Invariants of machines in wiring diagrams SPUR Final Paper, Summer 2015

Anderson Wang
Mentor Lyuboslav Panchev
Project suggested by David Spivak

July 30, 2015

Abstract

Spivak (2013) introduced wiring diagrams as a way to formalize objects or processes which are built out of smaller parts, such as individual neurons interacting in the brain, or parts of a computer program working together to perform a task. These "parts" can be thought of as elements of a category of state machines, with smaller state machines being wired together to make larger machines. In this paper, we study the problem of finding invariants of these state machines which both preserve equivalence and satisfy a functoriality property. Although we have not fully answered the question of whether there exists any such nontrivial invariants, we will give some methods which could prove useful and provide some concrete invariants that work for various classes of state machines.

1 Background

1.1 Overview

As stated in the abstract, the goal of this project is to investigate possible invariants of state machines in wiring diagrams. We will first define wiring diagrams and state machines in formal terms in order to explain what precisely is an invariant of a state machine. We will then present our results, which include details of various attempts to find these invariants and several examples which work in some ways but fail in others.

Most of the material in the Background section is adapted from [1].

1.2 Boxes and wiring diagrams

Wiring diagrams consist of boxes connected by wires. A *box* is defined by some number of input wires and some number of output wires; Figure 1 shows a box X with 2 inputs and 2 outputs.

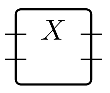

Figure 1: A box

In a box, each input and output can take values in any given set. For example, the two inputs of X might be a boolean and an integer, and the two outputs might be a positive integer and an element of $\{1, 2, 3\}$.

Definition 1.1. A *wiring diagram* $\phi : X \to Y$ between two boxes X and Y consists of putting box X inside box Y and connecting the inputs and outputs of X and Y with directed wires, with the following conditions:

1. Each output of Y must come from exactly one output of X.
2. Each input of X must come from either exactly one input of Y or output of X.

If a wire goes from an output of X to an input of X, we call it an *internal wire.* Let $\text{int}(\phi)$ denote the set of internal wires of ϕ.

Figure 2 shows a possible wiring diagram from the box X given above to a box Y also with 2 inputs and 2 outputs.

A few remarks:

- In general, the inputs of Y and outputs of X do not need to be connected to anything else, as seen in Figure 2 with the second input of Y. If we think of the wires as carrying data for use in an algorithm, then this can be thought of as discarding some of the data.

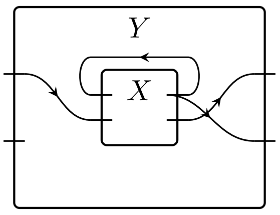

Figure 2: A wiring diagram $\phi : X \to Y$

- We stated earlier that each input and output take values in some given set. To avoid having to convert between different sets, we will assume from now on that wires are only allowed to connect ends which have identical sets.
- While it's not important to know the exact rules that wiring diagrams must follow, for completeness we will note that the following are *not* allowed: wires directly from inputs of Y to outputs of Y, multiple wires going into the same input of X, or multiple wires going into the same output of Y.

We can view these boxes as objects in a category $\mathbf{W}$, with the morphisms in $\mathbf{W}$ being wiring diagrams between boxes. The composition $\psi \circ \phi$ of two wiring diagrams follows naturally: if we have $\phi : X \to Y$ and $\psi : Y \to Z$, then $\psi \circ \phi : X \to Z$ is defined by putting X inside Y inside Z, drawing the wires as specified by ϕ and ψ and then "pretending" Y doesn't exist. Existence of identities and associativity are easy to verify.

Note that up until now, boxes and wiring diagrams have been defined as abstract objects without any particular meaning, although intuitively we are viewing them as some sort of functions that take inputs and produce outputs. We can formalize this notion of filling in the boxes with something as follows:

Definition 1.2. A *filling* of W is a functor $F : \mathbf{W} \to \mathbf{Set}$. This means that for each box X, we associate to it a set of objects $F(X)$ that X can be filled with, and for each wiring diagram $\phi : X \to Y$, we have to define $F(\phi) : F(X) \to F(Y)$, or how ϕ sends fills of X to fills of Y (a *fill* of X is a choice $f \in F(X)$).

