Circuits via topoi

Arnaud Spiwack

Circuits via topoi

Arnaud Spiwack

September 30, 2015

Abstract

Leveraging topos theory a semantics can be given to sequential circuits where time-sensitive gates, such as unit delay, are treated uniformly with combinational gates. Both kinds of gates are functions in a particular topos: the topos of presheaves over the natural ordering of $\mathbb{N}$. This is used to show that sequential circuits validate the equational theory of traced categories.

When giving semantics to circuits (typically boolean circuits), it is customary to treat the combinational – *i.e.* time-independent – parts of the circuits differently from time sensitive ones. Since it is usually assumed that the only time-sensitive gate is the unit delay, each outgoing wire from a delay is considered an additional input, and each incoming wire an additional output. Some care is taken to feed the right output into the right input at next iteration, and so time-sensitivity is eliminated and one can reason on a purely combinational circuit.

This is not very convenient to reason equationally about moving unit delays for better placement. But this approach really breaks down when considering time-sensitive gates which are not simple unit delay. This article takes its root in the study of compilation of the circuit programming language Faust [13]. Faust features a somewhat unusual kind of delay gate, written `s@d`, where `s` is an arbitrary signal, and `d` is a *time-varying* bounded natural number signal whose value at time t determines how far in the past of `s` to fetch the value of the delayed signal.

With such a construct, it becomes impossible to "cut" a circuit into a combinational circuit. At least not without heavy modifications (the constraint that `d` is bounded is imposed in order to be able to compile the program in constant memory, so such a circuit can be reduced to use only unit delays).

To address this issue, let us turn to presheaves and topos theory. The critical property which we shall use is that topoi are models of constructive mathematics. Therefore we shall first develop a theory of combinational circuits in ordinary constructive mathematics, then lift it to sequential circuits *via* a presheaf construction.

Much work has been put [3, 1], recently, in using category theory to explain and exploit the linear algebraic aspects of circuits from control-theory. This article explores an orthogonal axis of the design space. Both can, and should, in principle be combined to obtain linear algebra with time. It is what signal processing is made of.

Before we move on, I have to start with an apology: despite the subject of topoi and presheaves being rather technical, I will be assuming quite a bit of familiarity with them in this article. I realise that this will make this article unnecessarily arduous for many. But in order for this article to be written at all, I felt I had to limit its scope so. A good, exhaustive, introduction to topos theory can be found in Mac Lane & Moerdijk's *Sheaves in geometry and logic* [10].

Acknowledgement I was once sitting with Noam Zeilberger listening to a seminar by Gérard Berry. About circuits. Berry's presentation was obviously of great interest to the both of us as we went on discussing its content for quite a while after that. At some point Zeilberger remarked: "I don't really know what a circuit *is*", and I suddenly realised that I didn't quite either; despite the intuitive, and occasionally concrete, nature of circuits. If Zeilberger's remark gave you pause, as it did to me, then read on for my attempt at a definition.

1 Combinational circuits

It is direct to give a definition of combinational circuits if they are not allowed to have loops: just interpret each gate as a function a compose things appropriately. Or, more generally, if you are so disposed, interpret each gate as an arrow in some cartesian category, and interpret appropriately.

The case that drove my interest, however, requires loops – aka feedback. I am not particularly in need of delay-free loops, although this is of legitimate interest (see for instance [12]), but the semantics which is developed in this section will be lifted in Section 2 to sequential circuits which are useless without some form of feedback.

1.1 Constructive domain semantics

As good computer scientist ought to when faced with tricky fixed point (even in circuits, loops are, after all, fixed points), let us turn to domains. Before we give a formal description of our semantics, let me note that it is a straightforward variant of the rather venerable *three-valued semantics* of combinational boolean circuits [11], which, by the way, has been shown, with *caveats*, to be a good semantics for electronic circuits [12].

Boolean circuits have, of course, a special relevance in computing science due to their being the basic building block of computers. But we will not restrict ourselves so. Wires will be allowed to carry values of any type we wish. The Faust programming language, for instance, has wires of type $\mathbb{N}$ and $\mathbb{R}$ (floating point numbers, in practice). The types, which, for the purpose of this article, are simply sets of permitted values, allowed for the wires by a circuit language will be called *base types*.

