
Designing Interactive, Collaborative Agents

The Interactive Agents Group (IAG) at DeepMind has the aspirational goal of building agents that interact with humans and communicate using natural language to collaboratively solve problems. The abstract of their 2020 paper summarizes their initial attempt to achieve this goal:

A common vision from science fiction is that robots will one day inhabit our physical spaces, sense the world as we do, assist our physical labors, and communicate with us through natural language. Here we study how to design artificial agents that can interact naturally with humans using the simplification of a virtual environment. This setting nevertheless integrates a number of the central challenges of artificial intelligence (AI) research: complex visual perception and goal-directed physical control, grounded language comprehension and production, and multi-agent social interaction. To build agents that can robustly interact with humans, we would ideally train them while they interact with humans. However, this is presently impractical. Therefore, we approximate the role of the human with another learned agent and use ideas from inverse reinforcement learning to reduce the disparities between human-human and agent-agent interactive behavior. [...]

The agents in the IAG paper interact in a simulated environment and learn to perform relatively simple skills that involve searching for and identifying objects by their shape or color and then moving them to different locations. During training two agents participate in what are referred to as language games – borrowing the term from Wittgenstein.

In each game, one agent takes the role of the *setter* and the other the role of the *solver*. Both agents can move around and examine the objects in the environment. Given an externally generated prompt, the setter issues instructions to the solver constrained by the prompt and the solver attempts to carry out the instructions.

The role of the setter was therefore primarily to explore and understand the situational context of the room (its layout and objects) and to initiate diverse language games constrained by the basic scaffolding given by the prompt. They communicate with one another using natural language which they have to learn through their interactions. The tasks are simple. Learning a language is complicated.

In class this year we will carry out a gedanken experiment for the purpose of exploring whether something like the approach described in the paper might be employed to train an agent to take on the more demanding task of a programmer's apprentice. In carrying out this experiment, we will consider seriously the problem of transforming natural language descriptions of algorithms, computer programs, and programming strategies into the unnatural language of working code.



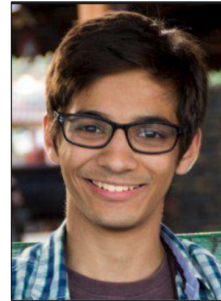
Greg Wayne



Rishabh Singh



Dan Abolafia



Yash Savani



Gene Lewis



Matt Botvinick



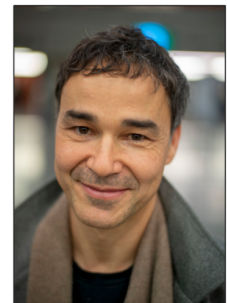
Eve Clark



Josh Merel



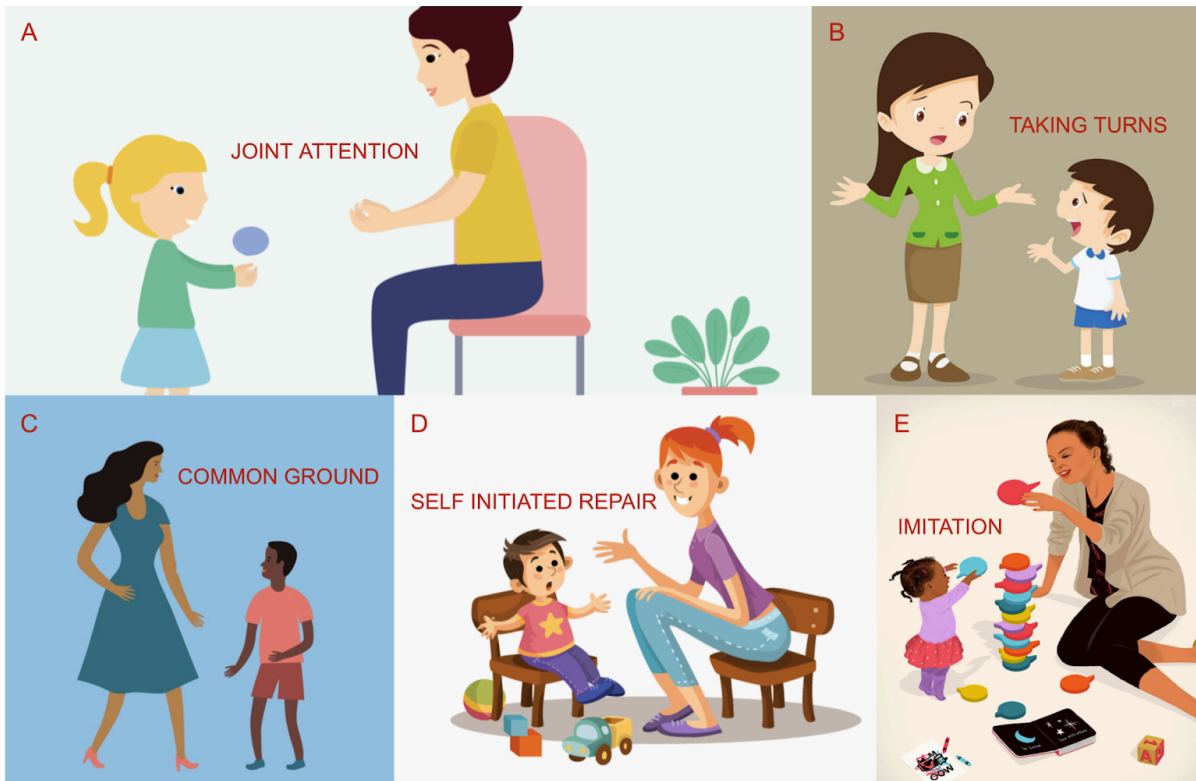
Felix Hill



Nando de Freitas

To provide technical advice and direction on how to tackle this problem we have enlisted the help of several accomplished research scientists from DeepMind, Google Brain, and Stanford. Four of whom have volunteered to give presentations on topics directly relevant to the experiment we are conducting and contribute to class discussions during the first half of the quarter, and all of them willing to provide advice on class projects relating to their expertise. In addition, we will review several relevant invited talks from the 2018 and 2019 classes, including Peter Battaglia, Matt Botvinick, Jessica Hamrick, Tejas Kulkarni, Josh Merel, Randal O'Reilly, Neil Rabinowitz, Oriol Vinyals, and Greg Wayne.

GROUNDING & LANGUAGE ACQUISITION



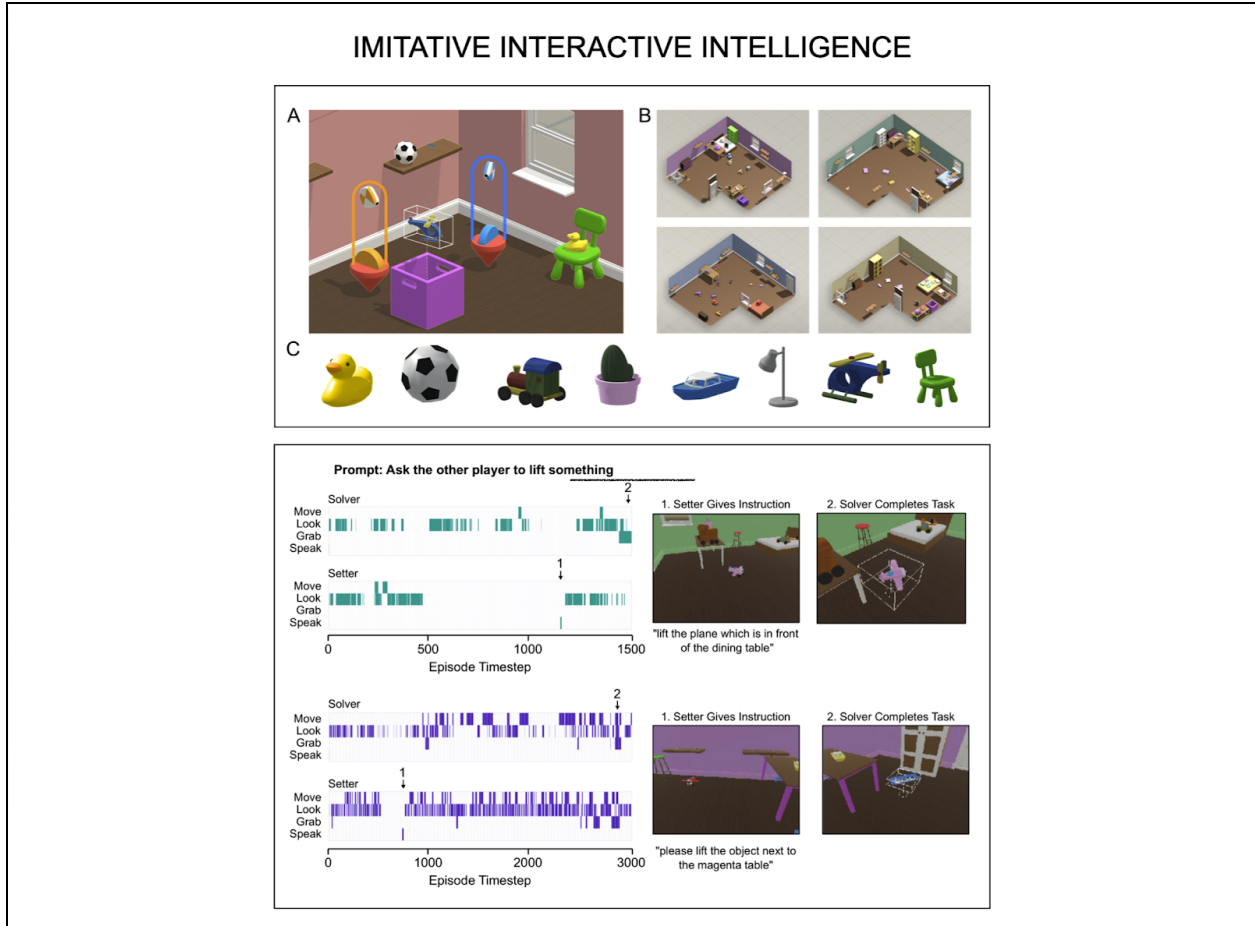
Learning your first language is generally a collaborative effort involving the child, his or her parents, and siblings, and eventually a larger cohort of teachers and playmates. The process of creating a human language is an ongoing collective enterprise with the lasting products of that effort being culture and technology. It's no exaggeration to claim that language is the most important invention of humanity and artificial intelligence hasn't begun to exploit its value.

The words we speak or write belie the complexity of the information they are able to convey. This is because those words tap into the greater context of our collective adaptation and application of those words. The fundamental meaning of those words is not to be found in dictionaries or encyclopedias but rather in our shared experience of the world we all live in.

We say that our use of language is *grounded* in that shared experience, and the foundation of that experience is our protracted development including the multi-year process during which we learned our first language. Eve Clark uses the term [common ground](#) to refer to the larger context established between individuals engaged in conversation, and part of learning language involves learning how to participate and contribute in the construction of common ground.

Human beings are good at constructing narratives to share with one another and in the process of explaining ourselves to others and, importantly, to ourselves. These narratives are essentially

stories that can convey, among other things, recipes, plans and procedural knowledge of all sorts including computer algorithms. Understanding an algorithm often requires an extended context and vocabulary specific to the practice of programming, but it is important to note that the words in that vocabulary still derive from our grounding in the physical world.