We now want to choose a specific F that matches with our idea of the boxes acting as functions. To do this, we will send each box to a set of state machines.

Example 1.3. It might seem more natural to choose F to simply send each box to the set of functions on that box, but in this case, applying a wiring diagram ϕ to a particular function might not result in a pure function, which makes it impossible to define $F(\phi)$. For example, let X be the box with two inputs and one output, all of which are integers, and let Y be the box with one input and one output, also integers. Define ϕ in the diagram below.

In this example, $F(X) = \mathrm{Hom}(\mathbb{Z} \times \mathbb{Z}, \mathbb{Z})$, the set of functions from $\mathbb{Z} \times \mathbb{Z}$ to $\mathbb{Z}$. If we fill X with, say, the plus function $+$, then $F(\phi)(+)$ needs to be an element of $F(Y)$. However, this is impossible because if we input, say, 1 into Y, then the output depends

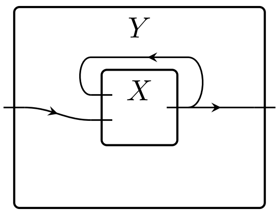

Figure 3: $\phi : X \to Y$

on the value of the internal wire. We introduce state to deal with the problem of these internal wires, because the values of internal wires can then be seen as parts of the overall state.

1.3 State machines

Definition 1.4. (from [1]) Let A and B be sets. An (A, B)-*machine* consists of

1. a set S, called the *state-set*
2. a function $f : S \times A \to S \times B$, called the *state-update function*

An (A, B)-machine is called *initialized* if we have chosen

3. an element $s_0 \in S$, called the *initial state*

We write such a machine as a pair (S, f), or (S, s_0, f) if it is initialized.

Definition 1.5. For a box X, an $\overline{X}$-*machine* is an (A, B)-machine with $A = \overline{\text{inp}(X)}$ and $B = \overline{\text{out}(X)}$, where $\overline{\text{inp}(X)}$ is the set of all possible inputs to X and $\overline{\text{out}(X)}$ is the set of all possible outputs.

For example, in Example 1.3 we have $\overline{\text{inp}(X)} = \mathbb{Z} \times \mathbb{Z}$, $\overline{\text{out}(X)} = \mathbb{Z}$, $\overline{\text{inp}(Y)} = \mathbb{Z}$, and $\overline{\text{out}(Y)} = \mathbb{Z}$.

As stated earlier, we use these state machines to define a filling of $\mathbf{W}$.

Definition 1.6. Let $\mathcal{P} : \mathbf{W} \to \mathbf{Set}$ be the filling of $\mathbf{W}$ that sends a box X to the set of all possible $\overline{X}$-machines.

We've defined how $\mathcal{P}$ acts on the objects of $\mathbf{W}$, but for $\mathcal{P}$ to be a valid filling, we also need to define how it acts on the morphisms (i.e. wiring diagrams). In other words, given an $\overline{X}$-machine $(S, f) \in \mathcal{P}(X)$ and a wiring diagram $\phi : X \to Y$, we want to produce a $\overline{Y}$-machine $(T, g) = \mathcal{P}(\phi)(S, f)$. To do this, we first define the state-set T to be $S \times \overline{\text{int}(\phi)}$, where $\overline{\text{int}(\phi)}$ is the set of all possible values of the internal wires of ϕ. For example, if there were two internal wires which both carried integers, then $\overline{\text{int}(\phi)} = \mathbb{Z} \times \mathbb{Z}$. The definition of g then falls naturally from ϕ and f: to compute $g(S \times \overline{\text{int}(\phi)} \times \overline{\text{inp}(Y)})$, we first use the wiring diagram to get $\overline{\text{inp}(X)}$ from the values of $\overline{\text{int}(\phi)}$ and $\overline{\text{inp}(Y)}$, apply f to $S \times \overline{\text{inp}(X)}$ to get $S \times \overline{\text{out}(X)}$, and finally use the wiring diagram again to determine the new $\overline{\text{int}(\phi)}$ and the output $\overline{\text{out}(Y)}$ from the value of $\overline{\text{out}(X)}$. g takes $S \times \overline{\text{int}(\phi)} \times \overline{\text{inp}(Y)}$ to $S \times \overline{\text{int}(\phi)} \times \overline{\text{out}(Y)}$, as required.

