(Project Narrative)

Pixel matrices and other compositional analyses of interconnected systems

David I. Spivak

Submitted June 2016

Abstract

Frederick A. Leve—Program Officer—Dynamics and Control
(*Abstract may be publicly released*)

The new world of interconnected-everything brings new challenges to those who wish to understand it and keep society safe from unintended and unimagined consequences. With constant communication and feedback loops being the norm, the space of behaviors is too large to analyze by simulation alone. Today's nearly unlimited computational power must be used more wisely, so that our knowledge of a system can evolve along with the system itself. New mathematical techniques are needed to provide the algebraic formulas for *combining our insights*, just as we combine components, allowing us to anticipate the behavior of an assembled system. Category theory is the mathematics of combination and compositionality, so it is well-suited as a foundation for such work.

We propose to investigate compositional techniques for analyzing systems of all sorts. At the mathematical center of many disciplines, one needs to solve a system of simultaneous equations. As mundane, abstract, and worked-over as this may seem, a new elementary technique was recently discovered with the potential to change how we approach such problems. This technique is highly compositional—the solutions to subsystems can be combined to form a solution of the whole—and it emerged out of a similarly compositional approach to understanding the behavior of networked machines. Just as circuits can be combined to form computers, machines of all scales can be interconnected to form more complex machines. The common theme is compositionality: whether combining the constraints and requirements necessary to design a robot, or combining the equations that describe its function, our goal is to find analyses that are scalable and reusable, so that the knowledge we gain today is efficiently utilized in the networks of tomorrow.

Contents

Statement of Objectives

0 Statement of Objectives

The distinction between part and whole becomes blurry as one moves from systems to *systems of systems*; whether a system is a complete whole or a subsystem of some larger whole becomes a matter of perspective. Thus to understand complex systems, the ability to rapidly change perspective is becoming crucial. The desire for compositional systems must be met with a demand for *compositional analyses*, so that as systems are integrated, the independent analyses and certifications of the component systems can likewise be integrated, without loss, to provide analyses and certifications of the larger whole.

The objective of the proposed research is to provide a wide variety of mathematically-rigorous compositional analysis techniques, which can be applied to systems of all sorts. As a formal underpinning for this research, we will use category theory, for which compositionality is the core principle. Rather than focus on a particular system, such as the smart grid or the national air space, we consider systems of all sorts: from interconnected dynamical systems, to systems of nonlinear equations, to the systems of constraints that emerge in engineering design.

A primary interest is to investigate a new technique for solving systems of nonlinear equations. This technique is remarkably simple to describe: each equation is plotted as a matrix of pixels and the approximate solution set for the entire system can be computed using matrix multiplication and other standard matrix operations. The simplicity of this *pixel matrix* technique makes it ripe for investigation and innovation. It is easily parallelizable: subsystems can be solved independently, and the solutions compose. Unlike most mathematical techniques, it can be applied to raw data without requiring that it be fit to any sort of model. We will consider not only questions of computational complexity and accuracy of this approach, but more foundational questions as well.

Systems of equations often arise when variables are shared among interacting components of a given sort. For example, open dynamical systems—machines whose states evolve as they receive and send signals—can be interconnected to form larger-scale dynamical systems. Here the signals themselves act as variables shared between machines. If a given analysis of machines can be algebraically manipulated as the machines are interconnected, so that properties of the whole are derivable from those of the parts, we call it a compositional analysis. Steady states and bifurcation diagrams are examples of compositional analyses: knowing only the steady state behavior of component systems, one can derive the steady state behavior of an interconnected system. We will investigate other compositional analyses, such as reachability and control. We will also explore compositionality as it shows up in engineering design, to determine the sorts of analyses that lend themselves to large-scale team projects.

The common theme in all of this work is to formalize those conceptions of a system which admit algebraic rules for combination, as systems of any scale become the component parts of a larger-scale system. We aim to determine the sort of behavioral guarantees which, made individually on parts, formulaically combine to produce behavioral guarantees on the whole interconnected system. Such compositionality is crucial to predicting and auditing the behavior of complex systems.

Research effort

For the purpose of exposition, we divide the proposal into two parts. Although they are deeply analogous, the goals of the first are classical—a new technique for solving systems of equations and inequalities—whereas those of the second address a contemporary problem, namely compositionality in systems of systems. These two parts are discussed in Sections 1 and 2 respectively. The analogy between these two parts can be expressed in the language of category theory, though we keep our discussion of this fact to a minimum, especially in Section 1. Though the parts are separated in this proposal, they are really two parts of a larger whole, and any effort spent on one will likely yield dividends in the other.

1 Research Topic 1: Pixel matrices for solving systems of relations

1.1 Introduction

The need to compute solutions to systems of equations or inequalities is ubiquitous throughout mathematics, science, and engineering. A great deal of work is continually spent on improving the efficiency of linear systems solvers—both for dense and sparse systems [CW87; FSH04; GG08; DOB15]—and new algebro-geometric approaches are also being developed for solving systems of polynomial equations [GV88; CKY89; Stu02]. Less is known for systems of arbitrary continuous functions, and still less for systems involving inequalities or other relations [Bro65; Mar00]. Techniques for solving nonlinear systems are often highly technical and specific to the particular types of equations being solved. Moreover, most techniques are iterative and thus find *one* solution near a good initial guess, rather than finding all solutions to the system.

It would be useful to have a new *elementary technique*—say, one that can be understood by an undergraduate math major within three hours—for providing the approximate solution set, in its entirety, for arbitrary nonlinear systems. If it were to be faster, more accurate, more flexible, and more widely applicable than existing techniques, that would be even better.

We present a new technique, currently at a very early stage of development, with which to find the approximate solution set for arbitrary systems of relations (nonlinear equations, inequalities, etc.). It is elementary in the above sense, as it relies only on matrix arithmetic applied to what we will call *pixel matrices*. The Pixel Matrix (PM) technique appears to be more flexible and widely applicable than other techniques; however, it does have limitations, and it has not yet been compared to existing techniques in terms of speed and accuracy. The PM technique is based on category theory [Mac98; Awo10], which has been put forth as a potential foundation for applied mathematics [Bae13], and which has already found a number of applications throughout science and engineering [FGR03; BW90; CGR12; AC04; GSB12; Spi14]; we will not emphasize this aspect of pixel matrices, but it exists in the background.