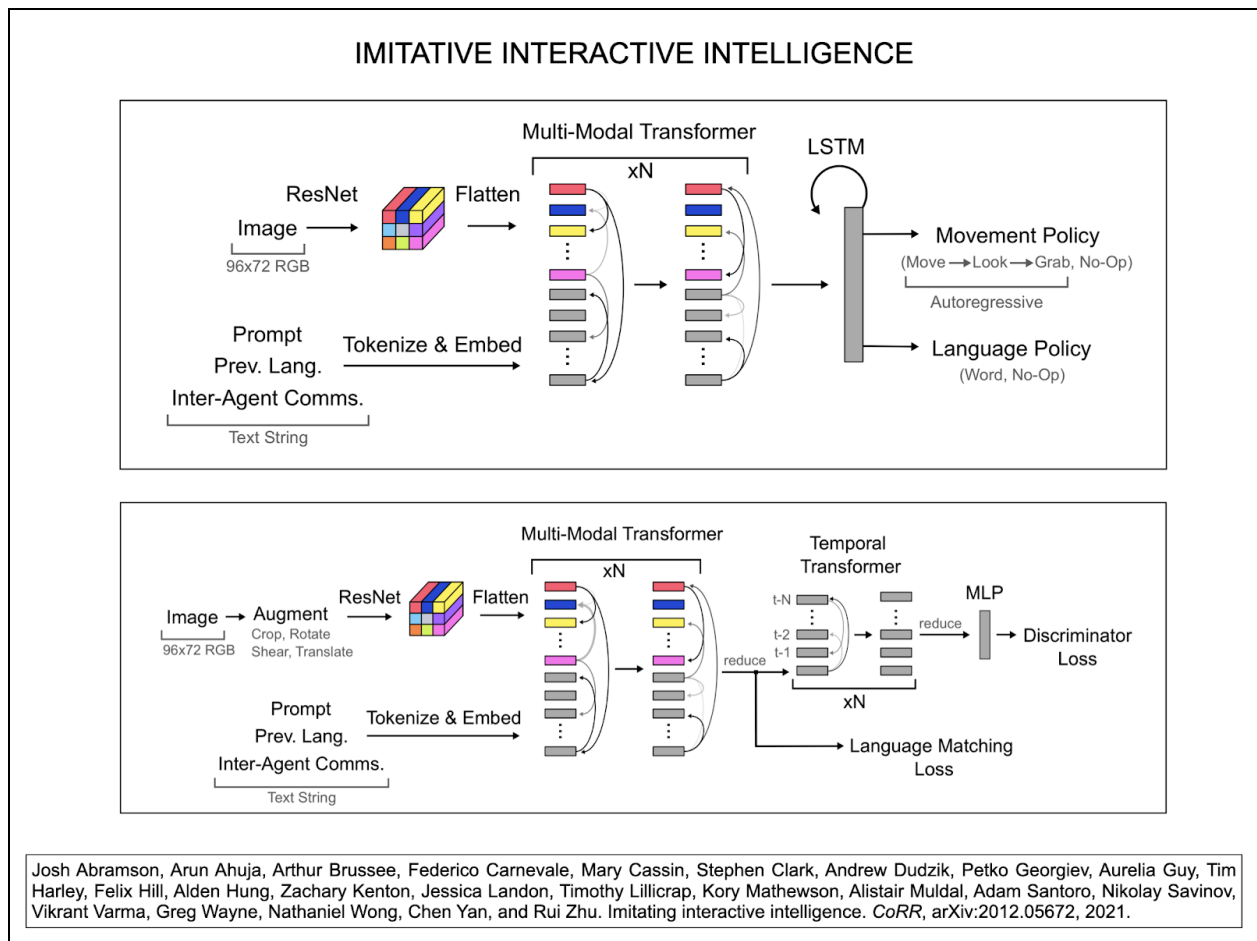
The agents described in the 2020 AIG paper live in a 3D virtual environment referred to as a *playground* consisting of a set of rooms populated by children’s toys and various domestic objects like tables and chairs. The environment also includes a *prompter* that sets the agenda for each learning trial. There are two agents that populate the environment; one of them called the *setter* instructs the other agent called the *solver* to perform a task.

The environment is rich enough to support interactions that involve reasoning about space and the relationships between objects including containment, construction, support, occlusion and partial observability – the last being a fundamental characteristic of the environments that we encounter in the real world and that considerably complicate action and perception.

This simulated environment serves as a testbed for exploring the possibility of building agents that can naturally interact with and usefully assist humans. In particular, it forces us to directly tackle the problem of how to use language as the basis for communication in collaborations, a

problem whose solution we take for granted and as a consequence risk the danger that we take shortcuts in training agents to acquire linguistic skills. Learning a language for the purpose of collaboration requires that the agent ground its understanding of the meaning of words to align with how its collaborators ground those same words – this is essentially what Eve Clark means by common ground.

The process of grounding begins long before an infant utters its first words and is facilitated by a form of *shared attention* in which the infant learns to follow the gaze of its mother to ensure that they are looking at the same thing and eventually learns to employ other modalities such as pointing to establish the relationship between an object and its referent. Grounding is much more complicated than simply learning the names of objects; the child has to learn the naïve physics governing both the static and dynamic relationships between objects and figure out the preferred words used to convey those relationships. This physical understanding of the world – often called *commonsense reasoning*, also serves as the basis for creating and using analogies.



The neural architecture used in implementing the interactive agent is made much simpler by taking advantage of the power and generality of Transformer and ResNets. This relative simplicity is in stark contrast with the complexity and sophistication of the curriculum learning strategy required to train the model. Here are a few key features of the curriculum protocol:

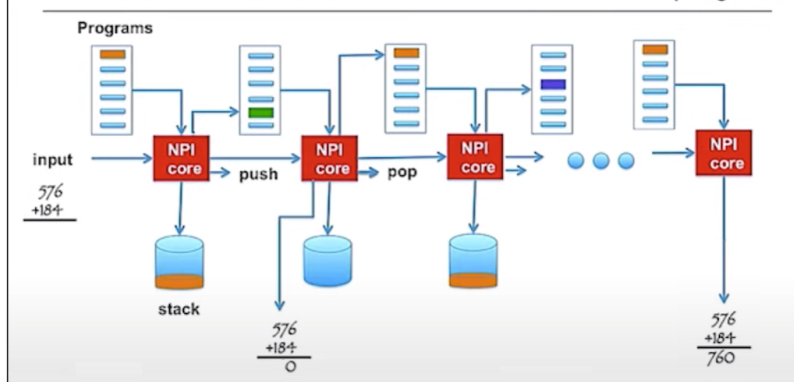
Regardless of the evaluation metric employed by a human evaluator, fundamental properties inherent in the reinforcement learning problem suggest that performance will remain substandard until the agent begins to learn how to behave well in exactly the same distribution of environment states that an intelligent expert is likely to visit. This fact is known as the *performance difference lemma* and the authors' answer to resolving this dilemma is to employ *imitation learning* using a technique known as *behavioral cloning* which frames the problem of copying behavior as a *supervised sequence prediction problem* and requires the collection of a dataset of observation and action sequences produced by expert *demonstrators*.

We would like the agent's policy distribution to match that of the demonstrator's policy, but it's not feasible to train the policy to exert active control in the environment in order to attain states that are probable in the demonstrator's distribution. To deal with this problem they employ *inverse reinforcement learning* (IRL) algorithms in an attempt to infer the reward function underlying the intentions of the demonstrator – intuitively, inferring the *goals* of the demonstrator – and then optimize the policy itself using reinforcement learning to pursue these goals.

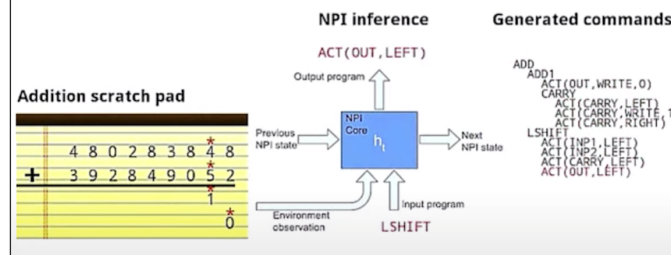
The setting for the agent is *partially observable* and so in the process of evaluating the agent during training we can't assume the agent has knowledge of the current state. To compensate for this they supply the *discriminator* responsible for determining whether the agent succeeds or fails in attempting to carry out a task with a sequence of the last K observations, effectively assuming that the underlying environment is *K-Markov*. These K -length sequences are essentially movies complete with the linguistic and visual information that the agent is privy to and are consequently high-dimensional incurring considerable computational overhead.

NEURAL PROGRAMMER-INTERPRETER

NPI – a net with recursion that learns a finite set of programs



Adding numbers together



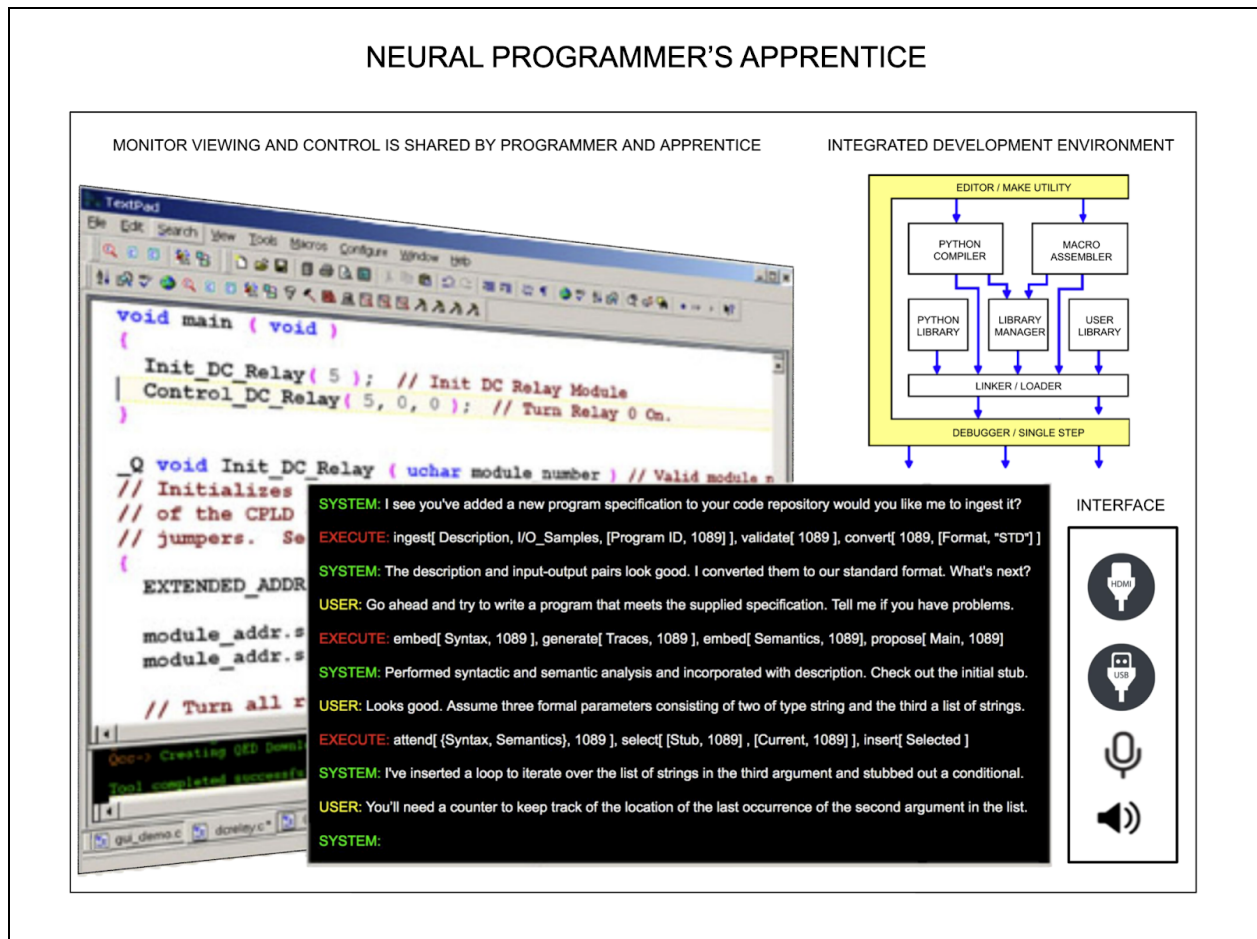
Scott E. Reed and Nando de Freitas. Neural programmer-interpreters. *CoRR*, arXiv:1511.06279, 2015.

A neural programmer-interpreter (NPI) is a recurrent / compositional neural network that learns to represent and execute programs. The NPI is given an environment in which a certain task is to be executed and a library of primitive operations that can be carried out to produce changes in the environment. For example, the environment might consist of a grid *scratchpad* in which each cell can contain a single digit, plus one or more pointers (*) that can be used to identify arguments or the cell in which to write. The task above is to add two multi-digit numbers, and the primitives consist of moving pointers and writing to cells. Borrowing from the paper:

NPI has three learnable components: a task-agnostic recurrent core, a persistent key-value program memory, and domain-specific encoders that enable a single NPI to operate in multiple perceptually diverse environments with distinct affordances. By learning to compose lower-level programs to express higher-level programs, NPI reduces sample complexity and increases generalization ability compared to sequence-to- sequence LSTMs. The program memory allows efficient learning of additional tasks by building on existing programs.

The core module is an LSTM-based sequence model that takes as input a learnable program embedding, program arguments passed on by the calling program, and a feature representation of the environment. The output of the core module is a key indicating what program to call next, arguments for the following program and a flag indicating whether the program should terminate. In addition to the recurrent core, the NPI architecture includes a learnable key-value memory of program embeddings.

The NPI paper was an important advance in neural programming for a number of reasons: it was trained with fully supervised program traces rather than input-output pairs; it produced whole programs as embeddings and stored them for reuse in a key-value memory; it was able to generate high-level programs as compositions of previously learned lower-level programs, and it was able to produce programs in diverse environments with very different affordances. These are all characteristics that we would like to reproduce in the programmer's apprentice.

