Project Description:

AI Models of Conceptual Abstraction and Analogy-Making

Last updated May 12, 2023

Conceptual abstraction and analogy-making are key abilities underlying humans' capacity to learn, reason, and robustly adapt their knowledge to new domains. Despite of a long history of research on constructing AI systems with these abilities, no current AI system is anywhere close to a capability of forming humanlike abstractions or analogies.

In this project we will further develop the *active symbol architecture* proposed by Hofstadter et al. [12], especially building on the Copycat [13] and Metacat [16] architectures for analogy and metacognition, as well as the Situate visual "situation-understanding" architecture, which aims to combine an active symbol architecture with deep learning (a preliminary version is described in [20]). We will apply ideas from these architectures to idealized domains, including letter-string analogies [11], Bongard problems [2, 7], and the Abstraction and Reasoning Corpus [4], as well as to domains inspired by real-world tasks. We will also compare the performance of the novel architecture to other methods for abstraction and analogy, including deep learning and program synthesis approaches [5, 9, 15].

This project, being carried out at the Santa Fe Institute, is part of a larger multi-institution effort entitled "Building Diverse Intelligences through Compositionality and Mechanism Design". The larger effort is focused on how intelligent behavior emerges from complex systems—in particular, the mechanisms by which semi-independent, adaptive processes are composed to function as an intelligent, higher-level whole. We will compare several model systems in which such processes occur in order to develop broad insights about intelligent systems.

Below is an excerpt from [18] with a brief description of the Active-Symbol / Copycat architecture.

Active Symbol Architecture

In the 1980s, Hofstadter designed a general architecture for abstract perception and analogy-making called the "active symbol architecture," based in part on Hofstadter's notion of active symbols in the brain: "active elements [groups of neurons] which can store information and transmit it and receive it from other active elements" [10]—and in part on inspiration from information processing in other complex systems such as ant colonies and cellular metabolism.

The active symbol architecture was the basis for several AI programs exploring abstract perception and analogy-making, many of which were described in Ref. 12. The best-known example is Hofstadter and Mitchell's Copycat program [13, 17]. The name "Copycat" is a humorous reference to the idea that the act of making an analogy is akin to being a "copycat"—that is, understanding one situation and "doing the same thing" in a different situation. A key idea of the Copycat pro gram is that analogy-making should be modeled as a process of abstract perception. Like sensory perception, analogy-making is a process in which one's prior concepts are activated by a situation, either perceived via the senses or in the mind's eye; those activated concepts adapt to the situation at hand and feed back to affect how that situation is perceived.

Hofstadter and Mitchell developed and tested Copycat using the domain of letter-string analogy problems, created by Hofstadter [11]. The following are some examples from this domain:

> If the string **abc** changes to the string **abd**, what does the string **pqrs** change to?
>
> If the string **abc** changes to the string **abd**, what does the string **ppqqrrss** change to?
>
> If the string **abc** changes to the string **abd**, what does the string **srqp** change to?
>
> If the string **abc** changes to the string **abd**, what does the string **xyz** change to?
>
> If the string **axbxcx** changes to the string **abc**, what does the string **pzqzrzsz** change to?

While these analogy problems are idealized "toy" problems, they are, similar to Ravens' matrices, designed to capture something of the essence of real-world abstraction and analogy-making. Each string is an idealized "situation" containing objects (e.g., letters or groupings of letters), relations among objects, events (a change from the first to second string), and a requirement for abstraction via what Hofstadter termed *conceptual slippages* [11] (e.g., the role of "letter" in one situation is played by "group" in another situation, or the role of "successsor" in one situation is played by "predecessor" in another situation).

In the Copycat system, the process of analogical mapping between two situations (here, letter strings) is interleaved with the process of building representations of those situations, with continual feedback between these processes. This is achieved via four interacting components: a *concept network*, which contains the system's prior knowledge in symbolic form; a *workspace*, which serves as a working memory in which representation of and mappings between the input situations takes place; a set of *perceptual agents*, which—competitively and cooperatively—attempt to adapt the system's prior knowledge to the input situations over a series of time steps; and a *temperature* variable, which measures the quality and coherence of the system's representations and mappings at a given time, and which feeds back to control the degree of randomness of the perceptual agents. When the system is far from a solution, the temperature is high, and the perceptual agents' actions are more random; as the system zeroes in on a coherent solution, the temperature falls, and the perceptual agents are more deterministic.

