Beginning of Infinity. Explanations That Transform the World.* Penguin, 2011, 496 pages

[7] D. Deutsch, "Creative Blocks", AEON Magazine, 02 October 2012

[8] G. Edelman, *Second Nature: Brain Science and Human Knowledge.* Yale University Press, 2006, 224 p.

[9] Koestler, A.: The Act of Creation, Penguin Books, New York, 1964.

[10] G. Lakoff, and M. Johnson, *Metaphors We Live By.* Chicago, IL: The University of Chicago Press, 1980

[11] Mokyr, J.: Mobility, Creativity, and Technological Development: David Hume, Immanuel Kant and the Economic Development of Europe. Session on Creativity and the Economy, German Association of Philosophy, Berlin, Sept. 18, 2005. In G. Abel, ed., Kolloquiumsband of the XX. Deutschen Kongresses fr Philosophie, Berlin 2006, pp. 1131-1161.

[12] Nanay, B.: An experiential account of creativity. In: Elliot Paul and Scott Barry Kaufman (eds.): The Philosophy of Creativity. Oxford: Oxford University Press, 2014, pp. 17-35.

[13] Valiant, L.: Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World, Basic Books, (2013)

[14] van Gelder, T.: What might cognition be, if not computation? The Journal of Philosophy, Vol. 92, No. 7. (Jul., 1995), 345-381.

[15] J. van Leeuwen, J. Wiedermann: Knowledge, representation and the dynamics of computation. To appear in: G. Dodig-Crnkovic, R. Giovagnoli (Eds): Representation and Reality: Humans, Animals and Machines, 2015, Berlin: Springer

[16] Wiedermann, J.: The creativity mechanisms in embodied agents: An explanatory model. In: 2013 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2013, pp. 41-45.

[17] Wiedermann, J. van Leeuwen, J.: Rethinking computation. In: *Proc. 6th AISB Symp. on Computing and Philosophy: The Scandal of Computation - What is Computation?*, AISB Convention 2013 (Exeter, UK), AISB, 2013, pp. 6-10

[18] Wiedermann, J., van Leeuwen, J.: Computation as knowledge generation, with application to the observer-relativity problem. In: *Proc. 7th AISB Symposium on Computing and Philosophy: Is Computation Observer-Relative?*, AISB Convention 2014 (Goldsmiths, University of London), AISB, 2014

[19] Wiedermann, J.,van Leeuwen, J.: What is Computation: An Epistemic Approach. In: G. Italiano *et al.* (Eds), *SOFSEM 2015: Theory and Practice of Computer Science*, Proceedings, Lecture Notes in Computer Science Vol 8939, Springer, 2015, pp. 1-13.