Definition 1 (Bounded height domain)**.** *A bounded height domain is a partially ordered set D equipped with a number b such that every increasing chain $x_1 \leqslant \ldots \leqslant x_n$ in D with $n > b$ has a pair $x_i \geqslant x_j$ with $i < j$ (equivalently, for every $i \leqslant k \leqslant j$, $x_i = x_k$).*

Circuits will be given a semantics as increasing functions between such domains. To the extent that the material present in this section is different from the usual treatment it is to render this section constructive to be compatible with the topos of Section 2. This is the reason why we focus on bounded height domains rather than the more usual ω-CPOs. Note also that b, in Definition 1 is *not* the height of the domain but rather an upper bound on this height. The reason is that, constructively, there may not be an exact height (see also [4] for more thoughts on finiteness in constructive mathematics). Every proof, in this section, is constructive.

Bounded height domain have the fixed-point property, just like other kinds of domains. Note that the fixed-point property of CPOs or ω-CPOs are also constructive. The added value of bounded height domains is that there are really few constructive CPOs or ω-CPOs (see Remark 1 below). Another practical advantage of bounded height domain is that the fixed-point property applies to all increasing functions, which will free us from proving continuity.

Theorem 1 (Fixed-point property)**.** *Every increasing function $f : D \to D$ for a bounded height domain D with a smallest element $\bot$ has a smallest fixed-point.*

Proof. Let b be a bound on the height of D. The sequence $\bot \leqslant f(\bot) \leqslant f^2(\bot) \leqslant \ldots \leqslant f^b(\bot)$ has length $b+1$. By definition, there is an $i < b$ such that $f^i(\bot) = f^{i+1}(\bot)$ hence $f^b(\bot) = f^i(\bot)$ is a fixed point.

It is the smallest since, by induction, for any fixed point x_0 of f and every k, $f^k(\bot) \leqslant x_0$. □

Remark that we can refine the proof to show that the least fixed point of f is also its least pre-fixed point (*i.e.* such that $f(x) \leqslant x$).

Base types, which are sets, can be naturally identified to bounded height domains (without a smallest element).

Definition 2 (Flat domains)**.** *Given a set A, the partially ordered set (also noted A) where $x \leqslant y \iff x = y$ is a domain of height bounded by 1, which we call a flat domain.*

Continuing on the subject of constructiveness, notice that flat domains are an example of domain which can't be assigned a height. Indeed, if A is inhabited then A has height 1, whereas when A is empty then A has height 0 but it is not possible, in constructive mathematics to decide whether A is empty or not, the height flat domains is, therefore, not well defined (it is, in fact, impossible to define a non-constant integer-valued function on sets [5]).

In order to use base types in conjunction with the fixed point property they need a smallest element which we add freely.

Definition 3 (Lifted domains)**.** *Given a domain A with a bound b on its height, we construct a domain $A_\bot$, called* lifted*, by adding a distinguished element $\bot$ to A and considering it smaller than every element of A: $\forall x^{\in A}.\ \bot < x$. The height of $A_\bot$ is bounded by $b+1$*

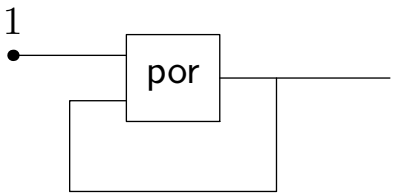
Wires in circuits will be interpreted as taking value in the lifted flat domains corresponding to base types. Increasing functions between lifted flat domain are such that if $f(\bot) \neq \bot$, then for any a, $f(a) = f(\bot)$. In particular, if f is such a function with several fixed points, then $f(\bot) = \bot$ and the smallest

fixed point is $\bot$. So $\bot$ represents both the absence of a well-defined fixed point and the presence of several fixed points.

