(Project Narrative)

A Category-Theoretic Approach to Agent Interaction: Information, Communication, Planning, and Learning

David I. Spivak

Contents

Chapter 1

Statement of Objectives

Whether we consider an "agent" as an individual or as a large system like a corporation, government, or military, we can say the following.

> In order to be effective an agent must consistently make successful plans and decisions, based on highly disparate sources of data.

This data comes in many different forms—from the agent's own sensors, memory, and conceptual frameworks, as well as from the sensors, memories, or conceptual frameworks of other agents. The data stands as a record of events that took place over multiple time-scales, and whose relevance to an upcoming decision is often unclear.

We seek a mathematical framework for specifying various aspects of an autonomous agent and its interactions with other agents, including humans. Just like linear algebra offers tools and ideas that can be used in a wide range of applications—machine learning, statistics, differential equations, graphics processing, and so on—we need a mathematical language and toolset that can be used to conceptualize and interrelate the sorts of concerns that arise when considering autonomous agents and their interaction, in all its forms.

In this proposal, we lay out a vision for using category theory to manage the interaction of various sorts of agents (including human agents), as well as the sorts of planning and learning that makes autonomy possible. Our objectives are:

1. Mathematically specify an intuitive and compositional interface for human-computer interaction, one that incorporates various strengths of logic to present the user with a graphical reasoning system for stored knowledge.
2. Define a new notion of temporal database based on topos theory, with a more robust and expressive query language and a greater ability to exchange data between various temporal databases and the agents who use them.
3. Explore the idea of planning as proof-search, where the notion of logical truth is replaced with that of reliability, and where successes and failures in executing plans are translated into promotions and demotions in reliability for the logical statements of which the plans are composed.
4. Develop the recently-noticed formal connection between deep learning algorithms and compositional economic games, e.g. the possibility of a learning, game-playing hybrid agent, capable of making economic transactions and learning from the results.

Chapter 2

Research Effort

2.1 Information and communication

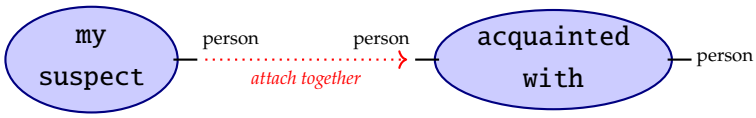
Information and communication are at the heart of how networks function. Current systems for storing information in a computer seem better-suited for static information, whereas humans provide the contextual and processual knowledge necessary to put these static facts together. In Section 2.1.1, we discuss how to improve the interaction between humans and computers, with the above sort of division of labor, through a graphical user interface that is both formal and intuitive. In Section 2.1.2, we discuss potential ways to improve how process-oriented information can be stored in computers, by formalizing a more robust form of temporal database.

2.1.1 Human-computer interaction

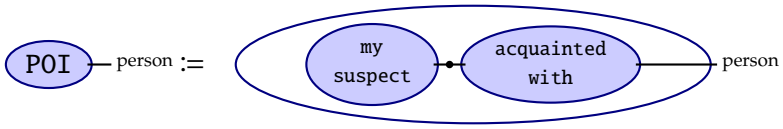
In order to solve complex problems, humans need to interact seamlessly with computers. Popular culture references like the movie *Minority Report* show an interface by which human thoughts and reasoning are directly translated into computational artifacts and operations. The interface itself is a sort of language game, a set of rules governing manipulations in an algebra of relationships.

An example scenario

Imagine you are an investigator on a case, searching for the acquaintances of a particular person, say the suspect in the case. Interacting with your computer, you pull up the `acquainted_with` cell, attach to it the cell corresponding to your suspect,



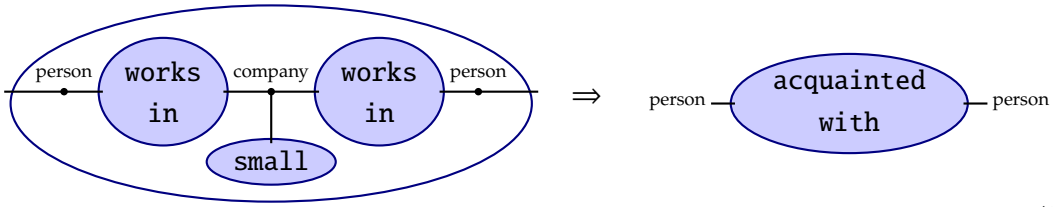
and define `POI` (person of interest) as the result:



A seamless interface between the human and the computer must "speak" directly to each. The visual language of wiring diagrams can be formalized in terms of a category-theoretic language of operads [Spi13], and hence implemented in a computer. Here we

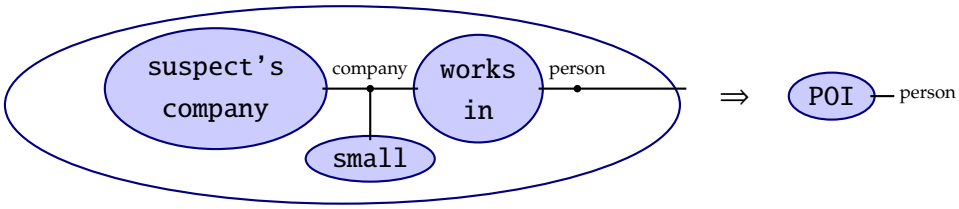
want to consider additional logical rules which govern how relationships are combined, and how these logical rules cohere with both the visual language of wiring diagrams and its operadic underpinning.

For example, your agency's computer system has stored many sources of acquaintances: church groups, kid's friends' parents, etc. You find the following implication in the knowledge base:



(2.1)

The implication in Eq. (2.1) encapsulates the belief that if two people work in the same company, and that company is small, then the two people are acquainted. Defining `suspect's_company` to be the result of attaching `my_suspect` and `works_in`, the compositionality of the logical rules then provide a source of `POI`'s.



(2.2)

To be sure you're reading it correctly, you print out the English text translation, which reads "Any person that works in a company that is both small and is also my suspect's company is a person of interest."

You then instantiate the suspect's company relation to see what's inside:

. . . .

suspect's company
Phil's moving company
Blackwater Inc.

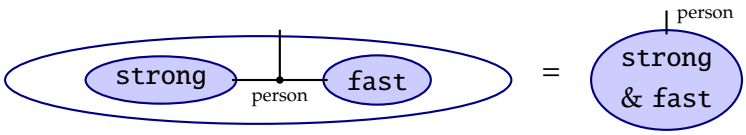
You note the Blackwater connection, but pursue the compound cell to the left of Eq. (2.2). The computer automatically filters out Blackwater Inc, keeping only the small moving company, and returns a table consisting of what turns out to be five other persons who work there. You attach this cell to the `person's_home_address` cell, attach the result to your car's map/route-finding algorithm, and a course is set for visiting the five addresses on your list.

The underlying logical system

The above scenario paints a picture of an investigator connecting various relationships, instantiating them as tables, and continuing with any given result as a new relationship,

either filtering it according to some criterion or substituting it into a cell of another concept web. The key aspect that makes this work is actually invisible in the story: it is the underlying mathematical logic, which ensures that the above interactions mean the same thing to the investigator as they do to the computer assistant. Let's consider what is going on under the hood.

The diagrams shown above have the flavor of concept webs—easy to produce and extract meaning from—and yet these diagrams are much more robust to reasoning than ordinary concept webs are. Indeed, substitution of one diagram into another, the fact that this operation preserves implication ($\Rightarrow$), how substitution works with logical operators like AND (&) and OR ($\vee$), etc., should all follow rules that are logically consistent, interoperable, and intuitive. For example the framework should enforce the following equality:



Connecting `strong` and `fast` should "obviously" produce `strong&fast`, but this sort of rule—like the rules about preserving implications discussed above—must be explicitly part of the underlying mathematical framework.

We aim to show that all of *regular, coherent, and geometric logic* can be encoded into rules for manipulating such wiring diagrams.[1] Technically speaking, the above wiring diagrams are the morphisms in an operad $\mathcal{W}$, and the detective agency's knowledge base forms a finitely-presented subalgebra of the algebra of all relations $\mathsf{Rel}\colon \mathcal{W} \to \mathbf{Set}$ [Spi12b]. However, the collection of relations that can inhabit a cell form not only a set, but a complete distributive lattice, meaning that the union or intersection of relations is still a relation. By having the algebra Rel target meet-semilattices, distributive lattices, or complete distributive lattices, rather than targeting **Set**, and by demanding it satisfy some extra conditions (e.g. Rel should be a coCartesian operad functor), we should be able to ensure the sorts of derivation rules that not only make the system easy and intuitive for our investigator to use, but also powerful and reliable as an instantiation of regular, coherent, or geometric logic.