Example 1.7. Once again, consider the wiring diagram in Figure 3. $\mathcal{P}(X)$ is the set of all $(\mathbb{Z}\times\mathbb{Z},\mathbb{Z})$-machines and $\mathrm{int}(\phi)=\mathbb{Z}$. Let (S,f) be an $\overline{X}$-machine with $S=\{s\}$ and $f(s\times a\times b)=(s\times(a+b))$ for all $a,b\in\mathbb{Z}$. Here, a and b are the inputs and $a+b$ is the output, so this is identical to the addition function because we only have one state. We then have $\mathcal{P}(\phi)(S,f)=(T,g)\in\mathcal{P}(Y)$, where $T=S\times\mathbb{Z}\cong\mathbb{Z}$ and $g:T\times\mathbb{Z}\to T\times\mathbb{Z}$ is defined so that $g(t\times a)=((t+a)\times(t+a))$ for all $t,a\in\mathbb{Z}$. Here, t is the state, a is the input, and $t+a$ is both the new state and the output. We can see how each possible $\overline{X}$-machine is taken to a $\overline{Y}$-machine by $\mathcal{P}(\phi)$.

To prove that $\mathcal{P}$ is indeed a functor, we also need to show that it preserves the identity morphism and composition of morphisms. This is easy to check and follows naturally from the definitions, so we won't do it here.

We now consider equivalence of machines.

Definition 1.8. Given a set A, define $\mathrm{List}(A)$ to be the set of all sequences of elements of A. Given $L=(a_1,\ldots,a_n)\in\mathrm{List}(A)$ of length n and an initialized (A,B)-machine (S,s_0,f), we can apply f to each element of L in order, getting a new state and output each time. So, starting at s_0, we have $f(s_0,a_1)=(s_1,b_1)$, $f(s_1,a_2)=(s_2,b_2)$, $f(s_2,a_3)=(s_3,b_3)$, and so on. In the end, we have a final state s_n as well as a list of outputs $(b_1,\ldots,b_n)\in\mathrm{List}(B)$. In this way, each (A,B)-machine can be viewed as a function from $\mathrm{List}(A)$ to $\mathrm{List}(B)$. We call two machines *equivalent* if their corrresponding $\mathrm{List}(A)\to\mathrm{List}(B)$ functions are identical.

Definition 1.9. (from [1]) A *morphism of machines* from some (A,B)-machine (S,f) to another (A,B)-machine (T,g) consists of a function $\rho:S\to T$ such that the following diagram commutes:

$$\begin{tikzcd}
S\times A \arrow[r, "f"] \arrow[d, "\rho\times A"'] & S\times B \arrow[d, "\rho\times B"] \\
T\times A \arrow[r, "g"] & T\times B
\end{tikzcd}$$

In other words, applying f and then ρ should give both the same final state and output as applying ρ first and then g. If the machines are initialized, then we also need $\rho(s_0)=t_0$.

It's easy to show from this diagram that if there exists a morphism of machines $\rho:(S,s_0,f)\to(T,t_0,g)$, then these two machines are equivalent. It's also true (though harder to prove) that if two machines are equivalent, then they must be connected by a sequence of morphisms, although there does not necessarily exist a morphism directly from one to the other. Therefore, two machines are equivalent if and only if they are connected by a sequence of morphisms, so from this point on, we will use "equivalent" to refer to both of these definitions.

Example 1.10. In Example 1.7, we saw the $\overline{Y}$-machine (T,g) with $T=\mathbb{Z}$ and $g(t\times a)=((t+a)\times(t+a))$. If we initialize this machine with $t_0=0$, then it takes $(a_1,\ldots,a_n)\in\mathrm{List}(\mathbb{Z})$ to $(a_1,a_1+a_2,\ldots,a_1+a_2+\cdots+a_n)\in\mathrm{List}(\mathbb{Z})$ with final state $a_1+\ldots+a_n$.