Obviously, in order for the smallest fixed point of f to be non-$\bot$, f needs to ignore some of its input wires, for instance f could be the well-know parallel or:

por	$\bot$	0	1
$\bot$	$\bot$	$\bot$	1
0	$\bot$	0	1
1	1	1	1

When one of the input of the parallel or is 1, then the output is 1, whatever the behaviour of the other input. In particular the following circuit is well defined (it outputs 1):



More useful examples can be found in [11, 12].

Remark 1. *Lifted flat domains are an example of bounded-height domains which are not necessarily ω-CPOs, constructively. Indeed consider $\mathbf{1} = \{0\}$, the singleton set, then $\mathbf{1}_\bot$ is not constructively an ω-CPO. An ω-chain in $\mathbf{1}_\bot$ is an infinite sequence of $\bot$ and 0 (such that after a 0, every element is 0). If an ω-chain has $\bot$ as an upper bound, then all of its elements are $\bot$, if an ω-chain has 0 as an upper bound, at least one of its elements must be 0[1]. If every ω-chain had an upper bound, it would give a way to decide whether they contain a 0 or not, which is equivalent to the limited principle of omniscience: a known-to-be-non-constructive principle.*

To formalise circuits with multiple wires we remark that bounded height domains are closed by cartesian products.

Lemma 1 (Cartesian product of domains)**.** *The product $A \times B$ of two bounded height domains of respective bound b_A and b_B, with order $(x_1, y_1) \leqslant (x_2, y_2) \iff x_1 \leqslant x_2 \wedge y_1 \leqslant y_2$, is a domain whose height is bounded by $b_A \times b_B$.*

Proof. A chain in $A \times B$ is simply a list of pairs (x, y), with an constraint on consecutive pairs. However, chains can be represented differently as a list of pairs (x, l) with l a chain in B, with the intent that the pair $(x, [y_1, \ldots, y_n])$ represents the chain $(x, y_1) \leqslant \ldots \leqslant (x, y_n)$. So that chains now have multiple representations depending on how successive pairs with the same first component are "contracted".

We begin with the simplest representation where every pair (x, l) is an $(x, [y])$. Now, as long as our list is of size longer than b_A (without loss of generality we can suppose both b_A and b_B to be non-zero) we can use the

[1] Accomplished constructive mathematicians may noticed that I have made use of Markov's principle in this statement: it's a valid thing to do, though since if something is unprovable from Markov's principle, it is certainly unprovable without. Alternatively, "must" in that sentence can be interpreted as the double-negation modality, in which case the statement is constructively true, and leads to a weaker, still non-constructive, version of the limited principle of omniscience.

fact that the first components describe a chain to find consecutive positions with the same x which we can contract. Hence strictly reducing the size of our list. This process gives us a contracted representation of the long chain of length b_A or less. But, since the total length of the chain is larger than $b_A \times b_B$, there must be at least one second-component list with length larger than b_B. Applying the definition of the bound b_B to this list concludes the proof. □

Combinational gates are, therefore, interpreted as increasing functions of type $A_{1\perp} \times \ldots \times A_{n\perp} \to A_{n+1\perp} \times \ldots \times A_{p\perp}$ (with each A_i being a base type).

1.2 Traced category

What is left is to use the fixed-point property to make precise the definition of feedback wires, the solution is given by Hasegawa [6, Theorem 3.1] who gives a method to transform fixed-point operators into traces.

Traced categories [8] provide a graphical language which is essentially the same as circuits with feedback. It is reassuring that circuits can be interpreted as arrows in such a traced category. It provides a natural equational theory on circuits which can be, among other things, leveraged to produce optimisation schemes [9].

Lemma 2 (Local fixed-point property)**.** *Theorem 1 can be extended to produce a local fixed point function: let $f : A \times X \to X$ an increasing function (A and X bounded height domain with a smallest element), there is an increasing function $\mu(f) : A \to X$, such that for any $a : A$, $\mu(f)(a)$ is the least fixed point of the increasing function $\lambda x.\, f(a, x)$.*

Proof. The proof bulk of the proof is the same as Theorem 1, taking into account that, by definition of cartesian product $\lambda x.\, f(a, x)$ is, indeed, increasing: let b a bound on the height of X, $\mu(f)(a) = (\lambda x.\, f(a, x))^b(\perp)$.