1.2 Motivating example

Suppose we plot the two equations $x^2 = w$ and $w = 1 - y^2$ as graphs on a computer screen. The result for each equation, say the first one above, will be an array of pixels—some on and some off—that represents the set of (x, w)-points which satisfy the equation. Thus we can plot each equation as a *matrix of booleans*—True's and False's—representing its graph; say M for the

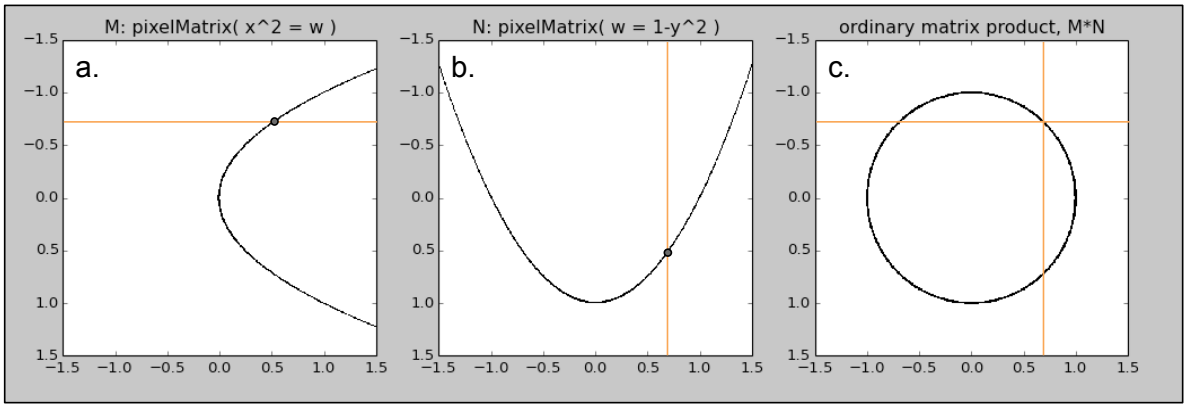


Figure 1: Parts a and b show plots of $x^2 = w$ and $w = 1 - y^2$, as pixel matrices, where each entry is a 0 or 1. The pixel is black or "on" if that entry in the matrix is 1, and white or "off" if that entry is 0. The graphs appear rotated 90° clockwise, in order to agree with matrix-style indexing, where the first coordinate is indexed downward and the second coordinate is indexed rightward, rather than ordinary Cartesian-style indexing, where the first coordinate indexed rightward and the second coordinate indexed upward. These matrices are multiplied, and the result is shown in part c. The horizontal and vertical lines in parts a and b respectively indicate an example row and column whose dot product is 1, hence the pixel is on at their intersection point in part c. The fact that the result of matrix multiplication looks like a circle is not a coincidence; it is the graph of the simultaneous solution to the system given in parts a and b, which in this case can be rewritten to a single equation $x^2 = 1 - y^2$.

first equation and N for the second. What happens if we multiply the two matrices together? It turns out that the resulting pixel array MN represents the (x, y)-pairs that simultaneously solve the two equations in the system. In other words, ordinary matrix multiplication returns a circle, $x^2 + y^2 = 1$; see Figure 1.

This simple fact about pixel matrix multiplication MN has several cousins, including matrix tensor product $M \otimes N$ and matrix trace $\mathrm{Tr}(M)$, which together form the basis of the pixel matrix approach to solving systems.

The PM technique, which we will briefly discuss below, appears to be unknown, though it is easily explained. There is nothing magical in this approach; it is simply a way to organize substitution and existential quantification into a matrix arithmetic formalism. This is analogous to the sense in which Gaussian elimination simply organizes the computation necessary for solving linear systems of equations, and how matrix multiplication simply organizes the computation necessary for composing linear transformations between vector spaces with chosen bases. Matrices are quite diverse in their applications—for example they are used to study directed graphs, Markov processes, finite metric spaces, linear transformations, etc.—we simply add another member to the list: approximately solving *nonlinear* systems.

1.3 Systems of relations, wiring diagrams, and matrices

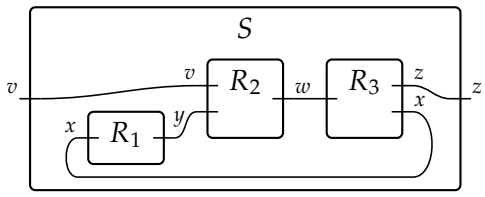
Suppose given a system of n equations, inequalities, or other relations $R_1, \ldots, R_n$, each of which includes some subset of k variables $x_1, \ldots, x_k$. Often, not every relation will use

every variable—that is, the system is partially decomposable [Sim91]—and only some of the variables have relevant values, the others being considered latent or *internal* variables [WP13]. (In Section 1.2, x and y were relevant and w was internal.)

For example, suppose we want to find all (v, z) pairs for which the following system of three relations R_1, R_2, R_3 has a solution:

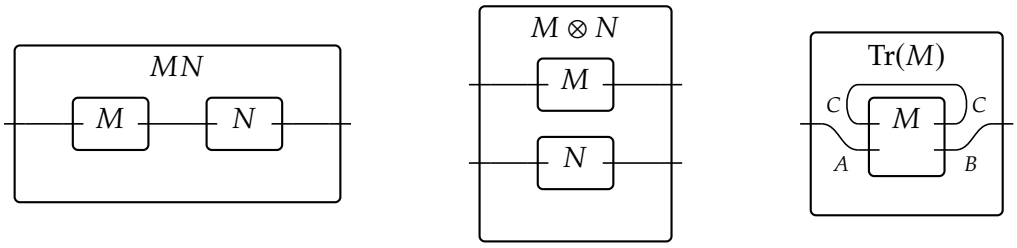
$$\begin{aligned} R_1: &\quad x^2 + 3|x - y| - 5 = 0 \\ R_2: &\quad y^2 v^3 - w^5 \leq 0 \\ R_3: &\quad \exists b.\ \cos(b + zx) - w^2 = 0 \end{aligned} \tag{1}$$

Each of the three relations can be plotted, using some range, tolerance, and level of pixel refinement, and the result is what we have been calling a *pixel matrix*; at the risk of confusion, we also denote the corresponding boolean matrix approximations by R_1, R_2, R_3. One then forms an associated *wiring diagram*, which shows how variables are shared among the relations (e.g. variable y is shared by R_1 and R_2):[1]



(2)

The wires that emerge from the outer box are v and z, as these were the relevant variables from the original query. The wiring diagram (1) tells us what arithmetic operations to perform on the corresponding pixel matrices [Spi15]. Serial composition corresponds to matrix multiplication MN, parallel composition corresponds to matrix tensor product (forming block matrices) $M \otimes N$, and feedback corresponds to partial trace $\mathrm{Tr}(M)$:



(3)

Thus the diagram in (2) represents the formula $S = \mathrm{Tr}\big((R_1 \otimes I_v)R_2R_3\big)$, where I_v is an identity matrix. To recapitulate, in order to solve system (1), we begin by plotting the three relations R_1, R_2, R_3 and considering the plots as matrices of booleans. We then tensor matrix R_1 by an identity matrix I_v, and multiply the result with R_2 and R_3. This will produce a block matrix with square $(x \times x)$-blocks; taking the trace of each block will result in a matrix S of booleans, which we can plot as an array of pixels. It shows the set of (v, z) pairs for which a simultaneous solution exists. Although this procedure is quite simple—involving only matrix arithmetic—it produces an approximate solution set for the fairly complex system (1).