Continuing this example, let's define another machine (T', g') with state set $T' = \mathbb{Z} \cup \{1'\}$, $g'(1 \times 0) = (1' \times 1)$, and $g'(1' \times a) = ((1 + a) \times (1 + a))$ for all $a \in \mathbb{Z}$. This machine is equivalent to (T, g), and the only difference is that it adds a state $1'$ which acts identically to 1, except that applying g' to 1×0 gives $1' \times 1$ instead of 1×1. There exists a morphism ρ from (T', g') to (T, g) defined by $\rho(a) = a$ for $a \in \mathbb{Z}$ and $\rho(1') = 1$. There does not exist a morphism ρ' from (T, g) to (T', g') because we must have $\rho'(b) = b$ for all $b \in \mathbb{Z} \setminus \{1\}$, but then setting $\rho'(1)$ to either 1 or $1'$ fails.

2 Invariants of machines

2.1 The central question

We have a functor $\mathcal{P} : \mathbf{W} \to \mathbf{Set}$ which takes each box X to the set of possible $\overline{X}$-machines. With the addition of morphisms of machines, the set of $\overline{X}$-machines becomes a category, so we can extend this to a functor $\mathcal{P} : \mathbf{W} \to \mathbf{Cat}$.

As previously mentioned, we want to study invariants of these machines, so we need to put them into equivalence classes. Let $\pi_0 : \mathbf{Cat} \to \mathbf{Set}$ be the functor which sends each category to the set of its connected components, where two objects in some category C are part of the same connected component if they are connected by morphisms in C. We see that for each X, the morphisms in the category $\mathcal{P}(X)$ are exactly the equivalences of machines, so the composite functor $\pi_0 \circ \mathcal{P} : \mathbf{W} \to \mathbf{Set}$ takes each box X to the set of equivalence classes of $\overline{X}$-machines, which is what we're interested in.

Definition 2.1. Given an $\overline{X}$-machine $(S, f) \in \mathcal{P}(X)$, let $(\widetilde{S}, \widetilde{f}) \in \pi_0(\mathcal{P}(X))$ be the equivalence class of (S, f).

Definition 2.2. An *invariant* of machines is a functor $I : \mathbf{W} \to \mathbf{Set}$ along with a surjective natural transformation $q : \pi_0\mathcal{P} \to I$. Here, I takes each box X to the set of possible invariants that X can have, and for each box X, there exists a morphism q_X that takes each equivalence class $(\widetilde{S}, \widetilde{f}) \in \pi_0\mathcal{P}(X)$ to some invariant $i \in I(X)$.

Immediately, we can see how to make two trivial invariants: send every machine to 0, or send every machine to its equivalence class. These correspond to the invariants $I(X) = \{0\}$ for all X and $q = 0$, or $I(X) = \pi_0\mathcal{P}(X)$ and q is the identity respectively. This brings us to the central question of this paper: do there exist any nontrivial invariants of these machines?

2.2 Some initial observations

Ideally, what we want to do first is finding a q that works, because once we define q, $I(X)$ is forced to be equal to the image of q_X for each X since q is surjective. However, thinking about entire equivalence classes of machines is much less natural than thinking about individual machines, so we will try assigning an invariant to each individual machine instead. In other words, for each box X, we want to define some invariant $r_X : \mathcal{P}(X) \to I(X)$. What conditions must r_X satisfy?

Firstly, we clearly want $r_X((S,f)) = r_X((S',f'))$ if $(S,f),(S',f') \in \mathcal{P}(X)$ are equivalent $\overline{X}$-machines, if this is to be an invariant in the first place. With this condition, we can define $q_X((\widetilde{S},\widetilde{f})) = r_X((S,f))$ for any representative (S,f) in the equivalence class $(\widetilde{S},\widetilde{f})$ because they must all take on the same value.