We need to check the $\mu(f)$ is indeed increasing. But since f is increasing, for any $a \leqslant a'$ and any $x \leqslant x'$ $f(a, x) \leqslant f(a', x')$; by induction, we conclude that $\mu(f)(a) \leqslant \mu(f)(a')$. □

Hasegawa tells us that there are three properties to verify for a fixed point operator to yield a trace (note that, reciprocally, all traces in a cartesian category yield such a fixed point operator). We shall write $\mu_a x.\, f(a, x)$ instead of $\mu(\lambda\,(a, x)\,.\, f(a, x))$. In addition and by definition, $(\mu_a x.\, f(a, x))\,(a_0) = \mu x.\, f(a_0, x)$; since the former is cumbersome, we will use the latter as a shorthand.

Lemma 3 (Naturality in A)**.** *For any $f : A \times X \to X$ and $g : B \to A$, the following holds: $\mu_b x.\, f(g(b), x) = \mu(f) \circ g$.*

Proof. Let $b : B$, $(\mu_b x.\, f(g(b), x))\,(b) = \mu x.\, f(g(b), x)$ is, by definition, the least fixed point of $\lambda x.\, f(g(b), x)$. And, also by definition, so is $\mu(f)(g(b))$. □

Lemma 4 (Naturality in X)**.** *Let f be an increasing function in $A \times X \to Y$. For any $g : Y \to X$, $\mu_a x.\, g(f(a, x)) = g \circ (\mu_a y.\, f(a, g(y)))$.*

Proof. Let us fix an $a : A$.

- Let us prove that $g(\mu y.\ f(a, g(y)))$ is a fixed point of $\lambda x.\, g(f(a, x))$, and therefore $\mu x.\ g(f(a, x)) \leqslant g(\mu y.\ f(a, g(y)))$. This follows immediately from the fact that $f(a, g(\mu y.\ f(a, g(y)))) = \mu y.\ f(a, g(y), a)$ and the fact that g is increasing.

- Conversely, we prove similarly that $f(a, \mu x.\ g(f(a, x)))$ is a fixed point of $\lambda y.\, f(a, g(y))$. This yields $f(a, \mu x.\ g(f(a, x))) \geqslant \mu y.\ f(a, g(y)) = f(a, g(\mu y.\ f(a, g(y))))$, and then, $\mu x.\ g(f(a, x)) \geqslant g(\mu y.\ f(a, g(y)))$ by monotonicity of $\lambda x.\, f(a, x)$.

The two inequalities prove the lemma. □

Lemma 5 (Bekič)**.** *Let $f : A \times X \times Y \to X$ and $g : A \times X \times Y \to Y$. Taking $h : A \to X$ to be $h(a) = \mu x.\ f(a, x, \mu(g)(a, x))$, the following holds* $\mu_a\, (x, y)\,.\ (f(a, x, y), g(a, x, y)) = \lambda a.\ (h(a), \mu(g)(a, h(a)))$

Proof. For $a : A$, let us prove that $(h(a), \mu(g)(a, h(a)))$ is a fixed point of $\lambda\, (x, y)\,.\ (f(a, x, y), g(a, x, y))$.

$$
\begin{array}{lll}
 & (f(a, h(a), \mu(g)(a, h(a))), g(a, h(a), \mu(g)(a, h(a)))) & \\
= & (f(a, h(a), \mu(g)(a, h(a))), \mu(g)(a, h(a))) & \text{(definition of } \mu(g)) \\
= & (h(a), \mu(g)(a, h(a))) & \text{(definition of } h)
\end{array}
$$

We also have that

$$
\begin{array}{ll}
 & \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y)) \\
= & (f(a, \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y))), g(a, \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y))))
\end{array}
$$

This allows us to test both components for being fixed points or the corresponding function, which will suffice to conclude.