[1] The orientations of the boxes are not canonically associated to the system: each box can be oriented in several ways, but all orientations give the same final result; see Section 1.5.4.

1.4 Potential advantages of the approach

1.4.1 Simplicity and support

The simplicity of the basic approach—using pixel matrix arithmetic to solve general systems—means that there is low barrier to entry for both researchers and users. There are many open questions, and mathematicians and software engineers of all stripes may be able to contribute new ideas that extend the basic seed presented above.

The technique relies on matrix arithmetic and multidimensional array handling, which are the "bread and butter" of any computational software system, such as MATLAB or Julia. Programming the PM technique on top of such a numerical computing environment could be given as a final undergraduate project; for example, the current (unoptimized) implementation involves under 500 lines of code. In other words, solving systems of relations using pixel matrices essentially "runs for free" in these environments. Improvements in the speed of matrix and sparse matrix arithmetic, as well as the speed of graphics processing units (GPUs), which are adept at handling the necessary sorts of vector arithmetic, will produce immediate improvements in the speed of the PM approach to solving nonlinear systems.

1.4.2 Propagation of true negatives

As explained above, in the PM technique, one begins to solve a system of equation by plotting each equation as a pixel matrix, where each pixel represents a "tiny" d-dimensional rectangle. It would be best if a given pixel were to be on (true) if and only if the equation holds for *some point* inside the corresponding rectangle. However, even if this is true for the original plots, the results lose fidelity as the matrix operations are performed. In the limit, as the mesh sizes are decreased, the PM approximation approaches the true solution; however, at any finite stage there will very likely be some error.

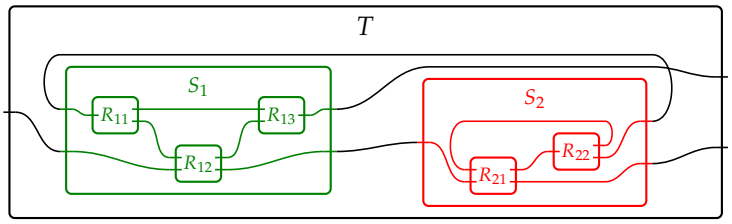
At least in certain cases it is possible to ensure that the plot of relation R includes *true negatives*, i.e. that if a pixel is off (false) then no point inside the rectangle satisfies the relation. If this holds, it is a theorem that there will be true negatives in the result of arbitrarily many matrix calculations, used to solve the system of relations. The upshot is that one can get a rough estimate of the solution set by pixelating with a very coarse mesh—the result of which will be very fast—and then refine it as desired inside only those pixels that are "on".

1.4.3 Entire solution set

When bounds for each variable are chosen, the PM approach approximates the entire solution set in the corresponding region. This is in contrast to iterative nonlinear systems solvers, which generally find only one solution and require a good initial guess [QS93]. Using the mesh refinement technique from Section 1.4.2, one could begin by approximating the solution set using the PM technique, and then choose any "on"-pixel with which to seed a more sophisticated nonlinear systems solver.

1.4.4 Naturally parallelized

To solve large systems of relations, it is advantageous to paralellize the solution algorithm. The wiring diagram formalism lends itself to parallelization. Any region of the wiring diagram can be "cordoned off", and the corresponding subsystem of relations can be solved independently. Then the results can be combined to produce the correct solution set for the whole system.[2] For example, in (4), the matrix operations needed to solve the five relations $R_{11}, R_{12}, R_{13}, R_{21}, R_{22}$ can be computed by first solving the first three to form relation S_1 and the last two to form S_2, according to the wiring subdiagrams shown here:



(4)

We now have a system of two relations S_1 and S_2, and applying a matrix multiplication and trace results in the total solution T. The computation for T proceeds from S_1 and S_2 without needing any information about the internal structure—nor the relations themselves—that went into computing the intermediate relations S_1 and S_2 from the given R's. This tends to reduce the computational complexity a great deal.

1.4.5 Solution counts and densities

As discussed above, pixel matrices have boolean entries, but the booleans can be replaced by the elements of any *semiring*, meaning a set S with elements $0, 1 \in S$ and operations $+, *\colon S \times S \to S$ satisfying well-known properties. For example, using the natural numbers $0, 1, 2, \ldots$ instead of the booleans, the PM approach will report the *number* of solutions, rather than just the *existence* of a solution. Values in the semiring $\mathbb{R}_+$ of nonnegative reals would instead encode solution *density*, encoding the expected number of solutions found within each pixel. Other semirings may have interesting semantics as well.

1.4.6 Real data, lost equations, and machine learning

The PM technique does not require that the relations or densities (see Section 1.4.5) come from equations or inequalities. A scientist may have raw data relating variables x and y and more raw data relating variables y and z. A standard approach is to fit smooth functions to these data sets and to then solve the system of simultaneous equations; however, curve-fitting requires a choice of model. There is often a trade-off between simplicity of the model and accuracy of the fit. Moreover, any error that does arise in this way will be an artifact of the choice of model, adding a confounding complication to the data analysis.

The PM technique offers an alternative—solving the system directly without curve fitting—which bypasses the above minefield completely. The pixel matrices for the data sets (including

[2] Results of this sort are proved category-theoretically; as an example, see [Spi15].

Gaussian error surrounding each data point if desired) can be multiplied directly. The ability to work with data directly, rather than first requiring a cleanup step, may prove to be the real value of pixel matrices.

Similarly, the notion of *concept* from computational learning theory is defined to be a subset $S \subseteq A$, where $A = A_1 \times \cdots \times A_n$ is a product of attribute spaces [KV94]. The characteristic function of such a subset is an A-shaped array of Booleans whose support is S; in other words, concepts are equivalent to pixel matrices. Thus it follows that composing concepts becomes a matter of matrix arithmetic. One may perhaps "solve" for concepts: given a wiring diagram as well as a composite concept and all but one interior concept, one could solve for the remaining interior concept. For example, one might use something like (block) singular value decompositions of the given interior concepts to find a best fit for the remainder.