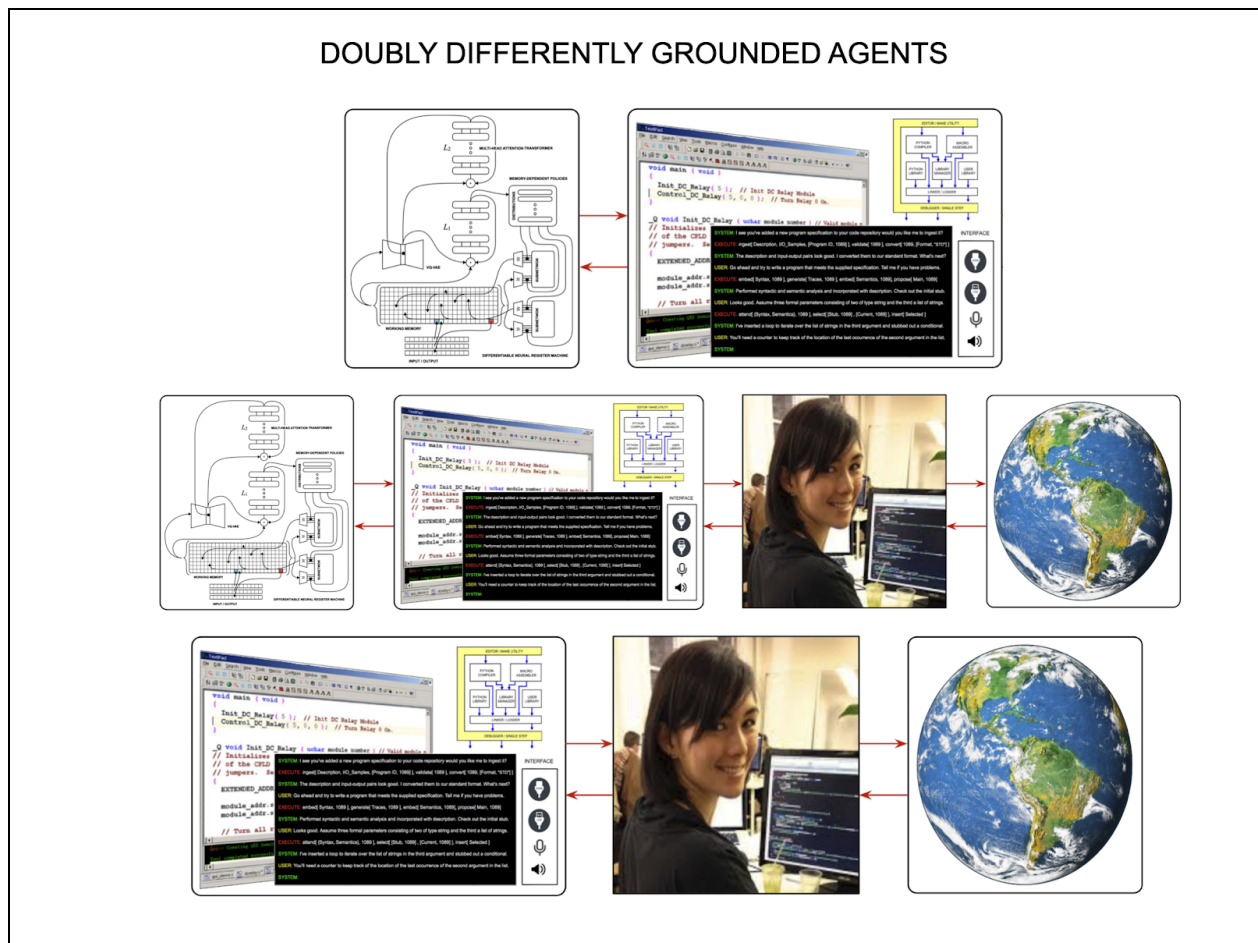


The above graphic depicts the "body" of the programmer's apprentice. Both the programmer and apprentice can view the screen of a monitor connected to a workstation that is running an integrated development environment (IDE) and a lisp interpreter in a shell. Both programmer and assistant can point to locations on the screen and highlight blocks of code in a program listing – this is what Eve Clark refers to as "shared attention". They can communicate with one another by pointing to the screen, speaking, messaging and adding comments to code blocks. Both can enter commands to the interpreter or IDE. The developer interface makes it possible for the apprentice to directly read output from the debugger and other IDE tools as well as ingest programs either as source code or in the form of abstract syntax trees.

It may seem odd or inappropriate to think of an agent as having as its body a computer running an integrated development system, interacting with a python interpreter, and able to use the full

toolchain of compilers, linkers, static code analysis tools like LINT, and runtime debuggers that can single-step through a running program and or trace any function. Since the interactive agent program is also running on the same program, the assistant could in principle run diagnostics on itself and even tweak its weights or modify its objective function.

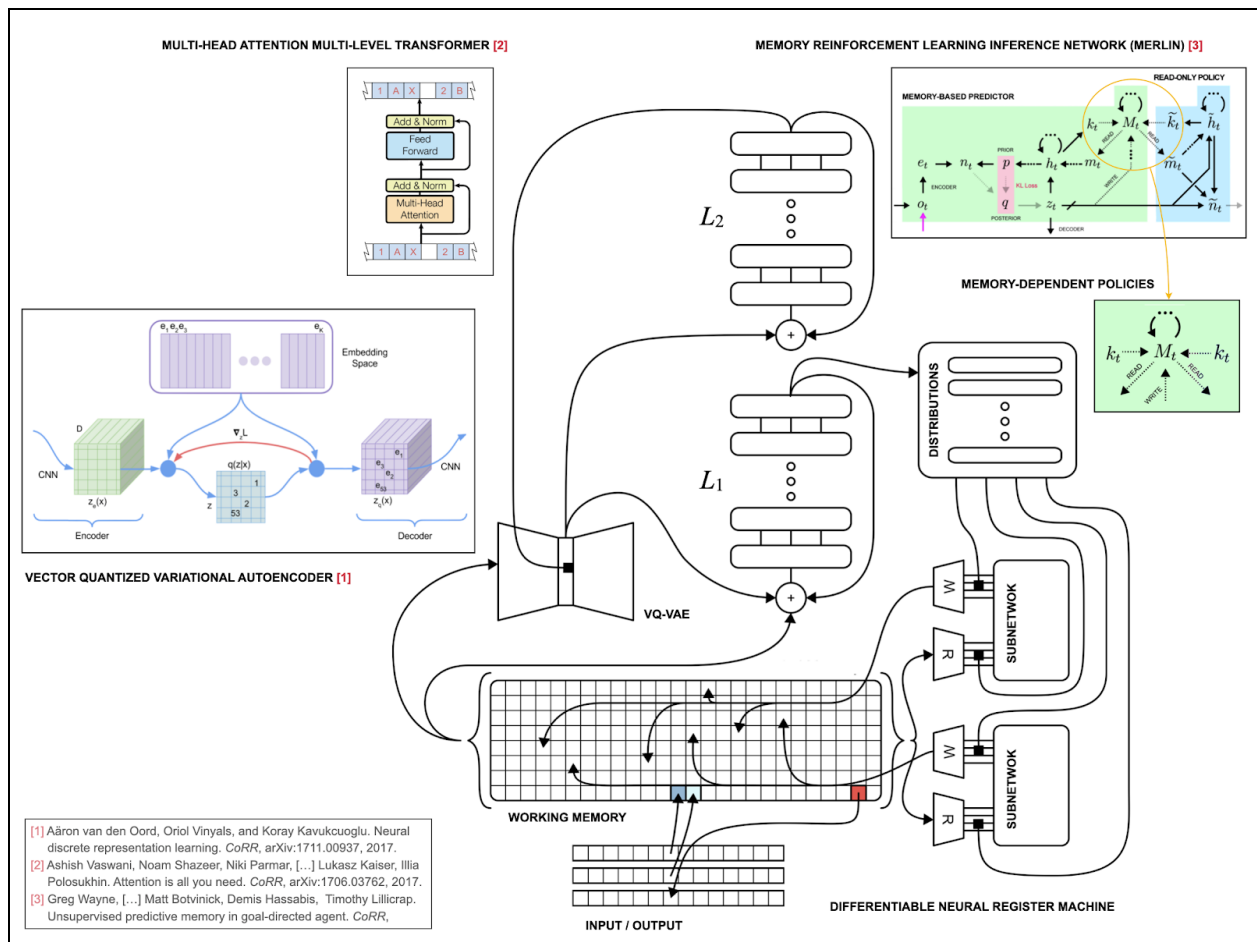
We tend to think of our cognitive apparatus as if it is fully contained within our skull and the rest of the body as a robotic prosthetic controlled by the brain. The fact is that the *central nervous system* (CNS) consisting of the brain and the spinal cord, is highly dependent on the *peripheral nervous system* (PNS) with its pervasive neural circuitry distributed throughout the rest of the body including neuromodulatory and hormonal systems integrated with our skin and internal organs that are as important in mediating cognition as is the CNS. Clearly, the apprentice's experience is different than ours, but there is no reason in principle that experience has to be impoverished any more than a human who has lost a limb or has impaired vision or hearing.



The apprentice agent – depicted here as its neural architecture – is grounded in the hardware and software of the developer environment and its associated suite of software engineering tools as shown in the top row of the above graphic. The programmer is grounded in both the world that she shares with other humans and the world of software engineering tools that she shares with the apprentice as shown in the bottom row. The apprentice doesn't have direct

access to the larger world of the programmer but can share some aspects of the programmer's experience as filtered through the audio and visual channels of its hardware A/V interface.

Think about the implications of this arrangement for the programmer and the apprentice being able to successfully achieve a suitably rich *common* ground. There is a danger that without a thorough grounding in the world of the programmer the apprentice will not be able to exploit the benefits of language that programmers take advantage of, and, in particular, tap into the power of analogical reasoning and algorithmic storytelling that provide programmers with a rich source of insight required to solve novel problems without relying entirely on combinatorial search.



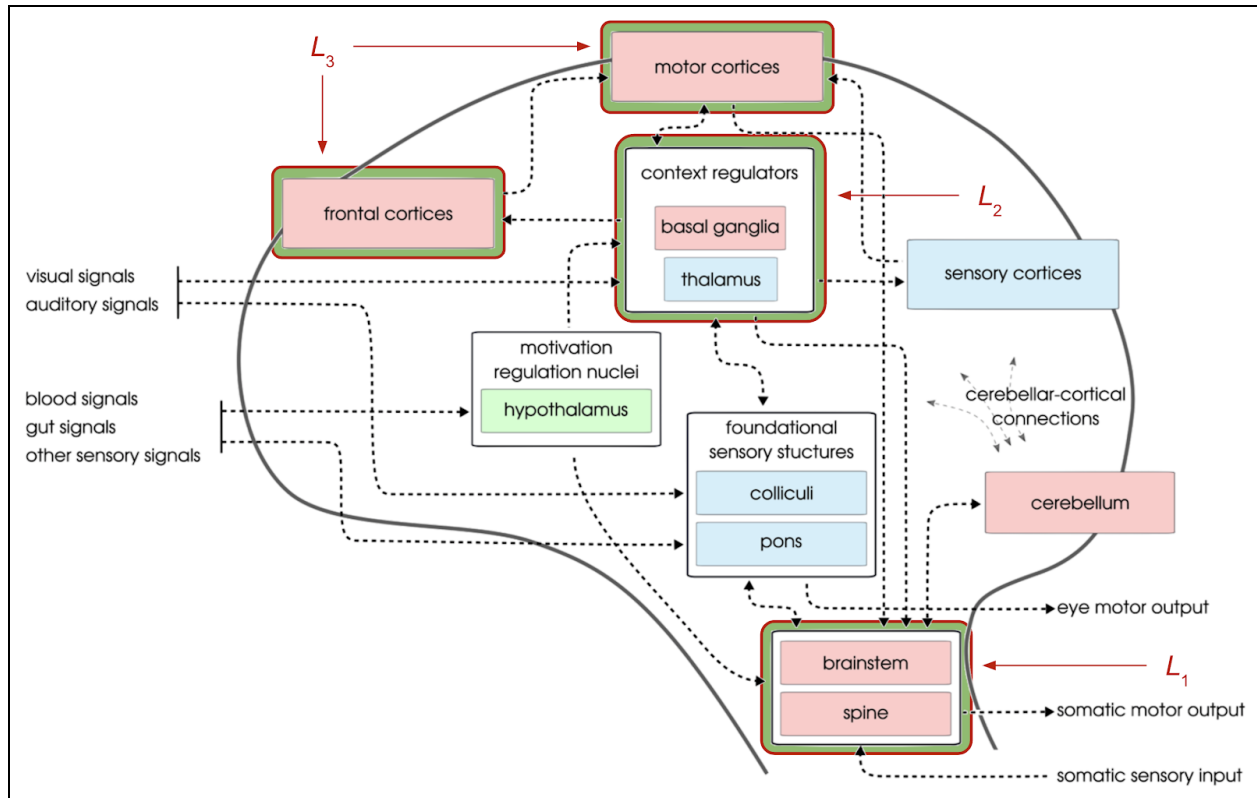
The above graphic illustrates several architectural features of an early instantiation of the neural programming subsystem of the programmer's apprentice:

The network is hierarchical with two levels labeled L_1 and L_2 that support compositionality by enabling the apprentice to write L_2 programs that are composed of L_1 subroutines. Both levels are implemented as transformers and rely on the masking, positional encoding, and multi-level attention mechanisms that have made transformer models so versatile. In particular, the assistant architecture manages a *structured working memory* that can be utilized to implement

data structures and keep track of variable bindings and procedure calls in running programs in the same way that the NPI models use scratchpad memory.