A possible critique and response

We have discussed knowledge in terms of a storehouse of facts, like Eq. (2.1) "any two people who work in the same small company are acquainted," and one may offer the critique that this is rehashing the path of *good old-fashioned artificial intelligence (GOFAI)*. The critique emphasizes that real knowledge is error-prone, i.e. that it can easily

[1] Regular logic is less expressive than coherent logic, which is itself less expressive than geometric logic; the latter has been proposed as a general logic for software specification [Vic93]. For automated proof search, however, the order of preference is reversed: regular logic proofs are more easily automated than coherent logic proofs, which are more easily automated than geometric logic proofs. Thus, we cannot simply choose the "best" and work with it; different logics have different advantages.

contain inconsistencies, and that this fact—together with its disadvantages as well as advantages!—is an important aspect of human reasoning and should not be ignored.

We are in complete agreement with this critique and have been careful not to ignore it. Instead of taking our knowledge (like Eq. (2.1)) to be statements of fact that are either true or false, we take them to be beliefs that are more or less *reliable*. We will discuss this idea in greater detail in Section 2.2.1, where it will show up again in the context of planning.

Summary. By providing a formalism for concept webs by which one can substitute, zoom in and out, remove or insert new cells, etc., all according to strict category-theoretic semantics that can be implemented in functional programming languages, we move one step closer to a seamless human-computer interface for thinking together.

2.1.2 More expressive and robust temporal databases

Information is almost never timeless; facts are only valid for some period of time. For example, information about the weather, or who is leading a group of agents, or who its friends or enemies are, or where something is located—all of these are non-permanent. The temporality of facts constitutes an implicit aspect of all process plans, where the arising truth or falsity of a fact ("the object has come into view", "preparations are complete") is used to indicate the moment where a new step should begin.

Temporal data as sheaves, and the logic of toposes

The idea that facts are valid over certain time-intervals can be formalized. First we notice that there are two evident rules about this association of valid time intervals to facts:

1. if a fact is valid on an interval, then it is valid on any subinterval; and
2. if a fact is valid on a set of intervals, then it is valid on their union.

Together, these two rules say that facts form a *sheaf* on the topological space $\mathbb{R}$. Sheaves are information structures that inter-relate localized data, automatically fusing together those time-local sections that fit together, to form unified global data.

But to talk about data we should go a bit further: data is generally organized according to some organizational schema. The data itself must fit the form predefined by its schema, and each source of data—each type of sensor, each laboratory, etc.—will generally need to organize its data according to a schema that is as unique as the data source itself. Databases, knowledge bases, ontologies, etc. provide general rules for what these various organizational schemas can look like. In one conception that fits both databases and knowledge representation [Spi12b; SK12; Sch+17], a schema is a category $\mathcal{S}$ and fitting data is given by a functor $D\colon \mathcal{S} \to \mathbf{Set}$. This point of view has good properties when it comes to data migration and data integration [SW15; Wis+15].

Combining the above two mathematical conceptions—of temporality and databases—we arrive at a notion of temporal databases. Technically speaking, these should be something like *category objects and internal presheaves on them, within the topos of sheaves on* $\mathbb{R}$. As category objects in a topos, our notion of temporal database would come ready-made with an internal language, logic, and type theory.

To repeat, the logic of temporal databases does not have to be created ad hoc; it comes with the category-theoretic package, so to speak. Rather than answering the question of *whether* some fact is valid according to the database, truth values in the topos-theoretic logical system naturally answer the question of *when*—over what intervals of time—a given fact is valid. For example, the truth of the statement "Bob lives in Canada" consists of intervals of time, "(2006 to 2009) and (2012 to 2016)", rather than a mere true or false. The logical constant TRUE is given by the infinite interval ($-\infty$ to ∞) and that of FALSE is given by the empty interval $\emptyset$; conjunction (AND) and disjunction (OR) are given by the intersection and union of intervals, respectively.