Pixel matrices may also offer pedagogical clarification or suggest new directions in machine learning and artificial neural networks. Typically, a multilayer neural network involves a variety of hidden layers, each of which consists of a matrix multiplication step followed by an activation function [EMA12]. Both of these have analogies in terms of pixel matrices—matrix multiplication corresponds to a certain way that variables are shared, and application of a activation function, such as the rectifier $x \mapsto \max(0, x)$, corresponds to something like a "green screen" for pixel matrices: one discards all data of a certain sort. Other matrix operations may also have correlates in this context; for example the partial trace operation may be employable to study the effects of feedback in recurrent neural networks.

1.5 Research directions

While the basic idea is simple and can be put directly to use, there is work to be done to evaluate and improve the pixel matrix technique. For example, there are questions regarding computational complexity, accuracy, and sampling procedures, as well as how to most efficiently orient and group the relations in a system. There is also an opportunity to find innovative applications and extensions to the PM technique itself. We briefly describe each of these research directions.

1.5.1 Sampling

Pixel matrices allow one to combine a system of plotted relations to obtain a plot of their simultaneous solutions, but plotting each individual relation remains difficult. The current PM implementation does this by sampling, but the sampling method could easily be improved, perhaps using something like interval arithmetic [Fat92].

1.5.2 Accuracy

Each pixelated relation may be assigned an *error bound*, e.g. the maximum distance (say, in the L^∞-metric) from a false positive to the nearest true positive. Even for a perfectly plotted relation, this will generally be nonzero because pixels have nonzero radius. One can ask: given an error bound on each plotted relation R_i, and given a wiring diagram as in (2) describing a sharing of variables between the relations, what is the error bound on the result? Another

avenue would be to use probabilistic—rather than boolean—pixels to mitigate the cost of under-sampling.

1.5.3 Computational complexity

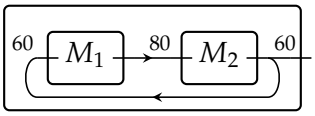
It is important to understand the computational complexity of the PM technique, in various cases. A common special case is when the relations involved are all equations—as opposed to inequalities or arbitrary relations—in which case each pixel matrix is sparse (codimension 1), greatly reducing the computational complexity of the matrix calculations. For example, plotting $x^2 + y^2 = 1$ is much faster by using sparse matrix multiplication for functions $x^2 = w$ and $w = 1 - y^2$, as in Section 1.2, than by using pure brute force sampling of the original equation.

For general relations, it appears that we can bound the complexity by clustering the relations in terms of links, and then exponentiating the maximal number of links over all the clusters. Proving and/or sharpening such bounds is a high priority for communicating the value of the PM approach. Even if it proves to be slower or not substantially faster than other techniques (which naively one may expect, given the exponential blowup in the number of local variables), it is worth exploring whether one can extract compositional data from these matrices, which still offer some sort of information about the solution set while being more computationally efficient.

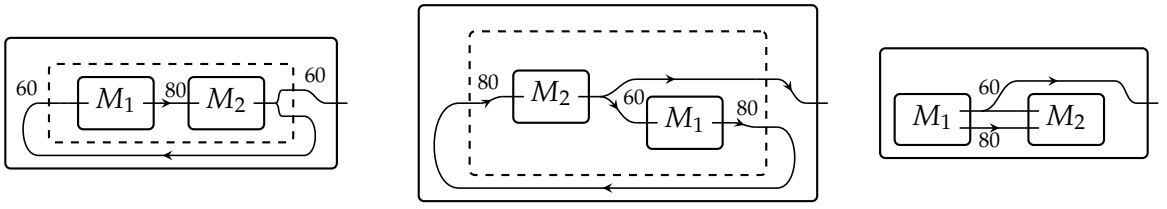
1.5.4 Orientations and grouping

The wiring diagram for a system is naturally unoriented; that is, the boxes natively have no notion of inputs and outputs, only *ports*.[3] Dropping orientation, a box corresponds to an n-ary array, which we organize into a block matrix in order to utilize matrix arithmetic. Understanding the most efficient way to choose orientations is a part of the methodology that remains unexplored.

For example, consider the wiring diagram below representing an operation on two 60×80 pixel arrays, M_1 and M_2:



The result will be a 60-entry pixel vector, but there are several equivalent ways to calculate it. Three options are shown diagrammatically below:



[3] Though note that orientation does make sense in many applications, e.g. dynamical systems (as in Section 2), where one does have a notion of input and output variables.

Each of these corresponds to a different sequence of matrix operations: multiplication, adding columns of 0's, tracing blocks, etc., and the computational complexities can vary greatly. Deciding the most efficient way to group the system is an important open question.

1.5.5 Categorical structure

Because pixel matrices are always approximations to the "real" plot (a matrix whose dimensions are uncountably infinite), they form a *domain* in the sense of Dana Scott [Sco70]. It is important to explore the categorical structure that arises from putting these domains together to solve systems.

There is much more categorical structure to be understood here. For example, it is virtually cost-free, computationally speaking, to change a variable by an invertible affine transformation. That is, given a plot of solutions where $x^2 + y^2 = 1$, for $x, y \in [-1, 1]$, precisely the same plot represents solutions to the equation $(5x)^2 + (2y - 5)^2 = 1$, where $x \in [-0.2, 0.2]$ and $y \in [2, 3]$. Other such transformations (e.g. using order-preserving maps to change the mesh-size) are similarly cost-free, and it would be useful to neatly arrange the cost-free operations into a category-theoretic structure.

Each pixelation is in fact an idempotent monad, or closure operation, on the monoidal category of relations, and the pixelated plots are the algebras for such monads. This may or may not be a useful perspective, but we include it here in order to point to the broad range of what should be considered further. Articulating categorical structures often leads to a deeper understanding, but it can also provide new ways of analyzing solutions without actually computing them. This depends on finding new compositional analysis techniques, as discussed below in Section 2.3. In fact, we will see in Section 2 that the PM technique itself was originally discovered as a compositional analysis of dynamical systems.

1.5.6 Innovations

There appears to be a great deal of room for finding innovative applications and extensions to the PM technique. The idea is simple enough that it can be explained easily and a prototype can be programmed quite rapidly. It follows that finding new applications areas should not be challenging, and the exemplary application domains should be catalogued. Innovations to the technique are similarly quite likely. There are many directions from which substantial improvements could be forthcoming, given the resources to explore them.