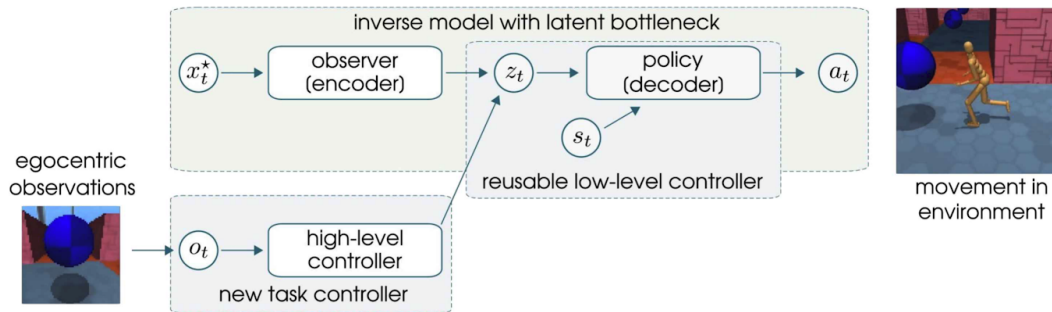
The apprentice architecture also uses a key-value memory for storing contexts in the form of policies and program embeddings borrowing the Reed and de Freitas' [2015] NPI architecture and the Wayne *et al* [2018] MERLIN model. As in the case of Abramson *et al* [2020] and Wayne *et al* the environment is partially observable. Initially, we tried *variational autoencoder* (VAE) predictive models and then *vector-quantized variational autoencoders* (VQ-VAE) in an effort to exploit their ability to model discrete latent representations. Later we figured out how to deal with partial observability using the attentional and positional-encoding features of transformers.

Versions of the assistant architecture focusing on learning hierarchical models that support simulating (interpreting) and writing (synthesizing) novel programs written in a high-level programming language. Yash Savani is one of our consultants for CS379C this year and would be happy to work with student teams interested in pursuing projects along these lines.



In the first of these introductory lectures, we showed a diagram borrowed from Merel et al 2019 *Hierarchical Motor Control in Mammals and Machines*. The above graphic was generated from that diagram to illustrate how the hierarchical levels in the apprentice neural architecture align with our current understanding of the primate brain. We added a third level L_3 to represent the executive level responsible for setting the overall strategy guiding the synthesis of programs.

Neural Probabilistic Motor Primitives for Humanoid Control – Merel *et al* 2019



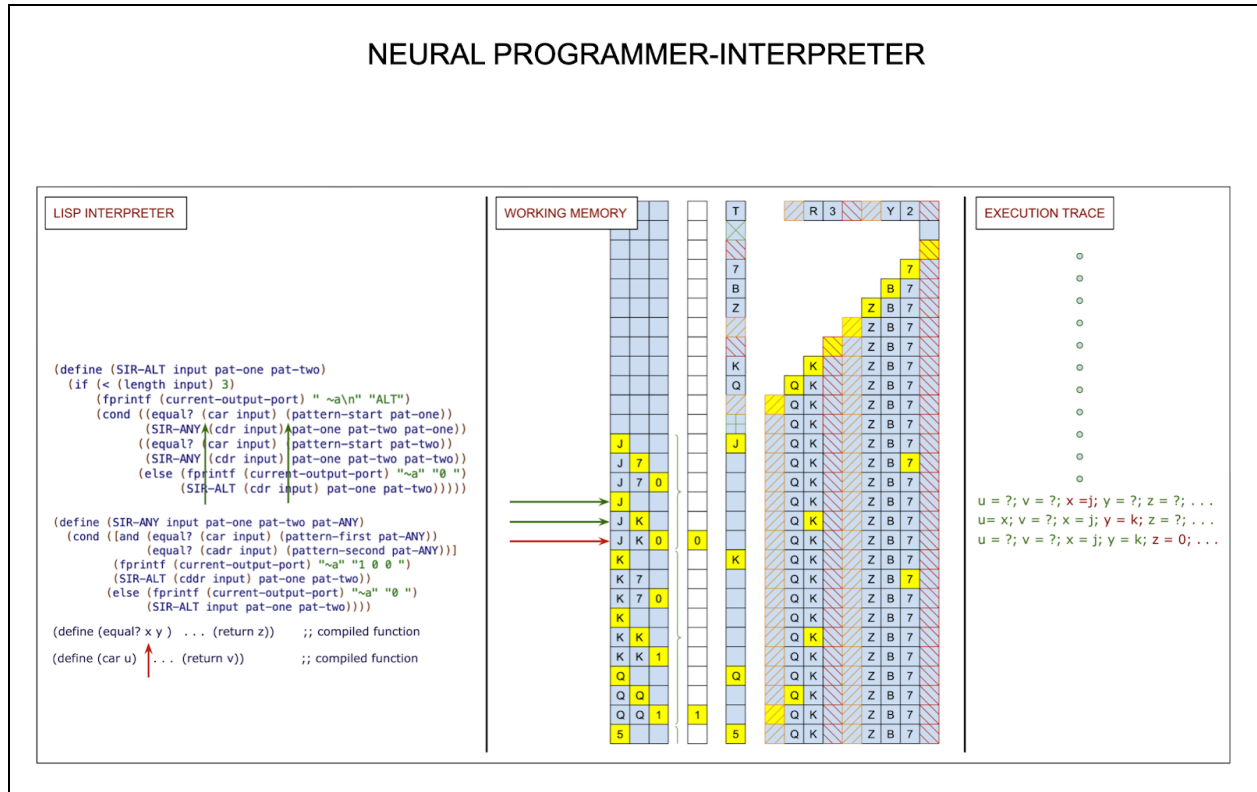
The system is first trained using motion capture data of humans performing movements. The motion capture data are time series of configurations of the body and joints. For each motion capture snippet, a neural network is trained by RL to produce actions, a_t , such that the resulting movement trajectory approximately tracks the kinematic position of the body in the original reference motion. Then, these movement controllers are combined or “distilled” into one large model that can track any of the movements given a description of the near future path of the body, x_t^* . A coding space, z_t , in the system comes to represent each of these movements and allows interpolation among them. Downstream of the coding space is a motor policy, which, when cued with z_t and proprioceptive information s_t , is able to generate patterns of human-like movement autonomously. Thus, exploration of the space of human-like movements becomes possible by varying the input z_t to the motor policy. To this low-level motor system, a high-level controller can be attached to solve complicated tasks in virtual environments. The high-level controller has full visual input and is provided task information, o_t . It learns by RL to produce actions of same size as the coding space, which modulate the movements carried out by the low-level policy. The modular hierarchical design has made it possible to solve complicated problems otherwise of great difficulty for flat RL.

The problem of controlling the motion of a simulated robot is substantially easier than learning to synthesize programs. This is in large part because the environment is forgiving and the required motions are continuous. These two characteristics make it possible to sample the manifold of possible joint configurations densely enough that one can always find a suitable solution by interpolating between known configurations, and, if the resulting movement deviates by a small error from the planned trajectory, it is relatively easy to compensate using a feedback controller.

The successful use of motion capture data as training data for supervised imitation learning serves to bootstrap learning the above-mentioned coding space, z_t , comparable to L_1 in the apprentice architecture. The reusable coding space makes it possible to learn several high-level controllers analogous to L_2 and swap them in and out as required to carry out different tasks. To train the L_1 layer in the apprentice architecture we rely on sequences consisting of consecutive snapshots of the contents of working-memory registers.

This method is analogous to using the sequences of microcode instructions carried out by the arithmetic and logic unit (ALU) of the computer running the IDE process interpreting a program on sample input – which is similar to the imitative learning strategy used by [Reed and de Freitas](#) in the original the neural-programmer-interpreter paper.

The 2019 [Merel et al paper](#) appearing in *Nature Communications* is full of useful insights drawn from both biology and the field of optimal feedback control. In particular, the summary of the key principles of hierarchical control in Table 1 and the accompanying discussion in the main text is worth the time to read carefully. I recommend that, at the very least, you read the highlighted text in the marked-up copy of the paper linked to the underlined text above.



The above diagram features three separate representations of running code. The Lisp code on the left depicts an animation of a program running in the interpreter displaying the current expression being executed. The red arrow indicates the program counter and the two green arrows indicate the interpretable context. The variable assignments on the right illustrate an execution trace for the program running in the Lisp interpreter provided by the IDE debugger operating in single-step mode. Both of these signals originate in the apprentice's "environment". The center graphic represents a sequence of snapshots of the contents of working-memory registers and corresponds to the apprentice's attempt to *interpret* the running program.

Neural Programming, Analogy, and Search

It is one thing to be able to *interpret a program* in a programming language that you are already familiar with and understand the behavior of its primitives, and quite another to *write a program*

in that language given a natural language description of what the program does or a representative set of input-output pairs. Researchers developing neural code synthesis systems often turn to sophisticated search methods with Monte Carlo Tree Search (MCTS) being one popular option [Pierrot et al, 2019].

However, MCTS solutions are only as good as their next-move evaluator/selector. The combinatorial space of programs is huge, and programming by randomly stringing together primitive expressions is unlikely to lead to success for any but the simplest problems. Certainly, some amount of search is unavoidable. At issue is the question of what space of programs to search in and how to explore it. Good programmers easily move back and forth between levels of abstraction to control search and reshape the context. Often they make big leaps and then clean up the mess, essentially making code substitutions followed by program repair.

Industrial strength tools for code completion leverage the power of vector space embeddings of large corpora – millions of lines of code – to *contextualize* search. These tools are particularly useful for projects like input-output drivers and libraries that follow a particular coding style and exhibit recurring patterns. They are not all that useful for implementing novel algorithms.

There are lots of problems that appear on the surface to be challenging, but on reflection can be made simple by exploiting easily extracted latent structure in the input. For example, while neural networks have been trained to write programs that produce drawings of images – a task that may seem quite difficult, but the problem is actually relatively simple and the programs are straightforward – think about why this is the case [Ramesh et al, 2021; Ganin et al, 2018].

In principle, natural language is expressive enough to describe any artifact that human beings have ever engineered. In particular, it is expressive enough to describe any algorithm or procedure intended for conventional computing hardware in enough detail that the algorithm can be translated into any suitably expressive programming language. That said, the specifications for programming problems of the sort encountered in industry and high-quality open-source projects typically assume a good deal of background knowledge on the part of the programmer.

Alternatively, most if not all algorithms can be expressed in terms of commonly available physical objects and familiar manipulations that alter the characteristics of those objects. Such descriptions are essentially *analogies* in the sense that they map the specialized terminology employed in textbooks on programming, e.g., variables, assignments, and declarations, to the informal language of everyday speech, e.g., filling, emptying, or adding to containers.


Assuming that our understanding of such commonplace objects is grounded in the physical world, it seems plausible that we could use our physical intuitions to evaluate, verify and adapt these analogies to serve alternative purposes thereby devising new algorithms or improving on existing ones. To be clear, it's not as though we are thinking about playing with toy blocks as a toddler when writing code to assign values to variables, but rather that our early experience of playing with toy blocks helped to form our earliest representations of the static and dynamic properties of objects and their relationships in the real world, and those early representations

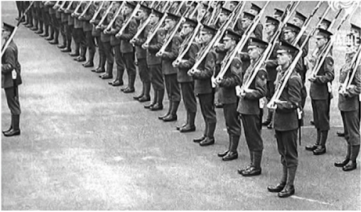
provided the basis – or inductive bias – upon which all of our subsequent refinements and extensions for reasoning about a range of physical and abstract dynamical systems are built.

The claim is that most if not all of our thinking about algorithms is built upon physical intuitions whether we use technical jargon or the language of everyday objects and their physical manipulation. In the following, we refer to these grounded algorithms as *algorithmic stories* and widen their use to include recipes for baking bread, instructions for installing new household appliances, and directions for assembling bicycles and exercise equipment.

Think about how a teacher might use stories and puzzles to teach students algorithmic thinking. Note that logic is critical in software development, and so include exercises using logic to analyze program properties.