Incorporating probability into the logic

The above description of topos-theoretic temporal databases is currently unproven; it is merely our current best guess, based on similar work [SS18] in temporal type theory. It is our goal to work these ideas out in full, as well as to expand upon them in various ways.

For example, it may be useful to add another dimension to temporal data: probability. For any given statement about the world, `Bob is in Texas` or `It is raining`, there is a set of time intervals on which the statement is true. Assigning to each time interval a *probability* of the statement being true, a reasonable theory should demand that this probability increases as the intervals decrease in size. Indeed, it is more likely that `Bob is in Texas` on September 5, 2018 than it is that `Bob is in Texas` throughout the entire month of September 2018. But what other rules are there for combining temporality and probability?

Recent research has produced inklings of a category-theoretic structure combining temporality and probability using something called *continuous valuations on an internal locale*, which may result in a generalization of *stochastic processes*, and which would be completely interoperable with (in fact, defined in terms of) the internal language and logic of the temporal database topos, as discussed above. Combining this with new category-theoretic approaches to probability theory, e.g. [FP17], could be part of a 21st century re-grounding of probability theory, statistics, and stochasticity.

Comparison to today's temporal databases

Temporal databases, as they are defined today, are generally rather ad hoc: the underlying mathematical formalism, e.g. relational algebra treats *valid time* as a database column like any other, even though they are playing very different roles. One is data, and the other is metadata. The applications may "know" this but the relational schema doesn't:

in an important sense, the column for valid time intervals is treated theoretically as no different from a column storing something like `First Names`.

We believe that a more robust logic would emerge if temporality was taken as a central aspect of the mathematical formalism, rather than an afterthought. In our proposed formalism, valid time intervals constitute the "site" on which the whole sheaf-theoretic model rests. Valid times are no longer captured by an ordinary data field; they constitute the truth values themselves, as discussed above. The goal of the project would be to get a feel for how this topos-theoretic logic and its semantics work in practice: does the math give us access to the kinds of operations we need from a temporal database?

We have good reason to hope that the notions provided by the topos-theoretic formalism will be highly expressive and relevant. Indeed, recent work [SS18] has shown that by working in a slightly "larger" topos $\mathcal{B}$ of *behavior types*, one can express properties of interacting dynamical systems of any conceivable sort. For example, discrete, continuous, hybrid, deterministic, and non-deterministic dynamical systems, defined by difference equations, differential inclusions, delay, arbitrary logical and time-varying properties, and much more can all be captured in this topos-theoretic formalism. It was shown that one can combine all of these different modeling formalisms into a big-tent framework, in the form of the topos $\mathcal{B}$, and prove properties of systems of systems, such as safe-separation of airplanes in the National Airspace System, within $\mathcal{B}$'s internal language and logic.

Thus we believe that by combining temporal databases with behavior types, we can produce many useful and interesting results. We propose to study temporal databases in their own right, as well as their interaction with agents, formalized as behavior types $X \in \mathcal{B}$, which access various databases and use them to record facts and make decisions about future actions.

Summary. Temporal databases, in which the truth of a fact is measured by the interval on which it is valid, can be modeled in terms of sheaves on the topological space $\mathbb{R}$. The resulting formalism comes with a ready-made internal language, which we conjecture supports and integrates temporal and probabilistic reasoning. We will study how this logical system can make for more expressive and robust temporal database queries, as well as facilitate the interaction (data migration) between different temporal databases.

2.2 Intelligence and planning

2.2.1 Planning and logic

Under one account, good old-fashioned AI (GOFAI) did not achieve the goals it set for itself because it put too much stock in logical reasoning. In hindsight, attempting to understand the world by accumulating encyclopedias of knowledge formulated in the language of first-order logic seems like a fool's errand. And yet it is true that logic—reasoning with rules—does play a major part not only in science and technology, but also in day-to-day discussions among ordinary people, regardless of how "consistent" their use of logic may in fact be. Logic was not invented out of whole cloth; like so much of science and technology, it was bio-inspired: people looked within at their most reliable sorts of thoughts and sought to formalize them.