Figure 1 illustrates the architecture of the Copycat program. Figure 1(a) illustrates part of the program's *concept network*, which contains the program's prior (symbolic) knowledge about the letter-string domain, corresponding to long-term memory. The concept network models a symbolic "semantic space," in which concepts are nodes (ellipses) and links (lines) between between concepts represent semantic distance, which can change with context during a run of the program. A concept (e.g., *letter-group*) is activated when instances of that concept are discovered in the workspace, and in turn, activated concepts (the program's "active symbols") trigger perceptual

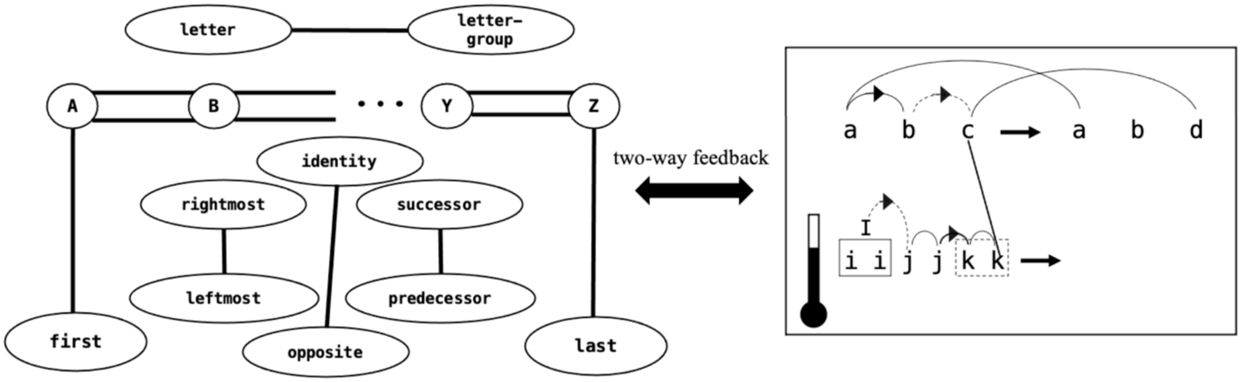


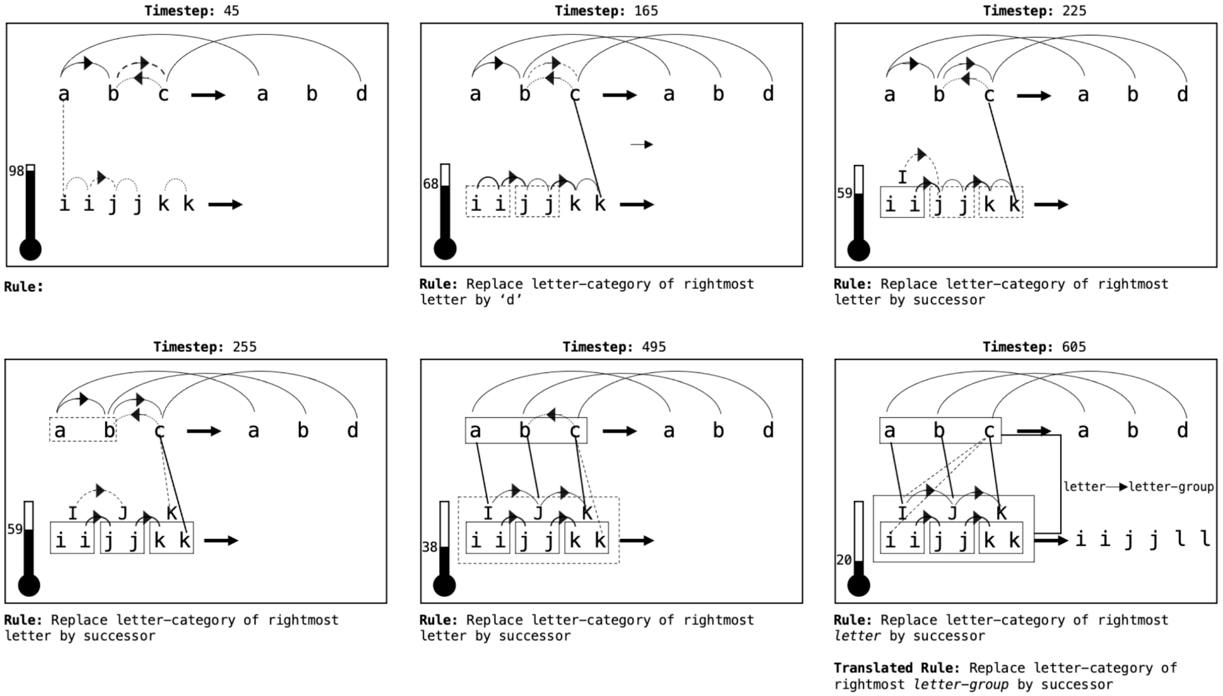
Figure 1: (a) Illustration of part of Copycat's concept network. (b) Illustration of Copycat's workspace, during a run of the program.