Imagine a column of soldiers in a parade or a row of children sitting in desks in a classroom. The participants – soldiers or students – are told to compare their height with the height of the person directly behind them, and change positions if necessary so the shorter of the two is closer to the front of the row or column.


A 

 **B**

Explain to the students that the goal is to arrange the participants so every student has an unobstructed view of the blackboard in the front of the room, or the soldiers an unobstructed view of their drill sergeant. Is this always possible? If not, ask for an example.

What would happen if you carried out this process repeatedly? Would there come a time when repeating the process yet again would result in no changes in the order of the participants? Consider the case in which two adjacent participants disagree about which one of them is tallest.

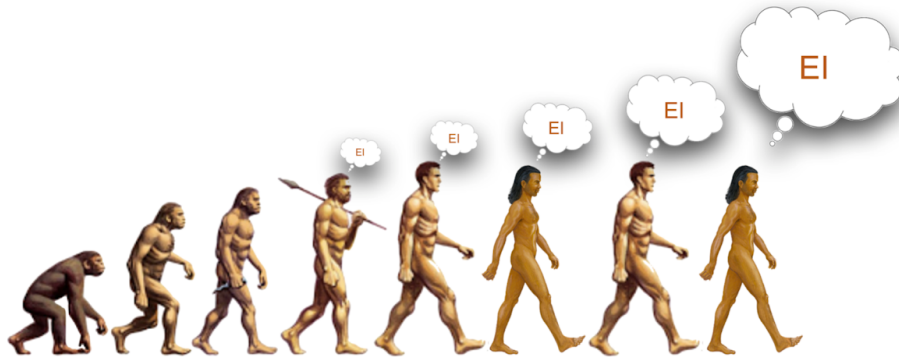
C 

 **D**

It becomes complicated if everyone does this at the same time. Suppose the process is carried out one person at a time starting from the front and progressing to the back. Note that the "next" person in the line is poorly defined. Why is this a problem?

Humans are inveterate storytellers. We think in stories, use stories to remember events, and turn every noteworthy experience into a narrative often embellishing the facts or omitting details to suit our purposes. We often recycle stories by substituting names – both real and fictitious – of people and locations to teach a lesson or serve as a plan of action. The substitutions have to make sense. If they don't we have put mental effort into making them fit the narrative structure [Smith et al, 2017]. Narratives take on a life of their own in the form of fairy tales and parables to teach the young and indoctrinate members of secret societies and religious sects.

Human Language is Grounded[†] in Our Experience of the World We Share



Much of the knowledge required to exploit extelligence is hidden in our experience of the world we share with other humans – we say it is grounded in the physical world. This hidden knowledge is implicit in the stories we tell our children because we can depend on our children having much the same experiences that we have had. It forms the basis of what we refer to as *common sense reasoning*. It plays a crucial role in making analogies because it is this knowledge that enables us to identify and anticipate problems with a proposed analogical mapping between stories and consider the possibility of relaxing or eliminating some of the constraints thereby generating new analogies.

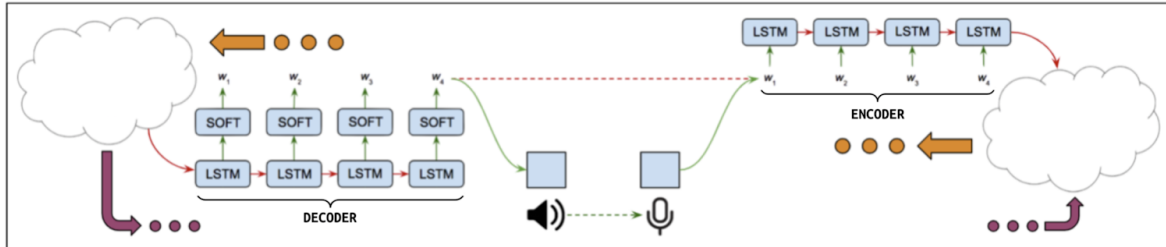
[†] I once believed any agent grounded in a suitably rich environment could learn to communicate with human language and take advantage of the knowledge available in human culture. I no longer think this is true. Words have meaning only in the context of other words. Facility with human extelligence would seem to require a remarkable alignment with the entities and processes that form the basis for grounding human experience.

Extelligence and Common Sense Reasoning

Extelligence is defined as the cultural capital available to humans through language in all its forms including written, spoken, and signed [Stewart and Cohen 1997; Deacon 1998]. It is the knowledge that exists external to our brains, but that we can access by virtue of our ability to understand natural language. Much of the knowledge required to exploit extelligence is hidden in our experience of the world that we share with other humans – we say it is *grounded* in the physical world.

This hidden knowledge is implicit in the stories we tell our children because we can depend on our children having much the same experiences that we have had. It forms the basis of what we refer to as *common sense reasoning*, and it plays a crucial role in making analogies because it is this knowledge that enables us to identify and anticipate problems with a proposed analogical mapping between stories and consider the possibility of relaxing or eliminating some of the constraints thereby generating new analogies [Gentner, 1983].

Phonological Loop, Internal Monologue & Hofstadter's Strange Loop

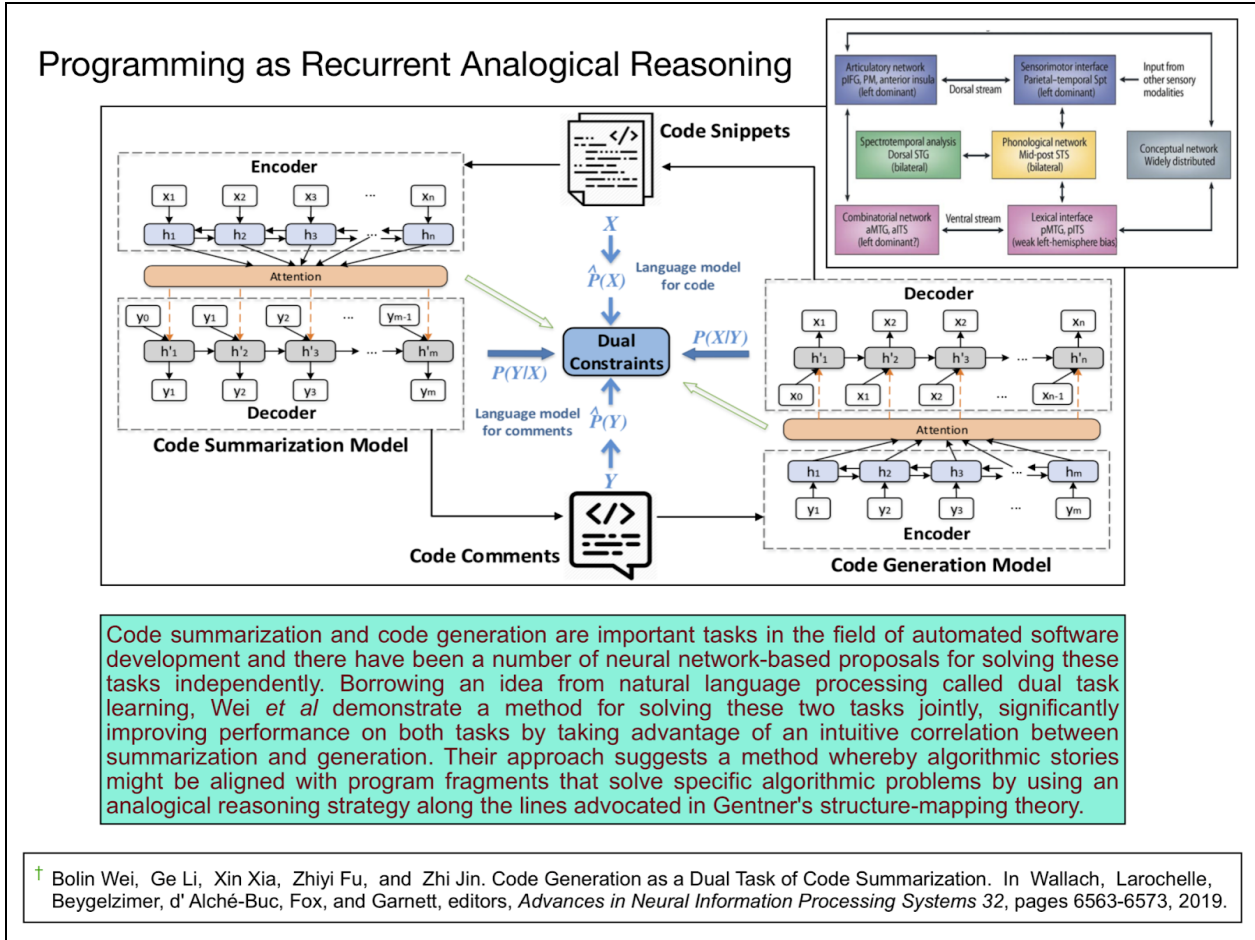


Consider inner speech as a constructive process in which an agent – call her Alice – hears a story and generates an internal representation of the story from her subjective point of view. Essentially Alice encodes the story using a combination of her interpretation of the words in the story as she read it and the context in which she heard it. By context we mean the perceptual, conceptual and procedural information encoded in working memory and corresponding to Alice's current "state of mind".

This process is recurrent and, in our case, goal driven. Alice then retells the story, decoding her internal representation of the story in her own words and in keeping with the new context informed by having ingested the original story. The process continues with the words of Alice's decoding now serving in the role of the original story and Alice hearing herself read the decoded story. Note Alice need not actually hear or speak at all – via the dashed red line. She could have started out with one of her own stories.

In this example, we assume Alice has the goal of understanding the story in order to apply it to solving a particular problem. In terms of using analogy as an aid in writing and understanding programs, we see this process as one of searching episodic memory for an *algorithmic story* starting with a story corresponding to a description of the current program or program fragment to be written or understood.

The recognition of some sort of inner monologue as a core component in cognition is often credited to Lev Vygotsky – *Thinking and Speech* [1934]. There are many hypotheses concerning the precise function of this phenomenon. Hofstadter [2007] describes it as a "sequence of mental shifts between levels in a hierarchy that eventually ends up back where it started. Each shift feels like an upwards movement, and yet the successive 'upward' shifts give rise to a closed cycle". The above graphic provides an alternative characterization couched in terms of neural networks representing embedding spaces and encoder-decoder architectures including transformers [Vaswani *et al*, 2017] for transforming one representation into another.



The above graphic depicts a neural network architecture [Wei et al, 2019] that borrows an idea from NLP called *dual-task learning* [Wallach et al, 2019] – where "dual" is used in the sense that it is applied in mathematical optimization theory – that treats code generation as the dual of code documentation. In a similar sense, the reciprocal sensory-motor connections in Fuster's hierarchy consist of a forward (primal) model and its corresponding (dual) inverse.

What if we were to implement a variant of their code-generation encoder-decoder unit as a forward model complimenting my inverse-inner speech-model in the previous slide and then couple the two with a regularizing loss function as in [Wei et al, 2019]? As a simple thought experiment, assume that both the forward and inverse models involve a latent-variable (informational) bottleneck layer [Tishby et al, 2015]. What might serve as the analog of their dual-constraints module integrating the language model for code and the language model for comments and what exactly would be their role in the loss function?

Cognitive Workflow and Interaction Networks

What is the "state" of a *multi-layer perceptron* (MLP)? For a specific input, an MLP produces a specific output and that is pretty much it. What about a convolutional network (CNN)? CNN's utilize a kernel specifying a bank of filters of a given shape and overlap but otherwise are not significantly different from MLPs. How about a recurrent neural network (RNN)?

Mathematically an RNN is an instantiation of a differential equation that can be employed to produce a discrete-time series or continuous trajectory through the phase space for the dynamical system described by the equation. Unless explicitly introduced, there is no notion of time aside from our interpretation of the points along that trajectory.

At different points, some variables change quickly while others slowly. In the case of artificial neural networks, these changes are due to special circuits that control the rate of change in other subcircuits of the network. For example, these special circuits can be used to control the timing of activity, persistently maintain state and isolate specific subcircuits from change.

If we think of the electrical potentials in biological neural networks as represented by differential equations as in the case of the Hodgkin and Huxley model [1952], we would still have to introduce ancillary functions to explain the sort of large scale coordinated changes in neural activity we observe in biological networks as a result of diffuse signaling by neurotransmitters and the patterns of activity orchestrated by specific frequencies of spiking neurons.

It is convenient for us to think of the latter as controlling the topology of the network architecture by periodically taking some component networks offline to protect them from alteration, but this approach runs counter to our interest in building fully differentiable systems that can be trained end-to-end at scale using efficient techniques like stochastic gradient descent.

The problem of catastrophic interference/forgetting in transfer learning is one example of how error propagation can result in unwanted changes to the weights of subnetworks that we would prefer to be treated as essentially offline. Biological networks in which only a small fraction of neurons are active at any given time have, with few exceptions, no such problems.

