What if machines could be creative?

Fabian Suchanek, Colette Menard, Meghyn Bienvenu, Cyril Chapellier

What if machines could be creative?

Fabian M. Suchanek[1], Colette Menard[2],
Meghyn Bienvenu[3], Cyril Chapellier

[1] Télécom ParisTech, [2] STIM, [3] LIRMM Montpellier; France

1 Introduction

Computational creativity is the field of research that is concerned with making computers creative. In this paper, we focus on a particular subfield of this domain, *combinatorial creativity*. This subfield is concerned with combining components of existing concepts into new concepts. For example, we can combine the concept of a watch with the capabilities of a smartphone – and obtain a smartwatch. Or we could combine a bus service with the characteristics of a taxi and obtain a bus service on-demand.

Combinatorial creativity serves different purposes: it can be used to develop new business ideas, to find plots for books or movies, to understand human creativity, to disrupt conventional assumptions, and to find design alternatives. In the long run, the dream is that computational creativity could help humans come up with new inventions.

With this demo, we take a first step in this direction. We propose a system for computer-assisted brainstorming, i.e., for generating hypotheses to aid human creativity. More precisely, given an input concept (such as "car"), our system proposes variations on it (such as "What if a car did not have wheels?", or "What if a car could park automatically?"). This paper explores different ways of generating such proposals.

1.1 Related Work

Combinatorial creativity has first been studied in the cognitive sciences [3]. These works do not focus on generating new concepts automatically. More computationally oriented works have investigated fictional ideation, which generates new concepts for narratives [7], as well as analogies, amalgams, and conceptual blending (e.g., [4, 2, 11]). The latter operations blend two concepts rather than varying a single concept, as we do. The COINVENT project [9] aims to develop computational models of concept invention, but has not yet done so. Also loosely related is work on non-standard reasoning in description logics, in which concepts are generated e.g., for the purposes of semantic matchmaking [8] or for discovering negative constraints in ontologies [5]. A more detailed discussion of related work can be found in [10].

2 Preliminaries

We will describe concepts using description logics (DLs) [1]. We assume that the reader is familiar with DLs. We concentrate here on the DL $\mathcal{EL}^u$, which allows for conjunctions, qualified existential restrictions, and the universal role u. We assume a given acyclic terminology $\mathcal{T}$, which defines concept names in terms of concept expressions. Fig. 1 shows an example. In line with [10], we will see $\mathcal{T}$ as a function that maps a concept name to a conjunction of concepts. In the example, $\mathcal{T}(Car) = Vehicle \sqcap \exists receives.Steering$. As in [10], we assume a total order on concept names. This allows us to define the function $child_i^{\mathcal{T}}(C)$, which retrieves the i^{th} child of C, i.e., the i^{th} concept name among all concept names whose definition contains C as a conjunct. In the example, $child_2^{\mathcal{T}}(Vehicle) = Plane$. As in [10], we assume that all conjunctions are in normal form, and that their conjuncts are ordered. This allows us to speak of "the i^{th} conjunct" of a conjunction. In the example, the 3^{rd} conjunct of $\mathcal{T}(Plane)$ is $\exists has.Wing$.

$$
\begin{array}{lcl}
Car & \equiv & Vehicle \sqcap \exists receives.Steering \\
Plane & \equiv & Vehicle \sqcap \exists hasProperty.Fast \sqcap \exists has.Wing \\
Fast & \equiv & \exists antonym.Slow
\end{array}
$$

Fig 1.: An example terminology $\mathcal{T}$, inspired from [6]

In our accompanying full paper [10], we have developed a language to express modifications of concepts. For two ordered conjunctions C, D in normal form, a role r, and an integer i, its main operators are:

Addition $(C+D)$ joins two concepts in a conjunction, $C+D = C \sqcap D$. In Fig. 1, $\mathcal{T}(Car)+\exists has.Propeller = Vehicle \sqcap \exists receives.Steering \sqcap \exists has.Propeller$.

Subtraction $(C -_i D)$ removes from C the i^{th} conjunct that is subsumed by D. In Fig. 1, $\mathcal{T}(Plane) -_2 \top = Vehicle \sqcap \exists has.Wing$.

Succession $(C \rightarrow_i \exists r.D)$ finds the i^{th} conjunct of C that is subsumed by $\exists r.D$. It is necessarily of the form $\exists r'.D'$. The operation then returns D'. In Fig. 1, $\mathcal{T}(Plane) \rightarrow_1 \exists has.\top = Wing$.

Selection $(C \uparrow_i D)$ returns the i^{th} conjunct of C that is subsumed by D. In Fig. 1, $\mathcal{T}(Fast) \uparrow_1 \exists antonym.\top = \exists antonym.Slow$.

See [10] for formal definitions and more details. We call a formula built from concepts using the above operators and the function $\mathcal{T}$ a *class expression*.

3 Proposing Concept Variations

We now go beyond our work in [10] and investigate systematic means of proposing variations of a given concept.

Definition 1 (Inspirator): An inspirator for a given terminology is a "What if" question that contains a variable x and a class expression over x.

We give some examples of such inspirators, using the terminology $\mathcal{T}$ of Fig. 1.