2 Research Topic 2: Systems of interconnected machines

2.1 Introduction

The pixel matrix technique, discussed in Section 1, stemmed from a similar technique for computing the steady states of coupled dynamical systems. Dynamical systems—which we think of as *machines* that take in time-varying input, change their state accordingly, and produce time-varying output—can be wired together into systems. That is, one machine feeds its output to another machine as input, and all together they produce an interconnected "net"

machine. Determining what can be known about this net machine, given knowledge of the components that compose it, is becoming crucial in the modern world.

Cyber-physical systems, the Internet of Things (IoT), smart cities, etc., provide examples of an emerging technological paradigm where the mathematics of interconnection should play a major role. Hardware and software components, at scales from micro-processors to self-driving cars, built by different manufacturers for different purposes, will be put together in a myriad of novel arrangements. The interaction between these components will be highly complex—multiple feedback loops being the norm—and it is important to be able to quickly put bounds on the outcome of such interactions.

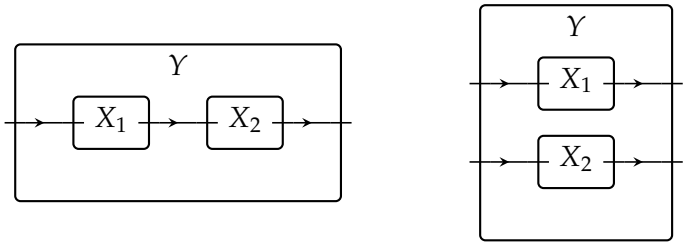
As an example of what one might desire, consider the following analogy with pharmaceutical drugs. The federal Food and Drug Administration requires that each drug be sold with a package insert, or *label*, providing information that explains how to use the product, what the active ingredients are, what types of symptoms may indicate an adverse event, what other drugs to avoid, etc. The purpose of these drug labels is to provide patients with an understanding of how they can use the drug in combination with other products and lifestyle choices (driving, sun-exposure, etc.) to minimize harmful side-effects. In an analogous way, casual and professional users of cyber-physical systems need to understand what to expect when they linke these systems in novel combinations with one another and use them in various environments.

In this section, we will discuss what sorts of "labels" might be appropriate when putting machines together in combination. For example, if a video camera is aimed at the screen to which it sends signals, how are the consequences similar to or different than when a microphone is placed near the loudspeaker to which it sends signals? Each involves a feedback loop, but only one is destructive. Could this fact be calculated in advance, without requiring the input of human experience and expertise?

2.2 Wiring diagrams and open dynamical systems

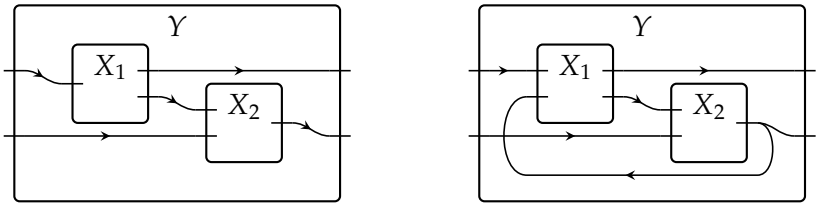
We will seek mathematical attributes that one can observe in component machines, with the rule that regardless of how the component machines may be wired together, one can formulaically decide the resulting attributes of the total, "net" machine. Certainly not all atributes have this property; we have been calling those that do "compositional", and we will describe this notion in more detail in Section 2.3.

Each component machine has some interface X, consisting of input and output ports; we say that the machine *inhabits* the interface. These interfaces can be be put together in series or parallel



(5)

or in more complex combinations, possibly with feedback and splitting wires



(6)

An *open dynamical system* is a type of machine that has a set or space of states and a rule for how its state changes in time. Signals passed to the system through its input ports influence how the state changes. An output signal is generated as a function of the state, and it is then passed through an output port to serve as an input to a neighboring system.

Interconnected dynamical systems have been studied for years, e.g. often under the name *coupled cell networks*, by Golubitsky, Stewart, and collaborators [EG93; DGS96a; DGS96b; SGP03; GS06; SP07], as well as many others [GH99; Agu+11; DL15; VSL15]. We can study many sorts of dynamical systems, including discrete and continuous models, but for the purposes of this proposal, we speak about dynamical systems in the abstract.

Category-theoretically, we consider dynamical systems as forming a certain *algebra* on the *operad of wiring diagrams* [Spi15]. That is, to each interface X, we associate the set $\mathsf{DS}(X)$ of all open dynamical systems D which have that interface, i.e. which accept input signals and produce output signals of the specified type. To each wiring diagram $\varphi\colon X_1, \dots, X_n \to Y$, we associate a formula $\mathsf{DS}(\varphi)\colon \mathsf{DS}(X_1) \times \cdots \times \mathsf{DS}(X_n) \to \mathsf{DS}(Y)$, which takes a tuple $(D_1, \dots, D_n)$ of dynamical systems—where dynamical system D_i inhabits inner box X_i—and produces a dynamical system $E = \mathsf{DS}(\varphi)(D_1, \dots, D_n) \in \mathsf{DS}(Y)$ inhabiting the outer box Y. These formulas must be *functorial*, in the sense of category theory, to ensure that nesting wiring diagrams within wiring diagrams is a coherent operation.

2.3 Compositional analysis of machines

Compositional analysis of open dynamical systems is an important concept in applications. An analysis is *compositional* if, rather than be applied to a whole composite system, it can be applied to the subsystems independently and the results compiled to give the global result.

As a simple example, averaging is *not* a compositional analysis. Indeed, suppose we want to compute the average age of US city-dwellers. If there are n cities $C_1, \dots, C_n$, with average ages $c_1, \dots, c_n$, the overall average age is *not* merely the average of $c_1, \dots, c_n$, because each city presumably has a different number of people. Knowing just the averages c_i is almost useless information in that it cannot be used in further calculations; one might say such analyses are truncating, as opposed to compositional. In contrast, if one knows not only the average age c_i but also the population size p_i for each city, then one can calculate *both* the overall average *and* the overall population size. Thus the attribute pair (population size, average age) forms a compositional analysis of US cities.