One way or another, we will have to incorporate strategies for avoiding catastrophic forgetting in the case of the workflow of an interactive agent like the programmer's apprentice and especially in the case of the assistant constantly having to integrate new information into evolving neural network architecture. New words are acquired rapidly by a child in the process of acquiring language so that knowledge concerning their use and evolving semantics can be acquired with little or no conscious effort in the midst of normal conversation.

Representing and Reasoning about Analogies

My advisor at Yale, Drew McDermott, was famous among AI researchers in the '80s and '90s for his paper with the pithy title "Tarskian Semantics or No Representation Without Denotation". His main point was to discourage AI researchers from giving predicates names like PLAN and ACT in their predicate calculus theories of common-sense reasoning as it was misleading.

Distributed representations in biological and artificial neural networks encode meaning contextually. Entities that co-occur, whether words in language or entities in the physical world are likely to be related in meaningful ways, and to fully understand those relationships one has to explain their composite behavior.

The state of a dynamical system and its dynamics can be modeled as a graph and represented as a vector in a high dimensional vector in an embedding space. Two such embedding vectors could be used to highlight the differences between two dynamical systems. Transformer networks can be trained to simulate the behavior of dynamical systems so as to compare the properties of two separate systems.

We suggest here that these models could be used to assist in learning new models and a collection of such models could serve as the basis for making analogies that might facilitate applying existing knowledge of one dynamical system to understanding or modifying another.

Indeed we suggest that analogies – in a restricted sense – are essentially mappings between dynamical system models and (successfully) making an analogy is tantamount to identifying such a mapping. This isn't a new idea. There is a long history of work along similar lines. What is missing, however, is how the brain carries out this process of analysis and how might we model that in an artificial neural network – see Hill *et al*, 2019 [[PDF](#)] for a recent alternative.

If pressed to suggest a network architecture for analogical reasoning, I would start with what Battaglia and colleagues refer to as *interaction networks* [Battaglia et al, 2016]. These models are designed to represent and enable the simulation of the objects, relationships between objects, and the physics that govern their dynamics. These representations could be stored in a library of such models implemented as a high-dimensional embedding space – see Hill *et al*, 2020 [[PDF](#)] for a discussion of why this might *not* be the best approach.

Computer programs at different levels of abstraction might be modeled as interaction networks, and the embedding space for programs could be employed to find similar models. It would also be useful to be able to map descriptions/specifications of programs to their corresponding embeddings. Then, assuming that descriptions are essentially analogical stories, ...

Let's Get Real! What Possible Related Class Projects Might We Pursue?

Stories, and algorithmic stories, in particular, have several characteristics in common that make them challenging to work with. Both are discursive in the sense that they jump around in ways that can be difficult to follow. In a computer program, the result of evaluating an expression may be simple if you are familiar with the relevant procedure, but more complicated if the procedure is specialized to the program that the expression invokes – which is often found elsewhere in the program listing, or if the expression describes a conditional branch.

In telling the story, it is not uncommon to interject a comment about some character in the story or even relate a side story that provides a context for understanding some aspect of the primary story being conveyed. In a computer program, if we ignore the comments, for the most part, we can follow the steps carried out by the program for a given input, generating a trace of the program running on a given input. However, the set of all such traces is likely to be exponential in the number of branch points, and hence generating an exhaustive set of traces is intractable.

For the most part, we have no trouble following someone's telling the story, though we may find it difficult to repeat the story to someone else without butchering the details or forgetting to include the side stories without which the behavior of the characters in the primary story may be incomprehensible. A well-told story is much like a well-written computer program with the main program documented, individual procedures given names that suggest their function and listed separately, and any libraries used documented clearly the beginning of the current.

Young children begin generating stories very early in their development, starting with a running commentary describing what they are doing and what they see others doing that is an important part of their interactions with their parents and other early interlocutors who seem programmed to expect such commentary and correct the child thereby providing feedback for self-initiated repairs [Clark, 2020]. If we are to expect the apprentice to learn to apply algorithmic stories, we will have to train the apprentice to first construct rudimentary algorithmic stories for more mundane purposes like riding a bike and then apply them to producing computer programs.

The literature on graph networks offers a reasonable place to start and the bAbI project of Facebook AI Research which is organized towards the goal of automatic text understanding and reasoning provides some ideas about possible datasets. Check out Daniel Johnson's 2017 ICLR paper for some early work on using graph networks to solve bAbI tasks. Taking a complex problem like the one addressed in the IAG paper and turning it into a tractable project that provides insight into possible solutions to the original problem is an art that every research scientist has to master in order to be successful. Like any art worth pursuing it takes practice.

I've found in both my academic and industrial experience, the best approach is to brainstorm with colleagues from multiple disciplines, with differing attitudes regarding the theory versus practice divide, who are comfortable exposing their half-baked ideas, and can suspend belief long enough to entertain and even help to fix the bugs in their colleague's ideas. It helps to assign someone the task of remaining impartial enough to referee disputes, encourage more reticent colleagues to speak up, and suggest taking a break or reconvening the next day.

Miscellaneous Loose Ends: Google offices have *micro kitchens* distributed all over their campuses. They are equipped with barista-quality espresso machines, fancy tea-making paraphernalia, the makings for just about any caffeine beverage, seasonal fruit, and regional specialties like English high tea, fresh-baked croissants, and scones. I've overheard casual conversations, carried out while the participants meticulously prepared their tea and espresso drinks, during which new algorithms, neural network architectures, infrastructure designs, etc., were conceived of or improved upon. Some visitors are appalled by the show of expensive kitchen equipment and gourmet food, but micro kitchens are incredibly efficient engines for generating technology and creating a rich culture of sharing ideas and working collaboratively.

Analogical Reasoning and Grounding Revisited

The following is provided, as are many of the comments in this document, as a thought exercise intended to provoke discussion. The exercise is an attempt to provide some rigor in describing the computational processes involved in creating and sharing analogies. The diagram shown in Figure 1 lays out the entities involved in these processes and the interactions between them. The boxes labeled *physical laws* and *environments* will be assumed to be obvious at least at least for the purposes of our analysis. The two pairs of identically coupled blocks enclosed in dashed red lines correspond to two distinct agents attempting to communicate with one another.

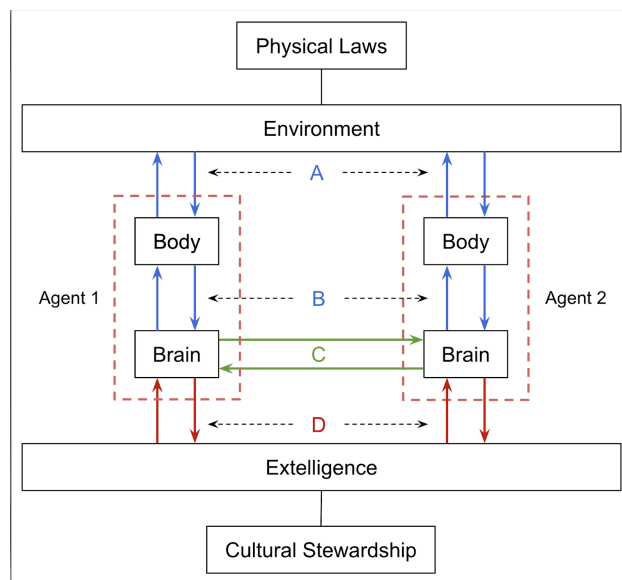


Figure 1: The above graphic illustrates the main components involved and the flow of information between them in a model of how grounding language in our interaction with the physical world naturally gives rise to storytelling and analogical reasoning, provides an impetus for societies to create technology to cultivate, disseminate and preserve useful stories and to encourage exploiting their application as a collective means for accelerating innovation in the social and physical sciences, and engineering disciplines. See the main text for a more detailed explanation.

In each case, the separation between the block labeled *body* and the block labeled *brain* is intended to convey the idea that – the substantial changes wrought during development notwithstanding, the agent's physical contrivance responsible for interacting with its environment changes at a much slower rate than the neural substrates responsible for processing the agents observations of its environment and its communication with other agents.

In each case, the separation between the block labeled *body* and the block labeled *brain* is intended to convey the idea that – the substantial changes wrought during development notwithstanding, the agent's physical contrivance responsible for interacting with its environment changes at a much slower rate than the neural substrates responsible for processing the agents observations of its environment and its communication with other agents.

The block labeled *extelligence*¹ is intended to represent the information available to human beings in the form of external resources through various technologies curated, managed and distributed by individuals, libraries and universities funded by governments and privately owned institutions. It is the cultural capital that is available to us in the form of external media, including tribal legends, folklore, nursery rhymes, books, film, videos, etc [82], and it is assumed for the present discussion that individuals can effortlessly tap into these resources and contribute to them by adding additional content or editing existing content.

The reciprocal connections labeled **A** are intended to represent the sensory and motor activities that precipitate changes in the physical substrate providing the-base level interface for interacting with the environment and other agents. We will assume that development of this substrate has advanced to a stage of maturity at which it will remain static for the remainder of the agent's life. We do not assume however that this substrate is necessarily identical for all agents, indeed we expect that the influences of nature and nurture to ensure there are substantial differences across individuals.

The reciprocal connections labeled **B** correspond to the actions performed and the observations made by the agent in exploring its environment and engaging with and learning from other agents. These actions and observations are precipitated and guided by plans and goals that the agent has learned has acquired in its exploration of its environment and, in particular, its exposure to the customs, norms and behaviors prevalent within its social milieu. We assume that agents routinely formulate what we will refer to generically in the sequel as "models" for the

purpose of making predictions concerning the consequences of its actions, and that furthermore agents routinely share those models.

The reciprocal connections labeled **C** correspond to communication channels that serve to facilitate the sharing of models. We assume for simplicity that all the agents use the same finite lexicon of words to communicate, but the meaning of those words can differ depending on the context of their usage, where that context extends to include their general cohort as well as their immediate interlocutors. Since individuals are unlikely to have exactly the same grounding, this implies another factor that can contribute to variability in the interpretation of words and cause the negative consequences of misunderstanding as well as the positive consequences of inventive reinterpretation.

Finally, the reciprocal connections labeled **D** are intended to represent the channels whereby an agent can contribute a model to the extelligence repository thereby making it available to other agents outside the circle of the contributing agent's cohort, or gain access to an existing model allowing them, in principle, to use any model previously added to the repository and successfully preserved. We will say that a model is *correct* if it makes accurate predictions when applied to systems of the sort that it was originally designed to handle and the words used to define the model are assigned the meaning intended by the contributor.

We assume for simplicity that models are correct, as contributed, if the words used to describe them are assigned their original intended meaning. This does not protect against an agent misapplying a model because they assign different meanings to the words used to formulate the model. It also begs the question of what constitutes the meaning of a word. To resolve these issues, we need to be more careful in defining models and assigning meaning to the words used to describe them.

We assume for simplicity that all models are represented as dynamical systems corresponding to a set of objects, a set of relationships involving those objects and a system of equations or alternative dynamical system models governing their kinematic and dynamic properties. What would it mean to ground such a model? Following the lead of cognitive psychologists like Liz Spelke and her students and collaborators, we might look for signs of grounding in the behavior of infants and young children.

In the case of a model corresponding to the physics governing the trajectory of a thrown ball, evidence of grounding might appear in the observed behavior of a child pantomiming the trajectory by tracing it out in the air. In the case of demonstrating the consequences of a person walking at a steady pace and temporarily disappearing behind an occluding screen, evidence of grounding might be indicated by the child registering surprise when the person doesn't reappear on the other side of the occlusion after a reasonable delay.

When we are talking about modeling physical movements, demonstrations of reproducing familiar movement patterns combined with extrapolating from those movements to predict novel patterns would seem to provide evidence of physical grounding. There is a growing body of research attempting to identify the neural correlates of grounding and naïve physical reasoning,

e.g., see the work of Bunge *et al* [18] examining hypothesized role of the anterior left inferior prefrontal cortex, but there are also predictions we might make based on the known architecture and areal function of the primate brain [90, 45, 38, 18]. See Figure 2 for a summary of experimental results from [90].

If we assume the same sort of hierarchical architecture we find in the sensory-motor cortex consisting of multiple layers of increasingly abstract, multi-modal features built on a foundation of primary unimodal features, we might expect the architecture for representing a hierarchy of models to be organized similarly [33]. For example, in observing the brain of a physicist thinking about a highly abstract model of quantum tunneling in semiconductors, we might observe activity in areas that become active upon observing someone walking along the sidewalk and temporarily disappearing from view for a few seconds upon walking behind an occluding building-construction barrier and subsequently reappearing on the other side.