agents that attempt to discover additional instances. Activation can also spread between conceptual neighbors. Activation decays over time if not reinforced.

Figure 1(b) illustrates the program's *workspace*, a short-term memory, inspired by blackboard systems, [6] in which perceptual agents construct (and sometimes destroy) structures (relations, groupings, correspondences, and rules) that form the program's current representation of the input situations and the analogy between them, at any given time during a run. Dashed lines or arcs represent structures with low confidence; solid lines or arcs represent structures with high confidence; the confidence of a structure can change during the run and structures can be destroyed depending on their confidence. A temperature variable (represented by the thermometer in the bottom right of the workspace) measures the quality of the current structures and feeds back to affect the randomness of the perceptual agents.

Figure 2 gives the state of the workspace at selected timesteps during a run of the program, illustrating how the program constructs representations of, and analogies between, its input situations. The workspace serves as a global blackboard on which agents explore, build, and destroy possible structures. The actions of agents are probabilistic, and depend on the current state of the workspace, concept network, and temperature. Perceived correspondences between objects in different situations (here letters and letter-groups) lead to conceptual slippages (e.g., *letter* slips to *letter-group*) that give rise to a coherent analogy. Details of Copycat's operations are described in Ref. 17.

In summary, the Copycat program is an example of Hofstadter's active symbol architecture, in which symbolic concepts become activated via bottom-up perceptions, spread activation to semantically related neighbors, and influence perception in a top-down manner by triggering probabilistic perceptual agents to find instances of the associated concepts in a blackboard-like workspace. In this way, processing in the system consists of a continual interplay between bottom-up and top down processes. A temperature variable controls the degree of randomness in the



Timestep: 45

Rule:

Timestep: 165

Rule: Replace letter-category of rightmost letter by 'd'

Timestep: 225

Rule: Replace letter-category of rightmost letter by successor

Timestep: 255

Rule: Replace letter-category of rightmost letter by successor

Timestep: 495

Rule: Replace letter-category of rightmost letter by successor

Timestep: 605

Rule: Replace letter-category of rightmost *letter* by successor

Translated Rule: Replace letter-category of rightmost *letter-group* by successor

Figure 2: State of the workspace at six different timesteps during a run of Copycat (adapted from Ref. 17).

system and in turn is dynamically determined by the quality of perceptual structures constructed by the system. Coherent representations of input situations, and analogies between them, result from the perceptual structures constructed by these probabilistic agents. A central idea underlying the active symbol architecture is that, in analogy-making, the mapping process cannot be separated from the representation-building process—these must be interleaved. This is a central point of disagreement with the structure-mapping engine approach described in the previous section (see also Ref. 3).

As described in Ref 13, Copycat's emergent dynamics show a gradual transition from a largely bottom-up (perception-driven), random, and parallel mode of processing—in which many possible representations of the input are explored—-to one that is largely top-down (concept-driven), deterministic, and serial. Copycat does not fit neatly into the traditional AI dichotomy between symbolic and neural systems; rather it incorporates symbolic, subsymbolic, and probabilistic elements. The architecture resonates with several ideas in psychology, psychophysics, and neuroscience, such as the Global Workspace hypothesis of Baars et al., [1, 22] in which multiple, parallel, specialist processes compete and cooperate for access to a global workspace, and the proposal that visual cortex areas V1 and V2 work as "'active blackboards' that integrate and sustain the result of computations performed in higher areas" [8, 21]. Copycat also resonates with the idea of neural "object files" [14]—temporary and modifiable perceptual structures, created on the fly in working memory, which interact with a permanent network of concepts. The system's dynamics are also in accord with Treisman's [23] notion of perception as a shift from parallel, random, "pre-attentive" bottom-up processing and more deterministic, focused, serial, "attentive" top-down processing.