Secondly, given a wiring diagram $\phi : X \to Y$, we want the following diagram to commute (this is from the definition of a natural transformation):

$$\begin{tikzcd}
\pi_0\mathcal{P}(X) \arrow[r, "\pi_0\mathcal{P}(\phi)"] \arrow[d, "q_X"'] & \pi_0\mathcal{P}(Y) \arrow[d, "q_Y"] \\
I(X) \arrow[r, "I(\phi)"] & I(Y)
\end{tikzcd}$$

Note that we are defining both $I(X)$ and $I(Y)$ from q as stated above, and the definition of $I(\phi)$ also follows from q because given any $i \in I(X)$, we can find some $(\widetilde{S},\widetilde{f}) \in \pi_0\mathcal{P}(X)$ that q_X takes to i. We can then apply $\pi_0\mathcal{P}(\phi)$ and q_Y to $(\widetilde{S},\widetilde{f})$ to get some $i' \in I(Y)$, so we must have $I(\phi)(i) = i'$. Therefore, the only way that this diagram can fail to exist is if some $i \in I(X)$ is sent to two or more different elements of $I(Y)$, so this imposes another condition on r: Given two $\overline{X}$-machines (S,f) and (S',f') with $r_X((S,f)) = r_X((S',f'))$ and a wiring diagram $\phi : X \to Y$, the two induced $\overline{Y}$-machines $\mathcal{P}(\phi)(S,f)$ and $\mathcal{P}(\phi)(S',f')$ must also have the same invariant when r_Y is applied to both of them. This is a necessary and sufficient condition for $I(\phi)$ being uniquely defined.

Finding an r that satisfies both conditions above, which we will call *condition 1* and *condition 2* respectively, is equivalent to finding an invariant as defined in Definition 2.2, so the rest of the paper will be devoted to finding such a nontrivial r.

2.3 A quick summary

Before moving on, we note that the actual problem is not particularly complicated and can indeed be stated without invoking any sort of higher-level math. However, we hope that by presenting the basic category theory formalism, we can better show the motivation and structure behind this problem. That said, we will now briefly summarize what we have so far without referring to category theory: A wiring diagram $\phi : X \to Y$ is made by placing a box X inside a box Y and connecting the various inputs and outputs with wires. If we fill X with a state machine (S,f), then this combined with ϕ induces a state machine on Y. We wish to assign an invariant to each state machine such that the following two conditions hold:

1. Equivalent state machines are assigned the same invariant, where we define two machines to be equivalent if they act the same way on all lists of inputs.

2. Given two machines on X that have the same invariant and a wiring diagram $\phi : X \to Y$, the two induced machines on Y also need to have the same invariant.

We also want the invariant to be non-trivial, so it cannot take all machines to the same value, and it also cannot simply take each machine to its equivalence class.

3 Potential methods for finding invariants

3.1 Satisfying condition 1

We first note that if we only want to satisfy condition 1, then we can easily come up with plenty of nontrivial invariants. For example, given an $\overline{X}$-machine (S, f), we can let $r_X((S, f))$ be the total number of distinct outputs $b \in \overline{\text{out}(X)}$ when f is run over all pairs $s \times a \in S \times \overline{\text{inp}(X)}$. Indeed, any property that only depends on the inputs and outputs of f will work, but it's not clear if any of these will also satisfy condition 2 because two $\overline{X}$-machines that have the same condition 1-invariant might not be equivalent, so plugging them both into some wiring diagram $\phi : X \to Y$ doesn't guarantee anything about the resulting $\overline{Y}$-machines and certainly doesn't imply that they have the same invariant. Because of this, it seems like invariants that satisfy condition 2 are a little more interesting and not as easy to find.