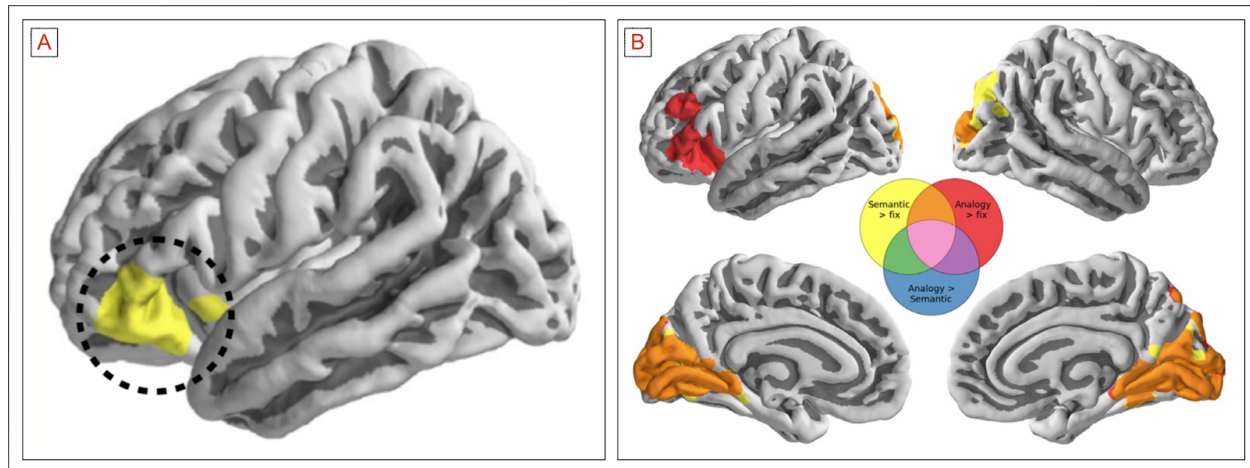


Figure 2: The graphic (A) on the left shows regions that demonstrate an increase in activation with accuracy on analogy trials across all participants after correcting for the effects of age. Only the contrast of semantic greater than fixation (shown in yellow) showed a significant correlation in left aLIPC (anterior left inferior prefrontal cortex) – see Bunge *et al* [18]. The graphic (B) on the right shows regions that indicate an increase in activation with age across all participants. Results for the semantic tasks greater than fixation contrast are shown in yellow, analogy tasks greater than fixation in red, and regions for which both are increasing are shown in orange. There were no regions that showed a within-person differential increase in activation during analogy trials compared to semantic trials. The graphics are from Figures 4 and 5 in [90].

Similarly the meaning of "the excavator shovel scooped up a huge pile of gravel and deposited it in the back of the dump truck" might employ – be grounded in – models that we learned while still in diapers playing with toys in the sandbox or even earlier when our parents would try to feed us our pureed peas with a spoon while we squirmed in our highchair².

These early experiences would ostensibly provide the seed for a model and naturally determine its locus for our subsequent integration of related but more complicated interactions. These subsequent interactions would attach themselves as new representations that are defined by the observable characteristics that both identify and distinguish them from the seed representation.

Over time models suggested by related experience would tend to encroach into areas devoted to representing other more abstract models, but, whenever possible, phenomena described using familiar terms like "scoop" and "throw" employed to represent abstract properties of behaviors that share some abstract characteristics with the physical acts of scooping and throwing would strengthen the connections to the original seeds planted in those early experiences.

The use of the word "locus" might appear to allude to some unidentified unitary neural substrate, but that was certainly not intended as it seems much more likely that the circuits implicated in analogical reasoning are widely distributed throughout the brain and the cortex in particular – see the references cited in the earlier discussion in this entry and in [Figure 2](#) and the [lecture](#) by Silvia Bunge.

These abstract variants of primal activity are useful insofar as they simplify the description of more abstract models by not requiring the invention of additional, less widely accepted terminology, i.e., jargon, and by doing so they make it easier to succinctly and clearly communicate abstract concepts with others insofar as they can convey the gist of what's involved in the abstract process without introducing impediments in the form of related usage norms that might contradict properties of the abstract process.

In a mature brain, experts may work primarily with the abstract models using specially-crafted cognitive tools designed to deal with the specific characteristics and properties of the abstract models. However, in a more exploratory mode – for example, struggling for intuition concerning some complex abstract mathematical object, we may resort back to the primitive, grounded levels of representation in searching for relevant analogies that might map onto the problem currently distracting us.

Grounding in Synthetic Environments

If we think of the modern world that we live in as extended to include the informational space of the extelligence, we might ask if there are evolutionary pressures on the ideas that reside in that space. One possible candidate would be the influence of modern scientific and engineering practice and we might conclude that the way in which the scientific and engineering communities judge the products of their respective disciplines serves to impose selective pressures to increase the probability that scientific theories and engineering artifacts stand up to intense scrutiny.

Combine this relentless scrutiny with the fact that there are billions of human beings many of whom have the intellect and education to propose a credible theory or invent a useful artifact, and you have the makings of a powerful engine for generating novel theories and artifacts. Diversity arises from the fact that differences in our grounding of language cause us to interpret what we've been calling *algorithmic stories* in subtly different ways that nonetheless adhere to a consistent set of rules that potentially could lead to new and novel ideas.

In addition to our natural talent in searching for novel ideas grounded in our experience of the world we live in, scientists and engineers are now experimenting with technologies that enable us to extend the perceptual and interactive capabilities we were born with, allowing us to broaden our experience – and hence grounding – to include physical processes that involve forces acting on arrangements of matter that we can't directly observe and that take place across a wider range of scales than than we can directly observe.

Such efforts include the gamification of protein folding ([Foldit](#)) and neural circuit reconstruction ([Flywire](#)). The simulators and user interfaces for these online games enable citizen-scientist volunteers to interact with and explore physical systems that would otherwise be inaccessible and inscrutable.

Sophisticated immersive interfaces for powerful simulators allow scientists to explore the universe at both [quantum mechanical](#) and [cosmological](#) scales. Perhaps more relevant in terms of grounding leading us to come up with new ideas is the ability of these simulators to explore worlds in which the fundamental constants and governing equations are different than the scientific consensus would lead us to believe.

The hypothesis we are entertaining here concerns whether the human use of analogy in the process of exploring / searching the space of possible theories for explaining natural phenomena, designs for engineered artifacts, policies for inducing social change, etc., is efficient when compared to traditional search methods such as Monte Carlo search. Of course it needn't be exclusively one or the other. Monte Carlo search requires a next move generator and clearly some form of analogical reasoning could potentially serve in that role.

In many cases including the use of MCTS (*Monte Carlo Tree Search*) in [AlphaZero](#), the next move generator is learned and one could imagine having different pretrained move generators for different search problems. Humans effectively have different move generators even though they employ the same neural structures to encode them and apparently are able to avoid catastrophic interference when training them. See Carstensen and Frank [21] for a discussion of neural network architectures that can learn graded – continuous versus symbolic – relational functions and the role of graded relational functions representations in explaining abstract thinking in primates.

Miscellaneous Loose Ends: The programmer's apprentice requires the programmer and the apprentice to establish a shared ground for both natural language and the programming

language they use for writing software, and the requirement holds despite the fact the apprentice can only vicariously experience the context in which the programmer learned language. This arrangement is worth more consideration in this discussion list, but, from the practical standpoint of selecting and carrying out a final project involving neural programming, we might dispense with natural language altogether, reduce the role of the programmer to simply a source of models in the form of programs that can be employed in analogical reasoning and incorporate this limited role directly in the apprentice's curriculum training protocol.

In our discussion on Sunday we talked about the problem of keeping track of where in memory we store items that we have to retain for indefinite periods of time. Specifically, we talked about the problem of keeping track of variable bindings and [namespaces](#), contexts and closures, recurrent calls to the same procedure or calling a different procedure that uses the same variable names as the calling procedure. All of these problems are solved in modern programming languages by using a *call stack*. In earlier work, we considered the possibility of using a differentiable external memory [44] and differentiable programs procedures to implement a call stack. The [MEMO](#) system [8] developed by researchers at DeepMind appears to provide an alternative solution that is more generally useful³.

What does it mean for something to be [grounded](#)? In some contexts, it means that you are sensible and reasonable – your feet solidly on the ground. In other contexts, it might mean that you are knowledgeable about the basics of something. It is often used in everyday speech to indicate that something said or done can be relied upon. In the sense that scientists and engineers generally use the word when talking about language – and, in particular from the perspective of machine learning and embodied cognition, it means that your understanding of a word is based upon your direct experience of the referent of the spoken, written or signed word.

Note that experiential grounding is not just about language. Grounding is key to understanding all forms of meaning and biological communication. No matter what situations you are being exposed to, your brain is anchoring you to the experience of those situations and those experiences will shape how you perceive and interpret the world around you. For example, the word "running" might be represented in a neural network as a pattern of activity associated with a particular subnetwork when the word is spoken, read or a physical instance of running is observed or recalled. Donald Hebb referred to these patterns and their corresponding neural circuits as [cell assemblies](#).

Take a moment to think about what it might mean for the programmer's apprentice to ground its experience of instances of the expression (if (equal? X Y) (set! X (+ X 1)) else (set! Y to (- Y 1))) where X and Y are variables assumed to vary? What about experiences of X where X always appears in this context bound to an integer?

Hebb suggested that cell assemblies are formed when such patterns occur repeatedly, as evidenced by the units that comprise such an assembly becoming increasingly strongly inter-associated. Unlike the sort of learning employed in most contemporary research on neural networks, cell assemblies are updated using Hebbian learning accomplished through [auto association](#). It is worth noting that Hebbian learning is unlike the most common methods of

learning used in training artificial neural networks. This is unfortunate since there are many problems for which Hebbian learning is appropriate and conventional end-to-end back propagation by gradient descent is inappropriate and unnecessarily introduces problems such as [catastrophic interference](#).

If you find the topic of grounding and Hebbian learning interesting, I'd be glad to discuss current theories about the role of cell assemblies in larger ensembles of reciprocally connected neurons such as the [global neuronal workspace model](#) of Stanislas Dehaene and Bernard Baars that attempts to explain how diverse cell assemblies throughout the cortex are recruited to solve novel problems. You might also be interested in a new [Hopfield network model](#) with continuous states that can store exponentially (with the dimension) many patterns, converges with one update, and has exponentially small retrieval errors.

References

- [1] Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Stephen Clark, Andrew Dudzik, Petko Georgiev, Aurelia Guy, Tim Harley, Felix Hill, Alden Hung, Zachary Kenton, Jessica Landon, Timothy Lillicrap, Kory Mathewson, Alistair Muldal, Adam Santoro, Nikolay Savinov, Vikrant Varma, Greg Wayne, Nathaniel Wong, Chen Yan, and Rui Zhu. Imitating interactive intelligence. *CoRR*, arXiv:2012.05672, 2020.
- [2] Alfred V Aho, Brian W Kernighan, and Peter J Weinberger. *The AWK programming language*. Addison-Wesley Longman Publishing Co., Inc., 1987.
- [3] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. In *International Conference on Learning Representations*, 2018.
- [4] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning distributed representations of code. In *Proceedings of the ACM on Programming Languages*, volume 3, New York, NY, USA, 2019. Association for Computing Machinery.
- [5] Bernard Baars, Stan Franklin, and Thomas Ramsy. Global workspace dynamics: Cortical "binding and propagation" enables conscious contents. *Frontiers in Psychology*, 4:200, 2013.
- [6] Alan Baddeley. Working memory: Theories, models, and controversies. *Annual Review of Psychology*, 63(1):1--29, 2012.
- [7] Alan Baddeley and Graham James Hitch. Working memory. In G.A. Bower, editor, *Recent Advances in Learning and Motivation*, volume 8, pages 47--90. Academic Press, 1974.

- [8] Andrea Banino, Adrià Puigdomènech Badia, Raphael Köster, Martin J. Chadwick, Vinícius Flores Zambaldi, Demis Hassabis, Caswell Barry, Matthew M Botvinick, Dharshan Kumaran, and Charles Blundell. Memo: A deep network for flexible combination of episodic memories. *CoRR*, arXiv:2001.10913, 2020.
- [9] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4509--4517. Curran Associates Inc., 2016.
- [10] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, arXiv:1806.01261, 2018.
- [11] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, arXiv:1612.00222, 2016.
- [12] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, arXiv:1506.03099, 2015.
- [13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41--48, New York, NY, USA, 2009. ACM.
- [14] John Hughlings Jackson [Biography]. An Introduction to the Life and Work of John Hughlings Jackson: Introduction. *Medical History. Supplement*, pages 3--34, 2007.
- [15] Matthew M. Botvinick. Multilevel structure in behaviour and in the brain: a model of fuster's hierarchy. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 362:1615--1626, 2007.
- [16] Valentino Braitenberg. Cell assemblies in the cerebral cortex. In Roland Heim and Günther Palm, editors, *Theoretical Approaches to Complex Systems*, pages 171--188, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.
- [17] Charlotte O. Brand, Alex Mesoudi, and Paul E. Smaldino. Analogy as a catalyst for cumulative cultural evolution. *PsyArXiv*, 2020.
- [18] Silvia Bunge, Carter Wendelken, David Badre, and Anthony Wagner. Analogical reasoning and prefrontal cortex: Evidence for separable retrieval and integration mechanisms. *Cerebral Cortex*, 15:239--49, 2005.