From this point of view, the error was not use of logic in AI, but instead *mistaking reliability for truth*. Logical rules are very useful, as evidenced by the immense power of modern technology. But in order for logic to be effective in some arena it must operate on a fairly complete system of axioms that is specific to that arena. And yet such a complete axiom system regarding a specific real-world situation is always wrong! Exceptions are the rule in natural languages because exceptions are the rule in life. The point is that wrong, inconsistent axioms can still lead to success in carrying out projects. Perhaps intelligence is about continued success, rather than about truth.

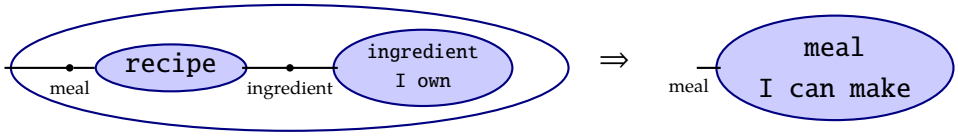
We propose that a system exhibiting intelligent behavior may very well use the scaffolding of some sort of logic in order to make, reason about, and communicate decisions and knowledge about the world, even if this knowledge is inconsistent.[2] Of course the worry is "but then the system could prove anything", to which we respond "but why would it want to?" If the system is supposed to use its knowledge to survive and be effective in the world, and if it gets punished whenever its plans fail to materialize, then it might well learn to avoid using inconsistencies to prove, i.e. expect, false things. Let's make this idea more precise.

2.2.2 Planning as recursive proof search

Consider an arbitrary planning task, such as "make a meal" or "get humans to Mars." Rather than thinking of it as planning to make the event take place, think of it as proving that the event is possible: prove that you *can* make a meal or prove that you *can* get humans to Mars. In both cases, the planner can use stored assumptions and beliefs about the world, and it can create arbitrary strategies to pursue the proof. It can also employ subordinate planners to take care of details, which would constitute *lemmas* in the overall proof.

[2] Similar ideas, namely of dealing with inconsistent logic, have been proposed, e.g. [Roo92]. However, they seem to require that whenever inconsistent statements are found, one of them must be discarded. We make no such requirement, instead asking only that non-executable statements be demoted; we explain in the main text.

For example, a lemma about making dinner might be "to make a meal, it is sufficient to find a recipe whose ingredients I own."



The system uses this lemma to change its proof goals: now it needs to prove that it can find a recipe and that it owns the ingredients in that recipe.

Subordinate planners, promotion and demotion

Each planner must create lemmas—planning subtasks—that together are enough to complete its own task. These lemmas act like milestones indicating progress. The planner must also take each such lemma and hand it off to a subordinate planner, which it deems capable of handling (proving that it can accomplish) that planning subtask. For example, to prove that it owns certain ingredients, the planning agent could create a lemma like "if I see the ingredient in the pantry or cabinet then I own it" and tell a subordinate planner to prove it can see the ingredient in the pantry. In turn the subordinate realizes that to prove it can see the ingredient in the pantry, it should first prove that it can walk to the pantry.

At any point a lemma can fail: either subordinates fail to prove it, or the execution engine fails to execute a proven plan. For example, I may take as an axiom that I can move my arm, but sometimes my arm just doesn't move. Perhaps the planning agent walks to the pantry but fails to see the ingredient: seeing has failed to execute. In this case, a low-level proof (plan) has failed, but the larger plan may still succeed. For example, the planner might realize that to prove ownership of salt, it is sufficient to prove the lemma "I can purchase salt at the grocery store," and go about trying to prove it.

Thus a subordinate planner can fail the mission it was assigned, either because the lemma is unproven after a certain time-period has elapsed (time-out) or a proven lemma has failed to execute. Either way, this failure is reported to the higher-level planner, who then needs to deal with this setback. To do so, it must both re-plan its own mission and learn from its failure to assign an achievable task to its subordinate.

Of these, the former has already been handled: re-planning is the same as planning in that one finds lemmas that should be feasible and then hands them off to subordinates to prove. But how should the planner learn from success or failure? When a subordinate is unable to achieve its mission, the higher-level planner should *demote* its reliance on that particular subordinate for that particular task, as if to say "next time I will not assume this subordinate can handle this task; perhaps none of my subordinates can." Subordinates should be called upon when their past record indicates that they are likely to succeed. Note that if the higher-level planner cannot re-plan fast enough, it too will be demoted in some capacity by its superior, and hence called on less frequently in the future.