Mitchell and Hofstadter showed how Copycat was able to solve a wide selection of letter-string problems; they also described the program's limitations [13, 17]. The Copycat program inspired numerous other active-symbol-architecture approaches to analogy-making, some of which are de scribed in Ref. 12, as well as approaches to music cognition, [19] image recognition, [20] and more general cognitive architectures [1].

It bears repeating that Copycat was not meant to model analogy-making on *letter strings* per se. Rather, the program was meant to illustrate—using the letter-string analogy domain—a domain independent model of high-level perception and analogy. However, the program has several limitations that need to be overcome to make it a more general model of analogy-making. For example, Copycat's concept network was manually constructed, not learned; the program illustrated how to adapt pre-existing concepts flexibly to new situations, rather than how to learn new concepts. Moreover, the program was given a "source" and "target" situation to compare rather than having to retrieve a relevant situation from memory. Finally, the program's architecture and parameter tuning were complicated and somewhat ad hoc. Additional research on all of these issues is needed to make active symbol architectures more generally applicable.

References

[1] B. J. Baars and S. Franklin. How conscious experience and working memory interact. *Trends in Cognitive Sciences*, 7(4):166–172, 2003.

[2] M. M. Bongard. *Pattern Recognition*. Spartan Books, 1970.

[3] D. J. Chalmers, R. M. French, and D. R. Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence*, 4(3):185–211, 1992.

[4] F. Chollet. On the measure of intelligence. *arXiv:1911.01547*, 2019.

[5] K. Ellis, C. Wong, M. Nye, M. Sabl´e-Meyer, L. Cary, L. Morales, L. Hewitt, A. Solar-Lezama, and J. B. Tenenbaum. DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. *arXiv:2006.08381*, 2020.

[6] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR)*, 12(2), 1980.

[7] H. E. Foundalis. Index of Bongard Problems. URL http://www.foundalis.com/res/bps/bpidx.htm.

[8] C. D. Gilbert and M. Sigman. Brain states: Top-down influences in sensory processing. *Neuron*, 54(5):677–696, 2007.

[9] F. Hill, A. Santoro, D. G. T. Barrett, A. Morcos, and T. Lillicrap. Learning to make analogies by contrasting abstract relational structure. In *Proceedings of the International Conference on Learning Representations, ICLR*, 2019.

[10] D. R. Hofstadter. *G¨odel, Escher, Bach: an Eternal Golden Braid*. Basic Books, New York, 1979.

[11] D. R. Hofstadter. Analogies and roles in human and machine thinking. In *Metamagical Themas*, chapter 24. Basic Books, 1985.

[12] D. R. Hofstadter. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamen tal Mechanisms of Thought*. Basic Books, New York, 1995.

[13] D. R. Hofstadter and M. Mitchell. The Copycat project: A model of mental fluidity and analogy-making. In K. J. Holyoak and J. A. Barnden, editors, *Advances in Connectionist and Neural Computation Theory*, volume 2, pages 31–112. Ablex Publishing Corporation, 1994.

[14] D. Kahneman, A. Treisman, and B. J. Gibbs. The reviewing of object files: Object-specific integration of information. *Cognitive Psychology*, 24(2):175–219, 1992.

[15] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[16] J. B. Marshall. Metacat: A self-watching cognitive architecture for analogy-making. In *Proceed ings of the Twenty-Fourth Annual Conference of the Cognitive Science Society*, pages 631–636. Routledge, 2019.

[17] M. Mitchell. *Analogy-Making as Perception: A Computer Model*. MIT Press, 1993.

[18] M. Mitchell. Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 1501(1):79–101, 2021.

[19] E. Nichols and D. Hofstadter. Musicat: A model of music perception and expectation. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*, 2009.

[20] M. H. Quinn, E. Conser, J. M. Witte, and M. Mitchell. Semantic image retrieval via active grounding of visual situations. In *Proceedings of the International Conference on Semantic Computing*, pages 172–179, 2018.

[21] P. R. Roelfsema and F. P. de Lange. Early visual cortex as a multiscale cognitive blackboard. *Annual Review of Vision Science*, 2, 2016.

[22] M. Shanahan. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and Cognition*, 15(2):433–449, 2006.

[23] A. Treisman. Feature binding, attention and object perception. *Proceedings of the Royal Society, Series B*, 6:1295–1306, 1998.