- [19] György Buzsáki. Neural syntax: Cell assemblies, synapsembles, and readers. *Neuron*, 68(3):362--385, 2010.
- [20] Ryan T. Canolty, Karunesh Ganguly, Steven W. Kennerley, Charles F. Cadieu, Kilian Koepsell, Jonathan D. Wallis, and Jose M. Carmena. Oscillatory phase coupling coordinates anatomically dispersed functional cell assemblies. *Proceedings of the National Academy of Sciences*, 107:17356--17361, 2010.
- [21] Alexandra Carstensen and Michael C Frank. Do graded representations support abstract thought? *Current Opinion in Behavioral Sciences*, 37:90--97, 2020.
- [22] Dorothy Cheney and Robert Seyfarth. Précis of how monkeys see the world. *Behavioral and Brain Sciences*, 15:135--147, 2011.
- [23] E.V. Clark. *First Language Acquisition*. Cambridge University Press, 2009.
- [24] Eve Clark. Grounding and attention in language acquisition. *Papers from the 37th meeting of the Chicago Linguistic Society*, 1:95--116, 2002.
- [25] Eve V. Clark. *Common Ground*, pages 328--353. John Wiley & Sons, Ltd, 2015.
- [26] Herbert H. Clark. *Using Language*. Cambridge University Press, Cambridge, 1996.
- [27] Maxwell Crouse, Constantine Nakos, Ibrahim Abdelaziz, and Kenneth Forbus. Neural analogical matching. *CoRR*, arXiv:2004.03573, 2020.
- [28] Forbus Kenneth D., Ferguson Ronald W., Lovett Andrew, and Gentner Dedre. Extending SME to handle large-scale cognitive modeling. *Cognitive Science*, 41(5):1152--1201, 2016.
- [29] Thomas Dean. Interaction and negotiation in learning and understanding dialog. https://web.stanford.edu/class/cs379c/resources/dialogical/zanax_DOC.dir/index.html, 2014.
- [30] Thomas Dean, Biafra Ahanonu, Mainak Chowdhury, Anjali Datta, Andre Esteva, Daniel Eth, Nobie Redmon, Oleg Rummyantsev, and Ysis Tarter. On the technology prospects and investment opportunities for scalable neuroscience. *CoRR*, arXiv:1307.7302, 2013.
- [31] Thomas Dean, Maurice Chiang, Marcus Gomez, Nate Gruver, Yousef Hindy, Michelle Lam, Peter Lu, Sophia Sanchez, Rohun Saxena, Michael Smith, Lucy Wang, and Catherine Wong. Amanuensis: The Programmer's Apprentice. *CoRR*, arXiv:1807.00082, 2018.
- [32] Thomas Dean, Chaofei Fan, Francis E. Lewis, and Megumi Sano. Biological blueprints for next generation AI systems. *CoRR*, arXiv:1912.00421, 2019.

- [33] B. Deen, H. Richardson, D. D. Dilks, A. Takahashi, B. Keil, L. L. Wald, N. Kanwisher, and R. Saxe. Organization of high-level visual cortex in human infants. *Nature Communications*, 8:13995, 2017.
- [34] Stanislas Dehaene. *Consciousness and the Brain: Deciphering How the Brain Codes Our Thoughts*. Viking Press, 2014.
- [35] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *CoRR*, arXiv:2006.08381, 2020.
- [36] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The Structure-Mapping Engine: Algorithm and Examples. *Artificial Intelligence*, 41(1):1--63, 1989.
- [37] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence Journal*, 2:189--208, 1971.
- [38] Jason Fischer, John G. Mikhael, Joshua B. Tenenbaum, and Nancy Kanwisher. Functional neuroanatomy of intuitive physical inference. *Proceedings of the National Academy of Sciences*, 113(34):E5072--E5081, 2016.
- [39] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable Inception Graph Neural Networks. *CoRR*, arXiv:2004.11198, 2020.
- [40] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations*, 2019.
- [41] Joaquín M. Fuster. *Chapter 8: An Overview of Prefrontal Functions*, pages 375--425. Elsevier, London, 2015.
- [42] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155--170, 1983.
- [43] Mary L. Gick and Keith J. Holyoak. Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1--38, 1983.
- [44] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471--476, 2016.

- [45] A. E. Green, D. J. Kraemer, J. A. Fugelsang, J. R. Gray, and K. N. Dunbar. Neural correlates of creativity in analogical reasoning. *Journal of Experimental Psychology, Learning, Memory and Cognition*, 38(2):264--272, 2012.
- [46] Mareike Grotheer, Zonglei Zhen, Garikoitz Lerma-Usabiaga, and Kalanit Grill-Spector. Separate lanes for adding and reading in the white matter highways of the human brain. *Nature Communications*, 10:3675, 2019.
- [47] Jessica B Hamrick. Analogues of mental simulation and imagination in deep learning. *Current Opinion in Behavioral Sciences*, 29:8--16, 2019.
- [48] Marc D. Hauser and Elizabeth Spelke. Evolutionary and developmental foundations of human knowledge: A case study of mathematics. In M. Gazzaniga and N. Logothetis, editors, *The Cognitive Neurosciences, III*, pages 853--864. MIT Press, Cambridge, MA, 2004.
- [49] Donald O. Hebb. *The organization of behavior: A neuropsychological theory*. Wiley, New York, 1949.
- [50] Douglas Hofstadter. *I Am a Strange Loop*. Basic Books, 2007.
- [51] Keith J. Holyoak. Analogy and relational reasoning. In Keith J. Holyoak and Robert G. Morrison, editors, *Oxford Handbook of Thinking and Reasoning*, pages 234--259. Oxford University Press, 2012.
- [52] Presley A. Ifukor. Modelling the mapping mechanism in metaphors. *Cognitive Science*, 6:21--44, 2005.
- [53] Daniel D. Johnson. Learning graphical state transitions. In *International Conference on Learning Representations*, 2017.
- [54] Sham Machandranath Kakade. On the sample complexity of reinforcement learning. PhD Thesis - Gatsby Computational Neuroscience Unit, University College London, 2003.
- [55] J. G. Klinzing, B. Rasch, J. Born, and S. Diekelmann. Sleep's role in the reconsolidation of declarative memories. *Neurobiological Learning and Memory*, 136:166--173, 2016.
- [56] P. A. Lewis, G. Knoblich, and G. Poe. How Memory Replay in Sleep Boosts Creative Problem-Solving. *Trends Cognitive Science*, 22(6):491--503, 2018.
- [57] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. *CoRR*, arXiv:1511.05493, 2015.
- [58] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. In *International Conference on Learning Representations*, 2018.

- [59] Phan Luu, Alexandra Geyer, Cali Fidopiastis, Gwendolyn Campbell, Tracey Wheeler, Joseph Cohn, and Don M. Tucker. Reentrant processing in intuitive perception. *PLOS ONE*, 5(3):1--10, 2010.
- [60] George A. Mashour, Pieter Roelfsema, Jean-Pierre Changeux, and Stanislas Dehaene. Conscious processing and the global neuronal workspace hypothesis. *Neuron*, 105(5):776--798, 2020.
- [61] Josh Merel, Matt Botvinick, and Greg Wayne. Hierarchical motor control in mammals and machines. *Nature Communications*, 10(1):5489, 2019.
- [62] Dharmendra S. Modha and Raghavendra Singh. Network architecture of the long-distance pathways in the macaque brain. *Proceedings of the National Academy of Sciences*, 107(30):13485--13490, 2010.
- [63] Sushobhan Nayak and Amitabha Mukerjee. Grounded language acquisition: A minimal commitment approach. In *Proceedings of COLING 2012*, pages 2059--2076, Mumbai, India, 2012.
- [64] John von Neumann. First draft of a report on the EDVAC. Technical report, Institute for Advanced Study, 1945.
- [65] Klaus Oberauer. Chapter 2 design for a working memory. In *The Psychology of Learning and Motivation*, volume 51 of *Psychology of Learning and Motivation*, pages 45--100. Academic Press, 2009.
- [66] Klaus Oberauer. Working Memory and Attention - A Conceptual Analysis and Review. *Journal of Cognition*, 2(1):36, 2019.
- [67] Günther Palm, Andreas Knoblauch, Florian Hauser, and Almut Schüz. Cell assemblies in the cerebral cortex. *Biological Cybernetics*, 108:559--572, 2014.
- [68] L.A. Petitto and P.F. Marentette. Babbling in the manual mode: evidence for the ontogeny of language. *Science*, 251(5000):1493--1496, 1991.
- [69] Laura Ann Petitto, Siobhan Holowka, Lauren E. Sergio, and David Ostry. Language rhythms in baby hand movements. *Nature*, 413:35--36, 2001.
- [70] Thomas Pierrot, Guillaume Ligner, Scott E. Reed, Olivier Sigaud, Nicolas Perrin, Alexandre Laterre, David Kas, Karim Beguir, and Nando de Freitas. Learning compositional neural programs with recursive tree search and planning. *CoRR*, arXiv:1905.12941, 2019.
- [71] David Poeppel, Karen Emmorey, Gregory Hickok, and Liina Pyykkänen. Towards a new neurobiology of language. *The Journal of Neuroscience: The official journal of the Society for Neuroscience*, 32:14125--14131, 2012.

- [72] F. Pulvermüller, M. Garagnani, and T. Wennekers. Thinking in circuits: toward neurobiological explanation in cognitive neuroscience. *Biological Cybernetics*, 108(5):573--593, 2014.
- [73] Friedemann Pulvermüller. How neurons make meaning: brain mechanisms for embodied and abstract-symbolic semantics. *Trends in Cognitive Sciences*, 17(9):458--470, 2013.
- [74] Friedemann Pulvermüller. Neural reuse of action perception circuits for language, concepts and communication. *Progress in Neurobiology*, 160:1--44, 2018.
- [75] B. Rasch and J. Born. About sleep's role in memory. *Physiological Review*, 93(2):681--766, 2013.
- [76] Scott E. Reed and Nando de Freitas. Neural programmer-interpreters. *CoRR*, arXiv:1511.06279, 2015.
- [77] R.W. Rieber, A.S. Carton, L.S. Vygotsky, N. Minick, J. Wollock, J.E. Knox, and J.S. Bruner. *The Collected Works of L.S. Vygotsky: Volume 1: Problems of General Psychology, Including the Volume Thinking and Speech*. Cognition and Language: A Series in Psycholinguistics. Plenum Press, 1987.
- [78] Dario D. Salvucci and John R. Anderson. Integrating analogical mapping and general problem solving: the path-mapping theory. *Cognitive Science*, 25(1):67--110, 2001.
- [79] Hendrig Sellik. Natural language processing techniques for code generation. Delft University of Technology Technical Report, 2019.
- [80] E. S Spelke and K. D. Kinzler. Core knowledge. *Developmental Science*, 10:89--96, 2007.
- [81] Elizabeth Spelke. What makes us special? Brain Inspired Podcast Episode #48 September 25, 2019, 2019.
- [82] Ian Stewart and Jack Cohen. *Figments of Reality: The Evolution of the Curious Mind*. Cambridge University Press, 1997.
- [83] Barbara Tversky. *Mind in Motion: How Action Shapes Thought*. Basic Books, 2019.
- [84] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, arXiv:1711.00937, 2017.
- [85] Ke Wang. Learning scalable and precise representation of program semantics. *CoRR*, arXiv:1905.05251, 2019.
- [86] Ke Wang, Rishabh Singh, and Zhendong Su. Dynamic neural program embedding for program repair. *International Conference on Learning Representations*, 2018.

- [87] Ke Wang and Zhendong Su. Learning blended, precise semantic program embeddings. *CoRR*, arXiv:1907.02136, 2019.
- [88] Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. Code generation as a dual task of code summarization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6563--6573. Curran Associates, Inc., 2019.
- [89] P. J. Whelan. Control of locomotion in the decerebrate cat. *Progress in Neurobiology*, 49(5):481--515, 1996.
- [90] Kirstie J. Whitaker, Michael S. Vendetti, Carter Wendelken, and Silvia A. Bunge. Neuroscientific insights into the development of analogical reasoning. *Developmental science*, 21:e12531, 2018.
- [91] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, arXiv:1901.00596, 2019.
- [92] Yujun Yan, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi. Neural execution engines: Learning to execute subroutines. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020.
-