The process of assigning blame and credit, both to oneself as a planner and to other planners, is the learning part. It is here that knowledge and facts, logical statements, are deemed more or less useful—promoted or demoted—rather than considered as true or false. Thus we could imagine a library (e.g. a "hint database" in Coq) that is constantly being managed by the system, where beliefs are promoted and demoted according to their history of success.

Exactly how this reliability and credit assignment should be managed to optimize future success is a sort of secret sauce that we do not claim to understand. Instead, our goal is to formalize the position and role of this credit-assignment aspect within the planning-as-proof model. We want to understand and formalize the sort of *selection* process and resultant *evolution* that can take place in such a system.

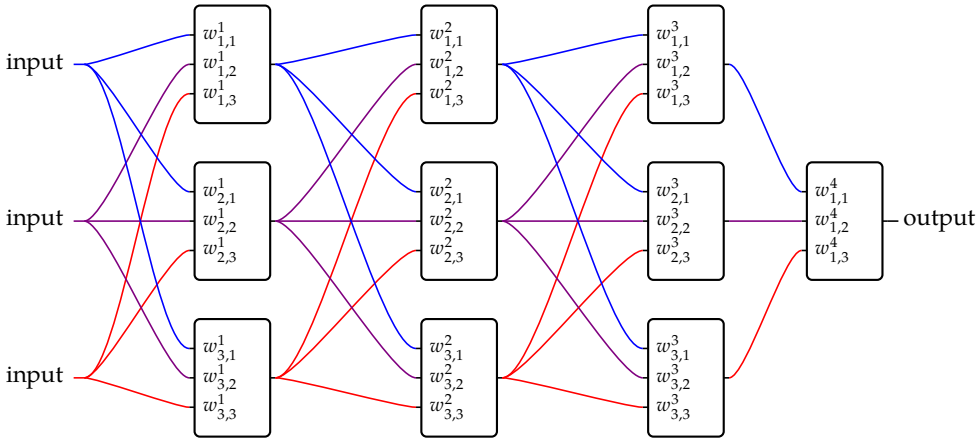
Summary. We propose to consider whether planning and general sorts of intelligence tasks can be understood in terms of an embodied agent, one that can create planning tasks for subordinate planners, and who must survive and remain relevant in a world. These planning agents could use a logical scaffolding to make and reason about the world, promoting and demoting beliefs in accordance with their proven usefulness in executing tasks.

2.3 Learning

Machine learning is still in its infancy. Like the parents of an infant, researchers today still have very little understanding of what's really going on inside these learners. But it is important to develop such an understanding if we want these learners to realize their full potential.

2.3.1 Understanding machine learners

One of the most opaque areas of machine learning is that of neural networks. A trained neural network—say one trained to classify images—can be imagined as an organized system of numbers (called weights and biases).



(2.3)

A researcher examining even a modestly-sized neural network has no idea how these numbers (e.g. all of the $w^k_{i,j}$ above) come together to produce the network's overall image classification ability. Human perception is very similar: it is hard to say how incoming signals are transformed throughout the brain to classify and find relevant features in raw sensory data.

It is not misleading to imagine neural networks as perceptive organs; indeed, they are little more than organized systems of continuous perceptrons, very close in spirit to those invented by Minsky and Papert in 1969 [MP69]. In order to reach the higher cognitive abilities of humans, as Minsky himself [Min88] and many others have later said, machine learners need much more of an ability to structure and restructure the network of interconnections throughout which the information propagates. In turn, we as researchers need to better articulate—distinguish and organize the various "moving parts" within—these information networks.

This is precisely the sort of goal for which category theory has proven most useful in the past. It provides a wealth of mathematical descriptions for various sorts of organizational structures, as well as formal methods for restructuring, i.e. for translating information from one such structure to another. Below we will briefly explain some recent research in this area, conducted by our group, and then some intriguing formal connections to compositional economic game theory.