3.2 Satisfying condition 2

In order to find a function r that satisfies condition 2, we want to choose r so that given an $\overline{X}$-machine (S, f) and a wiring diagram $\phi : X \to Y$, the invariant of the induced $\overline{Y}$-machine $\mathcal{P}(\phi)(S, f)$ should only depend on the invariant of the original machine (S, f). We can do this by including a lot of information in the invariant that carries over to wiring diagrams:

Theorem 3.1. *For an $\overline{X}$-machine (S, f), $a \in \overline{inp(X)}$, and $b \in \overline{out(X)}$, let $z_{a,b}$ be the number of states $s \in S$ such that $f(s \times a) = s \times b$. Define $r_X((S, f))$ to be the (possibly infinite) $|\overline{inp(X)}| \times |\overline{out(X)}|$ matrix whose (i, j)-entry is $z_{a,b}$. Then r satisfies condition 2.*

Proof. Consider a wiring diagram $\phi : X \to Y$, and let $\mathcal{P}(\phi)(S, f) = (T, g)$. It suffices to show that the entries of the matrix $r_Y((T, g))$ only depend on the entries of the matrix $r_X((S, f))$. Without loss of generality, choose $a' \in \overline{\text{inp}(Y)}$ and $b' \in \overline{\text{out}(Y)}$. We wish to show that the (a', b')-entry of $r_Y((T, g))$ can be written in terms of entries of $r_X((S, f))$. The value of this (a', b')-entry is equal to the number of states $s \times c \in S \times \overline{\text{int}(\phi)} = T$ such that $g((s \times c) \times a') = (s \times c) \times b'$ by definition. To compute this, we note the following facts:

1. Given a' (the input of Y) and the internal wires c, we can apply the wiring diagram to generate the exact input of X, say a, because each input of X comes from either an input of Y or an internal wire.
2. Given b (the output of X), we can apply the wiring diagram to generate b' (the output of Y), as well as the internal wires c, because all of these must come from the output of X.

Therefore, if we consider a single output $b \in \overline{\text{out}(X)}$ such that the wiring diagram generates b' as an output of Y, then the total number of states $s \times c$ such that $g((s \times$

$c) \times a') = (s \times c) \times b'$ *and* the output of X is b is equal to the number of states s such that $f(s \times a) = s \times b$, where a is determined from a' and c, and c is determined from b as mentioned above. However, this is an element of $r_X((S, f))$, so if we sum this over all valid b's, we see that the (a', b')-entry of $r_Y((T, g))$ can indeed be written as the sum of entries in the matrix $r_X((S, f))$. Therefore, r satisfies condition 2. □

In addition to the one specified in Theorem 3.1, we have found a few similar functions that also satisfy condition 2. Unfortunately, none of these also satisfy condition 1 because by including so much information about the machine, in particular information about the state-set, they end up assigning different invariants to many equivalent machines.

3.3 Wiring diagrams acting as constraints on invariants

So far, we have had little success in directly using properties of the machines to construct invariants. This brings us to the major idea in this section: what if we *indirectly* attempted to construct invariants by looking at the action of wiring diagrams on invariants? In other words, instead of focusing on defining q, we focus on possible values of $I(\phi)$ and define q based on that. By condition 2, given a wiring diagram $\phi : X \to Y$, we must have a function $I(\phi) : I(X) \to I(Y)$. Because I commutes with composition (i.e. $I(\psi) \circ I(\phi) = I(\psi \circ \phi)$), this has the potential to heavily constrain what invariants we can assign to particular machines. For example, if $\psi \circ \phi$ is equivalent to the identity wiring diagram and applying $I(\phi)$ to some machine with invariant i_1 gives a machine with invariant i_2, then $I(\psi)$ must take all machines with invariant i_2 back to machines with invariant i_1.

Example 3.2. Let's work in the subcategory $\mathbf{W}^*$ of $\mathbf{W}$ that only consists of boxes with two inputs and one output, all of which are boolean. Also, for simplicity, assume that we are only looking for boolean invariants, so $I(X) = \{0, 1\}$ for all X. In this case, given any two boxes X and Y, there are exactly 9 wiring diagrams between them because each of the two inputs of X can independently come from 3 places: the first input of Y, the second input of Y, or the output of X. Figures 4, 5, and 6 are three examples of these wiring diagrams, which we label a, b, and c.