Compositional analyses are robust to redesign. Large-scale systems are built from many component systems, each of which has many parts and subparts, down to micro-scales. Improvements can arise from reconfiguring these parts at any level of a hierarchy (which can

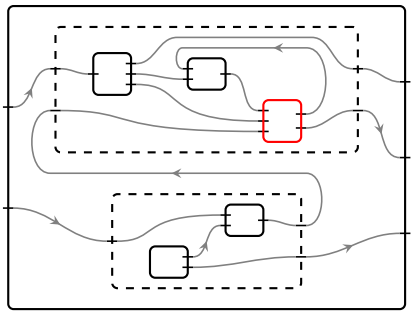
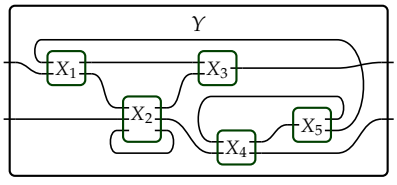


Figure 2: Compositional analyses facilitate the computation of global invariants, under rearrangement and replacement of internal components.

itself be re-conceived). As design changes become more frequent, it is important for analysis techniques to be compositional so that prior work is not lost as systems evolve.

We propose to search for a wide range of compositional analyses to describe machines of all sorts. One example of such a result is that *steady states* constitute a compositional analysis [Spi15]. This result is quite similar to the pixel matrix technique discussed in Section 1; indeed the latter was an extension of the former. The basic idea is to associate to every open dynamical system X, inhabiting a box $A \to B$, a matrix M of steady states. The rows of matrix M correspond to the possible input signals $a \in A$ and the columns represent possible output signals $b \in B$. The entry in $M_{a,b}$ is the number of states in X which are fixed by input a and which output b. We call M the *steady state matrix* for the dynamical system X. If the system has only a finite number of possible input and output values, M will be a standard (finite-size) matrix; however, when A and B are Euclidean spaces, e.g. $A = B = \mathbb{R}$, the steady state matrix is quite similar to what is generally known as a *bifurcation diagram*.

The interesting result is that as dynamical systems are composed in series, parallel, or with feedback, the steady state matrices are respectively multiplied, tensored, or traced (again analogous with pixel matrices; see Section 1.3). In fact, any wiring diagram corresponds to a combination of matrix arithmetic operations: matrices independently form an algebra on the operad of wiring diagrams. When these formulas are applied to the steady state matrices of component systems, the result is the steady state matrix of the net, interconnected system.



(7)

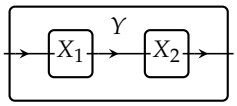
In other words, steady state matrices and bifurcation diagrams constitute compositional analyses.

2.4 Compositional behavior contracts

In our view, one of the most important compositional analyses for the coming age is what we call *compositional behavior contracts*.[4] Analogous to guarantees made by prescription drug labels, as discussed in Section 2.1, producers of machines at all scales could offer some guarantees about how their machine will behave given the receipt of certain input signals. For example, they could provide data about steady states behavior, or about reachability or controllability.

In a world where various machines will be interconnected in ways unforeseen by their designers, it is important that their behavioral guarantees are compositional. Only then will an engineer, who interconnects these machines in a novel way, be able to in turn produce a guarantee about its overall behavior.

As a simple example, consider two synchronized discrete-time machines X_1, X_2 wired in series to form a machine Y:



Suppose that machine X_1 satisfies the contract "Whenever I receive three True's in a row, I will output a False within five seconds", and that machine X_2 satisfies the contract "Whenever I receive a False, I will output a True within two seconds." Then the designer of Y, without knowing the internals of X_1 and X_2 can still guarantee that Y satisfies the contract "Whenever I receive two True's in a row, I will output a True within seven seconds."

Compositional contracts would greatly facilitate safety and failure analysis. Whenever an interconnected system violates its contract, we would be mathematically assured that either there is a connection problem—i.e. components are not wired together as advertised[5]—or that one of the system elements has violated its contract. We can then recursively dive down to precisely determine which actor violated its contract and hence is responsible for the outwardly-visible contract violation.

2.5 Dynamic reconfiguration in mode-dependent networks

Analysis becomes highly complex when the signals that arrive at individual components in a system cause a *reconfiguring of the interconnection pattern itself*, i.e. a change in the network topology. This takes place constantly in the world of human business relationships (e.g. one company can engage or disengage the services of another), but it is just as ubiquitous in computer networks. A switch is a component that takes in signals and uses them to change the

[4] Composing financial contracts has been previously considered in terms of functional programming (a close cousin of category theory) [JES00]. Although such work is inspirational, the scope that we propose here is quite a bit broader, and the mathematical formalism is more explicit.

[5] Unexpected interactions between component systems is a sort of connection problem, because in such cases the wiring diagram has failed to express all the communication that exists between the subsystems. For example, in a physical system, vibrating parts can often cause resonance that damages other components. This interaction can be regarded as an unexpected *signal* from one component to another, which should have been captured in the wiring diagram. Although we do not discuss it here, it is important to have many levels of analysis—logical, functional, operational, physical, etc.—and these analyses must be comparable and integrable. Category theory is well-tuned to this sort of comparison and integration of analyses.

topology of the network. A communication network is also reconfigured when sensors move or when smartphone users move between wireless access points. Similarly, online learning by neural networks requires continual updating of weight matrices in order to accommodate changing observations [CSC06].

Mode-dependent networks of open dynamical systems have the property that changes in the internal states of each component lead to changes in the topology of the network that connects them. In [ST15], it was shown that mode-dependent networks also form an operad, meaning they can be nested hierarchically. Each component has a *communicative mode*, which is a function of its current internal state, and the tuple of all communicative modes determines the changing shape of the wiring diagram that interconnects them.

It would be useful to find compositional analyses for mode-dependent systems. For example, one could consider a set of components to be autonomous to the extent that their set of communicative modes determines their interconnection pattern. Such autonomy may not occur when a powerful authority is present: who communicates with who is to some extent determined by the authority. It would be interesting to investigate whether the autonomy of components can be measured and made compositional.

Other potential compositional invariants may include measures of the extent to which reconfiguration is taking place in subsystems. That is, if we can quantify the degree of reconfiguration for each subsystem, as well as for the subsystems in the larger system, it should be possible (given the right formalism) to calculate the degree of reconfiguration in the whole system. Issues like this one are related to the predictability and auditability of a system's outward behavior.

While the mode-dependent framework describes dynamically-changing networks of systems—and the hierarchical nesting thereof—it may not adequately articulate issues relevant to management and administration of resources [Sim65]. For example, it does not address the hierarchies of time-scales necessary for planning, nor does it address notions of values (what a system is aiming to achieve), decision-making, etc. While we are not wedded to the precise definition of mode-dependent networks described in [ST15], we are committed to the operadic formalism, in which compositionality, and the relationship between part and whole, are paramount. A philosophical approach that may be worth exploring is that of holonic networks [KS69; Mel09].