A categorical underpinning for supervised learning

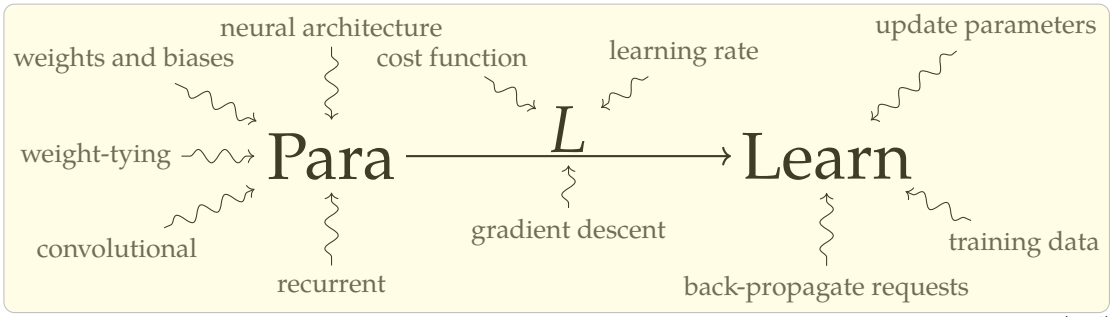
In [FST17], we showed that supervised learners—in particular the "neurons" of a neural network—form what's called a *monoidal category*, and that gradient descent can be formalized as a monoidal functor. As explained below, this roughly formalizes the idea that gradient descent mechanizes a data scientist's architectural decisions into an actual learning algorithm.

The basic story of supervised learning via neural network involves a host of keywords, such as

- neural architecture
- weights and biases
- gradient descent
- training data
- cost function
- learning rate
- back-propagation
- parameter update
- recurrent neural nets
- convolutional neural nets
- weight tying

These can be woven into a story about how neural nets function, but it has many moving parts and can be hard to follow. One may prefer a category-theoretic description, since it is a completely formal artifact that articulates the same story using ideas that are quite general in their applicability. We will see later that by formulating the ideas in this way, new connections to disparate fields can serendipitously appear. So here we briefly recall the more technical, categorical description of supervised learning from [FST17].

Neural architectures like Eq. (2.3) are string diagrams in a monoidal category of parameterized functions, where the parameters are called weights and biases. Choosing a cost function and a learning rate is enough to specify a strong monoidal functor to another monoidal category of *learners*. This functor translates the neural architecture into a learning architecture, assembling an identically-shaped network of learners with the same parameters as before, namely parameters consisting of weights and biases. But unlike mere parameterized functions, these learners update their parameters based on training data.



(2.4)

Thus the strong monoidal functor $L\colon$ **Para** $\to$ **Learn** shown in (2.4) organizes the basic story of neural networks, and shows the precise sense in which learning is compositional: "it's a strong monoidal functor!" Indeed, a strong monoidal functor is exactly that which preserves both serial and parallel composition: the act of combining a network

of parameterized functions into a single parameterized function and that of converting parameterized functions into learning algorithms "commute."

In fact, this categorical formalism provides a precise notion of what a supervised learning algorithm *is*, well beyond those that arise from neural networks. We do not get into the specifics here—they can be found in [FST17] as the definition of the morphisms in the category **Learn**—but, roughly speaking, a supervised learner is a "*parameterized* function that *updates* its parameters according to training data and back-propagates *requests* to its neighbors." The notion of neural network is then situated as just one functor into—one source of examples within—the world of supervised learning algorithms more broadly construed. For example, our framework would just as easily accommodate learners that do not use gradient descent at all but instead add experience to a database or use streaming algorithms [Bif13] to record the past in a way that can be quickly recalled for handling new problems that arise.

Some directions for future research

Having a categorical description of neural networks shows exactly where they currently sit within the broader categorical milieu, and hence gives a sense of some "neighboring" possibilities. For example, consider traced monoidal categories, which are monoidal categories equipped with an interpretation of feedback [JSV96; SSR16]. The network of neurons in the brain has massive amounts of feedback, whereas neural networks have none. (Even recurrent neural networks have no true feedback; what looks like feedback is merely a specification for how they unroll to handle sequences of data with some fixed length $k \in \mathbb{N}$.) It would be very interesting to find a similar traced monoidal category structure on a category of neuron-like learners.

This and many other open questions are written down in our recent article [FST17]. For example, perhaps there is a nice categorical description of reinforcement learning, like the (co-)algebraic approach of [FHM18], but which is also compositional. Having a variety of different learning architectures, all described category-theoretically, opens up the possibility of translating between them, so that they can each be used to solve those subproblems for which they are most suited, and the results combined category-theoretically.