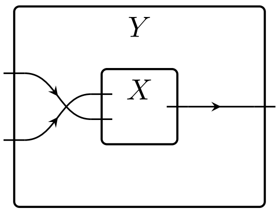

Figure 4: a

If we look at all possible compositions of these 9 diagrams, we find that c composed with anything is still c, because c ignores both inputs of Y. Without loss of generality, let $I(c)(0) = 0$. We have $\phi c = c$ for all morphisms ϕ in $\mathbf{W}^*$, so $I(\phi c)(0) = I(c)(0) \implies$

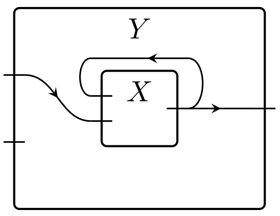

Figure 5: b

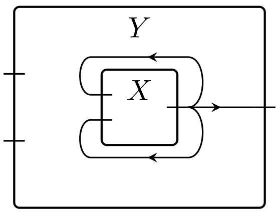

Figure 6: c

$I(\phi)(I(c)(0)) = 0 \implies I(\phi)(0) = 0$. Therefore, all possible wiring diagrams must take invariant 0 machines to invariant 0 machines.

We also have $a^2 = i$, where i is the identity wiring diagram, so $I(a^2)(1) = I(i)(1) = 1 \implies I(a)(1) = 1$. Using similar logic on the other compositions of wiring diagrams, we find that in this example, there are two possibilities for the $I(\phi)$'s:

1. $I(\phi)(0) = 0$ and $I(\phi)(1) = 1$ for all ϕ. As an invariant, this corresponds to assigning each connected component of $\mathcal{P}(X)$ either 0 or 1, because if two machines are connected, then they must have the same invariant.

2. $I(\phi)(0) = 0$ and $I(\phi)(1) = 0$ for all ϕ except a and i; $I(a)(0) = I(i)(0) = 0$ and $I(a)(1) = I(i)(1) = 1$. This corresponds to assigning 0 to all machines that "throw away" at least one of the two inputs, because the other 7 wiring diagrams besides a and i "throw away" at least one of the inputs of Y, either by using an internal wire to supply an input to X, or by using a single input of Y to supply both inputs of X. The remaining machines come in pairs, and each pair can be assigned a 0 or 1.

The second invariant possibility in the above example is particularly interesting and can be generalized to a broader statement, which is discussed in the next section.

4 Results

4.1 Our main result

Theorem 4.1. *Let A be a set of morphisms in* **W** *which satisfies the following properties:*

1. *For each $\phi \in A$ with $\phi : X \to Y$, there exists a left-inverse $\phi^{-1} : Y \to X$ (not necessarily in A) such that $\phi^{-1} \circ \phi$ is the identity on X.*

2. *If $\psi \notin A$, then $\phi\psi \notin A$ for all morphisms ϕ.*

Let $I(\phi)$ be defined as follows:

1. *If $\phi \in A$, then $I(\phi)(0) = 0$ and $I(\phi)(1) = 1$.*

2. *If $\phi \notin A$, then $I(\phi)(0) = 0$ and $I(\phi)(1) = 0$.*

Then the following invariant is consistent with $I(\phi)$ and satisfies both conditions 1 and 2:

1. *For each equivalence class of machines, assign all of them invariant 0 if there exists some representative machine (T, g) in the class as well as some $\phi \notin A$ and any machine (S, f) such that $\mathcal{P}(\phi)((S, f)) = (T, g)$.*

2. *Otherwise, assign all of them invariant 1.*

Proof. We will first show that the definition of our invariant is consistent with our definition of $I(\phi)$. This is split into four cases, depending on whether ϕ is in or not in A and whether (S, f) is sent to 0 or 1 by our invariant. As usual, we let ϕ be a wiring diagram from X to Y, so (S, f) is an $\overline{X}$-machine.

1. $\phi \notin A$ and (S, f) has invariant 0. In this case, we want to show that the induced machine $(T, g) = \mathcal{P}(\phi)((S, f))$ has invariant 0 so that it's consistent with $I(\phi)(0) = 0$. However, this is obvious by definition because (T, g) is itself a representative machine in its equivalence class that is the image of a wiring diagram not in A, namely ϕ, so this case is fine.