2.6 Designing complex systems

Another place where we find networks is in the engineering design process, where the requirements of one component must be met by the functionality of another [FKS95; EJ96]. For example, when constructing a simple mobile robot, one must design the chassis and the motor [Cen16a]. In order for the chassis to attain a given velocity with given extra payload capacity, it requires a certain torque and speed from the motor. To attain a given torque and speed, the motor certainly requires a certain voltage and current. But more interestingly, a better motor also weighs more. This creates a feedback loop: the chassis must carry the extra weight of the motor, which in turn delivers more power to the chassis.

In Censi's theory of *co-design*, design problems are, to the outside world, relationships

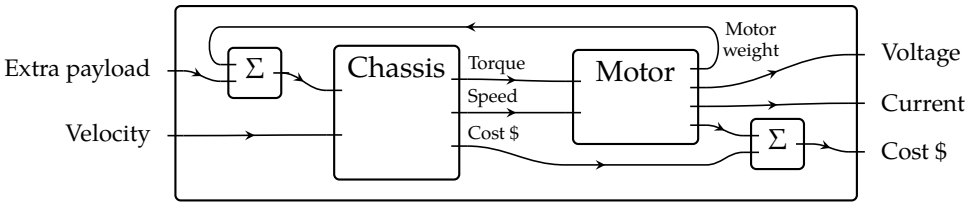


Figure 3: A co-design problem, adapted from [Cen16b].

between functionality provided and resources required. However, design problems may be broken down into interacting sub-problems, and these interactions are captured in terms of wiring diagrams, as shown in Figure 3. In other words, co-design problems form another algebra on the operad of wiring diagrams. The mathematical theory of co-design developed by Censi in [Cen16a; Cen16b] is ripe for category-theoretic interpretation. Indeed Censi writes, "Category theory is the right formalism: functionality and resources are the objects in a category and the design problems are the morphisms of the category." A design problem can be formulated as a monotonic map from a poset of "functionality" to the downward-closed subsets of a "resource requirements" poset; such a map is also known as a Boolean-enriched profunctor.

Once given a system of co-design problems, Censi's algorithm produces a system of discrete dynamical systems, which take input anti-chains (Pareto optimal subsets) to output anti-chains. The optimal result of the co-design problem is a fixed point, or steady state, of the dynamical system. It would be worthwhile to construct a category-theoretic formalism of this entire approach, from co-design problem to dynamical system, to steady states, all using compositional mappings, as discussed in Section 2.3. Censi and the PI hope to produce such formal connections in the coming years.

References

[AC04] Samson Abramsky and Bob Coecke. "A categorical semantics of quantum protocols". In: *Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on*. IEEE. 2004, pp. 415–425.

[Agu+11] M Aguiar, Peter Ashwin, A Dias, and Michael Field. "Dynamics of coupled cell networks: synchrony, heteroclinic cycles and inflation". In: *Journal of nonlinear science* 21.2 (2011), pp. 271–323.

[Awo10] Steve Awodey. *Category theory*. Second. Vol. 52. Oxford Logic Guides. Oxford University Press, Oxford, 2010, pp. xvi+311. ISBN: 978-0-19-923718-0.

[Bae13] John Baez. *The foundations of applied mathematics*. 2013. URL: http://math.ucr.edu/home/baez/irvine/irvine.pdf.

[Bro65] Charles G Broyden. "A class of methods for solving nonlinear simultaneous equations". In: *Mathematics of computation* 19.92 (1965), pp. 577–593.

[BW90] Michael Barr and Charles Wells. *Category theory for computing science*. Vol. 49. Prentice Hall New York, 1990.

[Cen16a] Andrea Censi. "A Class of Co-Design Problems with Cyclic Constraints and Their Solution". In: *Robotics and Automation Letters* (Feb. 2016). To appear. URL: http://MCDP.mit.edu/.

[Cen16b] Andrea Censi. *A Mathematical Theory of Co-Design*. Tech. rep. Laboratory for Information and Decision Systems/MIT, Jan. 2016.

[CGR12] Justin Curry, Robert Ghrist, and Michael Robinson. "Euler calculus with applications to signals and sensing". In: *Proceedings of Symposia in Applied Mathematics*. Vol. 70. 2012, pp. 75–146.

[CKY89] John F Canny, Erich Kaltofen, and Lakshman Yagati. "Solving systems of nonlinear polynomial equations faster". In: *Proceedings of the ACM-SIGSAM 1989 international symposium on Symbolic and algebraic computation*. ACM. 1989, pp. 121–128.

[CSC06] Min Chee Choy, Dipti Srinivasan, and Ruey Long Cheu. "Neural networks for continuous online learning and control". In: *Neural Networks, IEEE Transactions on* 17.6 (2006), pp. 1511–1531.

[CW87] Don Coppersmith and Shmuel Winograd. "Matrix multiplication via arithmetic progressions". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM. 1987, pp. 1–6.

[DGS96a] B. Dionne, M. Golubitsky, and I. Stewart. "Coupled cells with internal symmetry Part I: wreath products". In: *Nonlinearity* 9 (1996).

[DGS96b] B. Dionne, M. Golubitsky, and I. Stewart. "Coupled cells with internal symmetry Part II: direct products". In: *Nonlinearity* 9 (1996).

[DL15] L. DeVille and E. Lerman. "Dynamics on networks of manifolds". In: *SIGMA Symmetry Integrability Geom. Methods Appl.* 11 (2015).

[DOB15] Steven Dalton, Luke Olson, and Nathan Bell. "Optimizing Sparse Matrix-Matrix Multiplication for the GPU". In: *ACM Transactions on Mathematical Software (TOMS)* 41.4 (2015), p. 25.

[EG93] I.R. Epstein and M. Golubitsky. "Symmetric patterns in linear arrays of coupled cells". In: *Chaos* 3.1 (1993).

[EJ96] Atila Ertas and Jesse C Jones. *The engineering design process*. Wiley New York, 1996.

[EMA12] Stephen W Ellacott, John C Mason, and Iain J Anderson. *Mathematics of neural networks: models, algorithms and applications*. Vol. 8. Springer Science & Business Media, 2012.

[Fat92] Richard Fateman. "Honest plotting, global extrema, and interval arithmetic". In: *Papers from the international symposium on Symbolic and algebraic computation*. ACM. 1992, pp. 216–223.