Negation: "What if a x did not have a $\mathcal{T}(x) \rightarrow_i \exists has.\top$?"
If we apply this inspirator to $x = Car$, we obtain "What if a Car did not have a SteeringWheel?". For such inspirators to work, concept names (and role names) have to be human readable. We defer the principled translation of class expressions to natural language to future work.
Antonym: "What if a x was $\exists hasProperty.(\mathcal{T}(\mathcal{T}(x) \rightarrow_i \exists hasProperty.\top) \rightarrow \exists antonym.\top$?"
With $x = Plane$ and $i = 1$, this inspirator yields "What if a plane was slow?". If the antonym does not exist in the terminology, the succession cannot be evaluated [10], and the inspirator fails.
Sibling: "What if a x $\mathcal{T}(child_i^{\mathcal{T}}(\mathcal{T}(x) \uparrow_j \top)) \uparrow_k \exists u.\top)$?"
With $x = Car$ with $i = 2, j = 1, k = 2$, this yields "What if a car had wings?".
Automation: "What if a x could $\mathcal{T}(x) \rightarrow_i \exists receives.\top$ automatically?"
With $x = Car$ and $i = 1$, this yields "What if a car could steer automatically?".
Synecdoche: "What if a x was a $\mathcal{T}(x) \rightarrow_i \exists has.\top$?"
With $x = Plane$ and $i = 1$, this yields "What if a plane was a wing?". That may seem absurd, but there are indeed such devices[1].
Enabling: "What if a x was able to $\mathcal{T}(x) \rightarrow_i \exists notCapableOf.\top$?"
Assuming that the terminology stated that a car cannot fly, this could yield "What if a car was able to fly?".
Expanding: "What if a x was $\mathcal{T}(\mathcal{T}(x) \rightarrow_i \exists u.\top) \uparrow_j \top$?"
Assuming that the terminology defined a steering wheel to be round, this could yield "What if a car was round?", with $x = Car$ and $i = 1$.

In all of these cases, the inspirator can propose a variation that is already part of the original concept. In this case, the proposal has to be discarded. More inspirators can be added in the future.

4 Demo

We have implemented the above inspirators in a Java program. For the terminology, we use ConceptNet [6], a large knowledge base of commonsense facts. ConceptNet knows, e.g., that cars have wheels, and that they are used for locomotion. This distinguishes it from instance-based knowledge based such as YAGO, DBpedia, and Wikidata. As in [10], we remove relations that describe words (EtymologicallyDerivedFrom, etc.), as well as relations that describe agents and events. To clean out noise, we also remove all definitions that have 2 or less conjuncts. This leaves us with a terminology of 5485 concept definitions in $\mathcal{EL}^u$.

Our demo allows users to generate variations of a given input concept. For example, the user can choose to vary the concept of a car, obtaining suggestions such as "What if a car were inexpensive?". The goal is not to generate concepts so that each and every one of them is an implementable innovation. Rather, the intention is to do computer-assisted brainstorming: to generate concepts liberally, in the hope that some of them may inspire. In this vein, many of the

[1] `https://en.wikipedia.org/wiki/Flying_wing`

suggestions we generate are just nonsensical. However, for most concepts, our inspirators generate reasonable variations. In some cases, these are creative or funny. For example, the tool proposes shoes made of cotton, keyboards that are used to get into your house, or cars that can park automatically. Interestingly, the tool also proposes the title of our paper, "What if a machine was creative?".

5 Conclusion

The goal of our demo is two-fold. First, we want to demonstrate a first step towards making computers automatically propose new concepts. Second, we want to gain feedback from the audience about the usefulness of the generated concepts. This feedback will allow us to better steer the process of concept generation in the future. The audience, too, will benefit from the demo. First, the audience can see to what degree machines can generate new concepts. Second, even if the concepts are not all reasonable, they are at least entertaining ("What if a car could eat spaghetti?").

For future work, we plan to investigate how reasonable concepts can be generated with higher probability. We want to study how factual and physical constraints could be integrated, how concepts could be ranked, and how we can measure the usefulness of the generated concepts. Our demo is available at `https://suchanek.name`.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. T. R. Besold and E. Plaza. Generalize and blend: Concept blending based on generalization, analogy, and amalgams. In *ICCC*, 2015.
3. M. A. Boden. *The Creative Mind: Myths and Mechanisms*. Routledge, 2004.
4. B. Falkenhainer, K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41, 1989.
5. S. Ferré and S. Rudolph. Advocatus diaboli - exploratory enrichment of ontologies with negative constraints. In *EKAW*, 2012.
6. H. Liu and P. Singh. Conceptnet. *BT Technology Journal*, 22(4), Oct. 2004.
7. M. Llano, R. Hepworth, S. Colton, J. Charnley, and J. Gow. Automating fictional ideation using conceptnet. In *AISB14 Symp. on Computational Creativity*, 2014.
8. T. D. Noia, E. D. Sciascio, and F. M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Artif. Int. Research*, 29, 2007.
9. M. Schorlemmer, A. Smaill, K.-U. Kühnberger, O. Kutz, S. Colton, E. Cambouropoulos, and A. Pease. Coinvent: Towards a computational concept invention theory. In *ICCC*, 2014.
10. F. M. Suchanek, C. Menard, M. Bienvenu, and C. Chapellier. Can you imagine... a language for combinatorial creativity? . In *ISWC*, 2016.
11. T. Veale and D. ODonoghue. Computation and blending. *Cogn. Ling.*, 11, 2000.