- $f(a, \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y))) \geqslant h(a)$: by definition of h it suffices to show that $f(a, \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y)))$ is a pre-fixed point of $\lambda x.\, f(a, x, \mu(g)((a, x)))$. After tedious calculations[2], it amounts to proving, calling $(x_0, y_0) = \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y))$, that $\mu y.\ g(a, x_0, y) \leqslant y_0$. It is easily checked that y_0 is a fixed point of $\lambda y.\, g(a, x_0, y)$, which concludes this sub-proof.

- $g(a, \mu\, (x, y)\,.\ (f(a, x, y), g(a, x, y))) \geqslant \mu(g)(a, h(a))$. The argument is similar to above.

□

2 Sequential circuits

Adding time-sensitive gates forces to change the semantics. Sequential circuits are not to be seen as functions from (product of) base types to base types, but rather as functions from streams of base types to stream of base types.

[2]So tedious, in fact, that I ended up formalising most of this section in the Coq proof assistant which, contrary to me, is not susceptible to calculation mistakes. Plus, I was getting lost and could use the help. This goes to prove that for certain mathematical proofs, proof assistant can be a productive way to develop proofs.

Unfortunately, the type $A^{\mathbb{N}}$ of streams of a finite height domain A is not a finite height domain in any useful way.

To be able to model feedback, a change a perspective will be needed. The typical approach to analysis of sequential circuits with feedback is to "cut" unit delays making their incoming wire into a special new output and their outgoing wire into a special new input. What makes this transformation even meaningful is the requirement that to compute a finite prefix of length n of a circuit's output, only a finite prefix of length n of the input is necessary. This requirement is called *causality*.

2.1 Causal sets

With that in mind, it makes sense to see streams not as a whole, but as a progression of prefixes $(A^n)_{n\in\mathbb{N}}$. All of the A^n, by virtue of being finite products of finite height domains, are finite height domains. A causal function can, then, be defined as a collection $(f_n)_{n\in\mathbb{N}}$ of functions $A^n \to B^n$ such that $f_{n+1}(w \cdot a)$ is of the form $f_n(w) \cdot b$.

To abstract over these notions, let us introduce a topos – *i.e.* a model of constructive mathematics – where such a presentation of streams and causal functions is natural.

Definition 4 (Causal sets)**.** *The topos of* causal sets *is the topos of presheaves over the set of natural number with its standard ordering.*

This topos has been extensively studied by Birkedal, Møgelberg, Schwinghammer & Støvring [2] under the name *topos of trees* to contribute to the related problem of step-indexing. Their article can serve as a reference.

A causal set is, therefore, given by a family $(A_n)_{n\in\mathbb{N}}$ of sets together with *restriction* functions $r_n : A_{n+1} \to A_n$. Causal functions are families of functions $(f_n)_{n\in\mathbb{N}}$ such that $r_n(f_{n+1}(a)) = f_n(r_n(a))$. Streams, seen, as above, as a progression of prefixes, form a causal set $\mathbb{S}_A$ with $(\mathbb{S}_A)_n = A^n$ and $r_n(w \cdot a) = w$. Causal functions, in the sense of the topos of causal set, on $\mathbb{S}_A$ are the same as causal arrows of streams; so that arrows in the topos of causal sets are, indeed, a generalisation of causal functions of streams.

By analogy with streams, the sets A_n, constituting the causal set A, are called the sets of prefixes of A or just prefixes of A. Since the indices of the functions can often be inferred from the context, they will often be omitted; for example: the compatibility of f with restrictions may be written $r(f(a)) = f(r(a))$.

Topos are models of constructive mathematics, hence there is an interplay between *internal* statements of the topos of causal sets which are derived using the rules of constructive mathematics and *external* statements of ordinary mathematics. Internal statements are related to external statements via the Kripke-Joyal semantics [10, Section VI.6]: when φ is an internal proposition in context Γ (Γ is a (conjunction of) causal set giving sense to the free variables of φ), then for $n \in \mathbb{N}$ and $\alpha_n \in \Gamma_n$ an external proposition $n \models \varphi(\alpha_n)$ is defined. The proposition $n \models \varphi(\alpha_n)$ means that φ holds at least until and including time n on α_n. The main property being that if φ is provable in constructive mathematics (usually written $\vdash \varphi$), then for all n and α_n, $n \models \varphi(\alpha_n)$. And conversely, if φ is such that $n \models \varphi(\alpha_n)$ then φ is internally valid.