[FGR03] Michael Fleming, Ryan Gunther, and Robert Rosebrugh. "A database of categories". In: *Journal of Symbolic Computation* 35.2 (2003), pp. 127–135. ISSN: 0747-7171. DOI: 10.1016/S0747-7171(02)00104-9.

[FKS95] Susan Finger, Suresh Konda, and Eswaran Subrahmanian. "Concurrent design happens at the interfaces". In: *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* 9.02 (1995), pp. 89–99.

[FSH04] Kayvon Fatahalian, Jeremy Sugerman, and Pat Hanrahan. "Understanding the efficiency of GPU algorithms for matrix-matrix multiplication". In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. ACM. 2004, pp. 133–137.

[GG08] Kazushige Goto and Robert A Geijn. "Anatomy of high-performance matrix multiplication". In: *ACM Transactions on Mathematical Software (TOMS)* 34.3 (2008), p. 12.

[GH99] Mario Galarreta and Shaul Hestrin. "A network of fast-spiking cells in the neocortex connected by electrical synapses". In: *Nature* 402.6757 (1999), pp. 72–75.

[GS06] M. Golubitsky and I. Stewart. "Nonlinear dynamics of networks: the groupoid formalism". In: *Bull. Amer. Math. Soc. (N.S.)* 43.3 (2006).

[GSB12] Tristan Giesa, David I Spivak, and Markus J Buehler. "Category theory based solution for the building block replacement problem in materials design". In: *Advanced Engineering Materials* 14.9 (2012), pp. 810–817.

[GV88] D Yu Grigor'ev and NN Vorobjov. "Solving systems of polynomial inequalities in subexponential time". In: *Journal of symbolic computation* 5.1 (1988), pp. 37–64.

[JES00] Simon Peyton Jones, Jean-Marc Eber, and Julian Seward. "Composing contracts: an adventure in financial engineering(functional pearl)". In: *ACM SIGPLAN NOTICES* 35.9 (2000), pp. 280–292.

[KS69] "Beyond Reductionism (The Alpbach Symposium)". In: ed. by Arthur Koestler and J.R. Smythies. Boston: Beacon Press, 1969. Chap. Beyond atomism and holism – the concept of the holon.

[KV94] Michael J Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[Mac98] Saunders Mac Lane. *Categories for the working mathematician*. 2nd ed. Graduate Texts in Mathematics 5. New York: Springer-Verlag, 1998. ISBN: 0-387-98403-8.

[Mar00] José Mario Martınez. "Practical quasi-Newton methods for solving nonlinear systems". In: *Journal of Computational and Applied Mathematics* 124.1 (2000), pp. 97–121.

[Mel09] Piero Mella. *The Holonic Revolution Holons, Holarchies and Holonic Networks, The Ghost in the Production Machine*. 2009.

[QS93] Liqun Qi and Jie Sun. "A nonsmooth version of Newton's method". In: *Mathematical programming* 58.1-3 (1993), pp. 353–367.

[Sco70] Dana Scott. *Outline of a mathematical theory of computation*. Oxford University Computing Laboratory, Programming Research Group Oxford, UK, 1970.

[SGP03] I. Stewart, M. Golubitsky, and M. Pivato. "Symmetry groupoids and patterns of synchrony in coupled cell networks". In: *SIAM J. Appl. Dynam. Sys.* 2.4 (2003).

[Sim65] Herbert Alexander Simon. *Administrative behavior*. Vol. 4. Cambridge Univ Press, 1965.

[Sim91] Herbert A Simon. *The architecture of complexity*. Springer, 1991.

[SP07] Ian Stewart and Martyn Parker. "Periodic dynamics of coupled cell networks I: rigid patterns of synchrony and phase relations". In: *Dynamical Systems* 22.4 (2007), pp. 389–450.

[Spi14] David I Spivak. *Category theory for the sciences*. MIT Press, 2014.

[Spi15] David I Spivak. "The steady states of coupled dynamical systems compose according to matrix arithmetic". In: *arXiv preprint: 1512.00802* (2015).

[ST15] David I. Spivak and Joshua Tan. "Nesting of dynamical systems and mode-dependent networks". In: *arXiv preprint: 1502.07380* (2015).

[Stu02] Bernd Sturmfels. *Solving systems of polynomial equations*. Vol. 97. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI, 2002, pp. viii+152. ISBN: 0-8218-3251-4. DOI: 10.1090/cbms/097. URL: http://dx.doi.org/10.1090/cbms/097.

[VSL15] Dmitry Vagner, David I. Spivak, and Eugene Lerman. "Algebras of open dynamical systems on the operad of wiring diagrams". In: *Theory Appl. Categ.* 30 (2015), Paper No. 51, 1793–1822. ISSN: 1201-561X.

[WP13] Jan C Willems and Jan W Polderman. *Introduction to mathematical systems theory: a behavioral approach*. Vol. 26. Springer Science & Business Media, 2013.

Assurances

A Assurances

A.1 Environmental impacts

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

A.2 Principle investigator (PI) time

The PI will spend 33% of his effort in year 1, and 45% of his effort in years 2–5. He also plans to hire postdocs and student research assistants to work on the project throughout its duration. There are no plans at this time to support graduate students on the project.

Current Projects and Pending Proposals The PI has three current projects—totaling 9.5 calendar months per year of support for the PI—and no pending proposals at this time. The current projects are

1. A five-year (2013–2018) AFOSR grant, titled "Categorical approach to agent interaction". This grant focuses on communication between agents, including information integration, communication protocols, and database queries. The PI is currently devoting 9 calendar months per year to this project.
2. A three-year (2014–2017) NASA grant, titled "Category-theoretic approaches for the analysis of distributed systems". Although the title sounds similar to the present one, the work is quite different. It is a 6.2 (applied research) program, and MIT is a subcontractor of Honeywell International Inc. The goal is to understand the National Air Space (NAS) as a distributed system; in particular, we focus on ensuring the safe separation (collision avoidance) of aircrafts. While some of the ideas for the current proposal stemmed from the NASA project, the current work will consist of fundamental math research, rather than NAS-specific applications. The PI is currently devoting 0.5 calendar months per year to this project.
3. A 1-year (2016) NSF grant, titled "Solving Information-Integration Problems Using Category Theory". This is an I-Corps Team program, for commercialization of database integration technology. The PI is no longer contributing, i.e. is devoting 0 calendar months per year to this project.

A.3 Facilities

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

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.

A.4 Special test equipment

None.

A.5 Equipment

None.

A.6 High performance computing availability

Not needed.