Category theory is about making connections, and it took only one seminar talk on the above subject before an Oxford researcher names Jules Hedges noticed a connection to his work on *compositional economic game theory*. Out of all the directions for future research, this one is currently the most interesting and deserves its own discussion.

2.3.2 Learning, lenses, and games

It turns out that the structure of a morphism in **Learn**—which consists of a parameter space, an implementation function, an update function, and a request (back-propagate) function—is quite similar to the structure of a morphism in a certain symmetric monoidal

category **Game** of open economic games [Gha+16; Hed+16]. The connection (functor) between these two categories is roughly given by the following analogy:

Category: **Learn**	Category: **Game**
neurons	games
weights & biases	strategy profiles
implementation	play function
parameter update	Nash equilibrate
back-propagation	"co-play" function

The co-play function shown at the bottom right is unknown in classical economic game theory. This is because classical games are "closed"—they do not depend on a state of the world and they do not change the state of the world—whereas in compositional economic game theory, games are open. In this theory the co-play function acts like back-propagation in neural networks, giving feedback to other players (neurons) in the environment, which in turn informs their own games (learning).

This formal similarity between machine learners and economic games is surprising, but in fact they are not the only two members of this family. [Hed17] shows that the same categorical structure shared by learners and games is in fact also shared by something called lenses from database theory [BPV06; GJ12], which are becoming increasingly popular in functional programming [OCo11]. Lenses are used to solve the "view-update" problem, where a user accesses a small part of a large database and then provides feedback on what that part *should* look like. The lens is a specified way of updating the large database to account for the user's input. It turns out that symmetric lenses generalize learners in a formal sense, where roughly speaking, the database user plays the role of the neural network trainer.

Fleshing out this connection is a promising research topic. For example, combining game-playing agents and learners could provide a strong mathematical underpinning for artificial intelligence. A mathematical hybrid of objects in the categories **Learn** and **Game** might formalize the notion of "agents with preferences, who learn to improve their position over time." What more is intelligence?

Summary. Our new category-theoretic description of supervised learning articulates and generalizes the structure of neural networks. This opens up many questions, e.g. how to introduce true feedback into learning algorithms. In addition, there are exciting formal connections between learners and both database theoretic lenses and compositional economic games, which could tie together various aspects of generalized intelligence and decision-making.

2.4 Environmental impacts

This research is purely mathematical, and thus it will have no environmental impacts; compliance with environmental statutes and regulations is thereby assured.

Chapter 3

Principal Investigator Time

The PI will spend about half his time working on this project, and a postdoc will spend most of his or her time working on this project.

There is a clear distinction between what I propose here and the work I am doing for my other government grant, an AFOSR grant with Jean-Luc Cambier and Fred Leve. The latter is about dynamical systems and computational techniques. The proposed work for the present grant (with Doug Riecken) is about information, communication, planning, and learning. The only overlap is that category theory is a central aspect in both; the actual work is very different.

As one can see for themselves, there is no relationship or overlap between the left and right columns in the table below:

Doug Riecken	**Fred Leve/Jean-Luc Cambier**
Information and communication	Dynamical systems
Planning and learning	Computational techniques

Chapter 4

Facilities

Laboratory: N/A

Animal: N/A

Computer: The campus at the Massachusetts Institute of Technology is networked by both a wired 100/1000Mbps Ethernet LAN and the campus-wide 802.11a/b/g/n wireless networks. MIT utilizes both proprietary and open source workstations and servers, including Linux, Unix and Macintosh platforms. In addition to office workstations, MIT provides clusters of workstations throughout campus running a customized distribution of Linux derived from Ubuntu. Network security is provided by the Kerberos authentication protocol.

The Math Department maintains its own subnet and domain, and provides separate email, file storage, computational, and internet services for faculty and staff over a Gigabit Ethernet LAN. Faculty workstations run Fedora Linux, and department servers run Ubuntu Linux and Fedora Linux. The department maintains 4 workstation clusters running Fedora Linux, and a variety of network printing services are available for networked computers.

Office: MIT provides basic office space and library services for faculty and research staff.

Other: N/A

Special test equipment: None.

Equipment: None.

High performance computing availability: Not needed.