2.2 Causal domains

Let us now endeavour to give an external description of internal bounded height domain, so as to show that $\mathbb{S}_{A_\perp}$ is an internal bounded height domain for some base type A.

The ordering relation is reflexive: $\vdash x \leqslant x$. That is, $n \models \alpha_n \leqslant \alpha_n$ for any n and α_n. In other words, a reflexive causal relation, is a family of reflexive relations on each set of prefixes (compatible with restrictions). The same holds for symmetry and transitivity, such that an internal ordering relation is an ordering relation on each prefix[3].

The translation of ordering relations illustrate the purpose of causal sets: to make it possible to reason on finite prefixes of infinite data. This is, indeed, what we were looking for, to be able to use the fact that prefixes of $\mathbb{S}_{A_\perp}$ are bounded height domains. We should expect, at this point, that internal bounded height domains are exactly those causal sets where A_n is a bounded height domain for each n, which is indeed the case.

The key observation is that the casual $\mathsf{List}(A)$ which is the initial algebra of the functor $A \times X + 1$ can be defined as $(\mathsf{List}(A))_n = \mathsf{List}(A_n)$. The restriction functions act pointwise on the elements of each list. Therefore, since subsets are taken pointwise *i.e.* $\{x \in A \mid \varphi(x)\}_n = \{x \in A_n \mid \varphi_n(x)\}$, chains internal to the topos of causal sets are chains on prefixes (compatible with restrictions).

Thanks to this observation, the internal definition of bounded height domain can be interpreted: $b_n \in \mathbb{N}$ is a bound on the height of A at time n if for all $k \leqslant n$, b_n is a bound on the height of A_k in the ordinary sense. Since being a bound is a monotonous property on b, an internal bounded-height domain is a causal set with all prefixes being externally bounded-height domains. An internal domain A has a smallest element if each of the A_n has and restrictions map smallest elements to smallest elements.

Lemma 6. *The causal set $\mathbb{S}_{A_\perp}$, for some ordinary set A, is a finite height domain with a smallest element internal to the topos of causal sets.*

Proof. Since the height $(\mathbb{S}_{A_\perp})_n = A_\perp^n$ is bounded by 2^n, which also bounds all the $A_\perp^k$ for $k \leqslant n$. The smallest element of A_n is $(\perp, \ldots, \perp)$. □

As a consequence, we can build circuits as causal increasing functions $\mathbb{S}_{A_1\perp} \times \ldots \times \mathbb{S}_{A_n\perp} \to \mathbb{S}_{A_{n+1}\perp} \times \ldots \times \mathbb{S}_{A_p\perp}$ and feedback wires can be interpreted as internal least fixed point like in Section 1. The n-th prefix of a causal increasing function is a sequence $(f_i \in A_i \to B_i)_{i \leqslant n}$ each of the f_i being increasing, and such that $r_i(f_{i+1}(a)) = f_i(r_i(a))$.

What remains to be figured out is what a fixed point internal to the topos of causal set is. The internal formula for a fixed point is $f(a) = a$ which translates to $f_n(a_n) = a_n$ for any n (note that f, being an internal function, *i.e.* an element of A^A, has prefixes $(f_i \in A_i \to A_i)_{i \leqslant n}$, so $f_n \in A_n \to A_n$). Therefore a is an internal fixed point if and only if it is a fixed point at each prefix.

[3]The reader may be worried about the implication in the statement of symmetry and transitivity, since the interpretation of implication is not direct in the Kripke-Joyal semantics. But it doesn't matter at "toplevel": $\vdash x \leqslant y \to y \leqslant x$ translates to $\forall n, \alpha_n, \beta_n.\ \forall k \leqslant n.\ k \models r^{n-k}(\alpha_n) \leqslant r^{n-k}(\beta_n) \to k \models r^{n-k}(\beta_n) \leqslant r^{n-k}(\alpha_n)$ which is equivalent to $\forall n, \alpha_n, \beta_n.\ n \models \alpha_n \leqslant \beta_n \to n \models \beta_n \leqslant \alpha_n$.