2. $\phi \notin A$ and (S, f) has invariant 1. In this case, we want to show that our invariant is consistent with $I(\phi)(1) = 0$ for $\phi \notin A$. We can use the exact same argument as in case 1 to show that the induced machine (T, g) has invariant 0, so we're good.

3. $\phi \in A$ and (S, f) has invariant 0. Here, we want to show that the induced machine (T, g) has invariant 0. To prove this, we note that because (S, f) has invariant 0, by definition there exists some equivalent machine (S', f'), some $\psi \notin A$, and another machine (R, h) such that $\mathcal{P}(\psi)((R, h)) = (S', f')$. Applying $\mathcal{P}(\phi)$ to both sides gives $\mathcal{P}(\phi\psi)((R, h)) = \mathcal{P}(\phi)((S', f'))$. The machine on the right-hand side of this is equivalent to (T, g) because $\mathcal{P}(\phi)((S, f)) = (T, g)$ and (S, f) is equivalent to (S', f'). However, from our definition of A, we know that $\phi\psi \notin A$ because $\psi \notin A$, so the equivalence class of $\mathcal{P}(\phi)((S', f'))$ must be sent to 0 by our invariant. As just stated, (T, g) is in this equivalence class, so it must also have invariant 0 as desired.

4. $\phi \in A$ and (S, f) has invariant 1. Here, we want to show that the induced machine (T, g) has invariant 1 so that it's consistent with $I(\phi)(1) = 1$. To do so, we will prove the contrapositive: that is, if $\phi \in A$, $\mathcal{P}(\phi)((S, f)) = (T, g)$, and (T, g) has invariant 0, then (S, f) also has invariant 0. Using the existence of an inverse $\phi^{-1} : Y \to X$, we have that $\mathcal{P}(\phi^{-1})((T, g)) = (S, f)$. We can then apply the same argument as in case 3 with (S, f) and (T, g) switched and ϕ replaced by ϕ^{-1}, because it did not depend on ϕ being a member of A.

This shows that our invariant is indeed consistent. It satisfies condition 1 by definition, because we are directly assigning invariants to equivalence classes. It also satisfies condition 2 because it's consistent with our definition of $I(\phi)$, so this completes the proof. $\square$

One example of such a set A is simply the set of all morphisms which have left inverses. Unfortunately, it turns out that the invariant associated with this A is trivial, sending everything to 0, because for any $\overline{Y}$-machine (T, g), if we let X be a box with more inputs than Y and (S, f) be any $\overline{X}$-machine, then it's easy to verify that any wiring diagram from (S, f) to (T, g) doesn't have a left inverse. However, if we limit ourselves to any subcategory of finitely many boxes, then such an X will no longer always exist, and indeed we can prove that there exist machines with invariant 1. So, although this idea doesn't necessarily work on the entire category $\mathbf{W}$, it does produce a nontrivial invariant for any finite subset of boxes and the wiring diagrams between them.

4.2 Further work

We certainly have not completely explored this idea of seeing how wiring diagrams act on invariants. Things to try out in the future to either search for invariants or to prove that no such invariant exists include

1. Redefining the set A in different ways.
2. Expanding the invariant set (for most of the time, we stuck with boolean invariants for simplicity).
3. Investigating the structure of the connected components in greater detail, starting with smaller examples.
4. Generally investigating the action of wiring diagrams on invariants more (we thoroughly looked at the example of a box with two inputs and one output, but perhaps going bigger will reveal new things).

5 Acknowledgements

The author of this paper would like to thank his mentor Lyuboslav for continually pushing and inspiring him to work despite his laziness and for being very helpful and accessible during the program. David Spivak suggested the project and also gave many insightful ideas in the meetings we had together. The author would also like to thank Professors David Jerison and Ankur Moitra for all the help and all the hard work they put into SPUR, and of course the program director Slava Gerovitch.

References

[1] David Spivak, "Wiring diagrams and state machines" talk, February 2014, http://math.mit.edu/~dspivak/informatics/talks/WD-IntroductoryTalk.pdf