2.3 Lifting traces

The results of the above section, while elegant, do not demonstrate effectively the usefulness of the topos-theoretic semantics: indeed, the treatment of the previous section could have been carried out directly just as easily without requiring topos-theoretic baggage. However, when all this material is developed, it becomes possible to easily lift more powerful theorems directly from the combinational semantics. Let us apply this principle to Hasegawa's theorem from Section 1.2.

Hasegawa's theorem being an external statement about categories, we will have to translate the statement (but, crucially, not the proofs) of all four lemmas and show that they correspond to the hypotheses of Hasegawa's theorem. Fortunately, this is rendered easy thanks to some standard properties: causal functions are the same as global sections $1 \to B^A$ (where 1 is the terminal causal set: 1_n is the singleton set for every n), internal equality is interpreted as external equality, and the terms $\lambda x.\, x$ and $\lambda x.\, f(g(x))$ are interpreted as identity and composition, respectively. From these, we can immediatly deduce that it makes sense to speak of a causal function which is internally increasing, and therefore, that the causal bounded-height domains with a smallest elements and internally increasing causal functions form a subcategory of the topos of causal sets.

A slightly trickier property is the local fixed-point operator whose existence is proven internally. Remember that $\exists x^{\in A}.\ P(x)$ is interpreted by $\forall n^{\in \mathbb{N}}.\ \exists {x_n}^{\in A_n}.\ P(x_n)$ there are no connection between the x_n chosen at each n so there is not necessarily a global section $1 \to A$ that witnesses the existential. However, when the x not only exists, but is also unique, then, since $P(x_{n+1}) \Rightarrow P(r_n(x_{n+1}))$, then the x_n necessarily respect the restriction maps of A, hence form a global section. There is, of course, at most one local least fixed-point map, hence, internal existence guarantees external existence of a global section, which can be turned into an external map μ from causal functions to causal functions.

The rest of the properties: that μ is, indeed, a local fixed-point map, that it is natural in A and X and that it verifies Bekič's lemma, are all universally quantified equalities involving composition of arrows (and μ). They can be changed into their categorical counterparts with just a bit of fiddling.

We can, therefore, conclude, with barely any proof pertaining to time, that sequential circuits obey the laws of traced categories.

Conclusion

The topos theoretic approach to the theory of sequential circuits could be unfolded and give rise to a semantics free of all things toposes. As I have hinted in the course of this article, the semantics itself would not be particularly complex, however proofs are significantly simpler when making use of the internal logic of the topos (sometimes called the synthetic point of view). Proving that sequential circuits form a traced category, for instance, almost completely ignored the difference between sequential and combinational circuits.

This article can be seen both as a contribution to the growing body of applications of the synthetic approach to mathematical problems, and as a

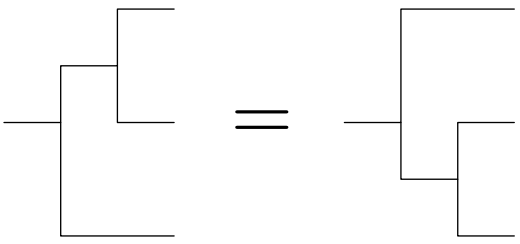
proposal to further the understanding of the mathematics of circuits which, despite being a fundamental concept in computer science is still rather obscure and hard to reason about.

I should mention that the circuits considered are partial, in that they may return $\bot$, an ill-formed value. We are really interested in total circuits which do not; but as is the case for total recursive functions, total circuits have no reason to be composable. Fortunately, it is reasonably easy to characterise total circuits: there is a (natural) causal function $\eta_A : \mathbb{S}_A \to \mathbb{S}_{A_\bot}$ we say that a circuit $c : \mathbb{S}_{A_\bot} \to \mathbb{S}_{B_\bot}$ is total if the composite function $c \circ \eta_A$ factors through η_B, *i.e.* if there is $c' : \mathbb{S}_A \to \mathbb{S}_B$ such that $c \circ \eta_A = \eta_B \circ c'$. This is straightforwardly extended to several inputs and outputs.

Circuits are built by composition and taking a trace. Composition of total circuits is total, only taking a trace can turn a total circuit into a non-total one. The standard way to take a trace safely is to ensure that "somewhere on the path" there is a delay. This condition is captured, in the topos of causal sets, by the notion of contractivity which can be internalised and used synthetically [2]. Contractivity is an example of notion, in the internal logic of the topos of causal sets, which goes beyond standard constructive mathematics. It, indeed, ensures that trace can be taken safely.

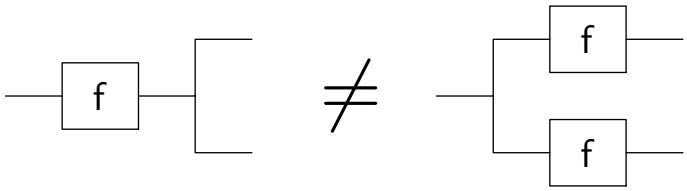
What may render the synthetic approach difficult is, beyond the need to use constructive mathematics, is the translation of a synthetic statement into an ordinary one. It is tedious and precise, and, though it probably gets better with training, it is hard to convince oneself no error has been made in the process. This is where a proof assistant would be of great help, and be much more efficient at such a task than a human. There is a prototype for the Coq proof assistant by Jaber, Sozeau & Tabareau [7] which handles the special case of presheaves over a preorder. It is sufficient for the topos in this article, so I could have used it to help with the translations of Section 2. Since most of Section 1 has already been formalised in Coq[4], it is not particularly far-fetched. However, I have unfortunately not taken time to learn how to use this tool.

To conclude, I feel I should say a few words about syntax, after spending this article on the semantics of circuits. When working with circuits we tend to assume that reorganising of wires preservers syntax so that the following two composition of diagonals are equal:



[4]The formalisation can be found at the following address: `https://gist.github.com/aspiwack/628761dab886728bf4db`

But that rearranging gates does not:



It is customary to take syntax to be a free something, and since our semantics is a traced cartesian category, we may be tempted to take the syntax of circuits to be the free traced cartesian category but that would identify both sides in the latter diagram. Instead, the syntax of circuits should be the free construction of some kind of traced categories with diagonals, where diagonals and augmentations (wires to the empty product) have the usual co-associativity and co-neutrality laws, but are not natural transformations.

Bibliography

[1] John C Baez and Jason Erbele. Categories in control. 2015.

[2] Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8(4), 2012.

[3] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Full Abstraction for Signal Flow Graphs. *ACM SIGPLAN Notices*, 50(1):515–526, 2015.

[4] Thierry Coquand and Arnaud Spiwack. Constructively Finite? In Laureano Lambán Pardo, Ana Romero Ibáñez, and Julio Rubio García, editors, *Contribuciones científicas en honor de Mirian Andrés Gómez*, pages 217–230. Universidad de La Rioja, 2010.

[5] Martin H. Escardo and Thomas Streicher. The universe is indiscrete. 2013.

[6] Masahito Hasegawa. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. *Typed Lambda Calculi and Applications*, 1997.

[7] Guilhem Jaber, Nicolas Tabareau, and Matthieu Sozeau. Extending Type Theory with Forcing. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pages 395–404. IEEE, June 2012.

[8] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(03):447, 1996.

[9] Hai Liu, Eric Cheng, and Paul Hudak. Causal commutative arrows and their optimization. In *Proceedings of the 14th ACM SIGPLAN international conference on Functional programming - ICFP '09*, page 35, 2009.

[10] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory*. Springer, 1992.

[11] Sharad Malik. Analysis of cyclic combinational circuits. *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pages 618–625, 1993.

[12] Michael Mendler, Thomas R. Shiple, and Gérard Berry. Constructive Boolean circuits and the exactness of timed ternary simulation. *Formal Methods in System Design*, 40:283–329, 2012.

[13] Yann Orlarey, Dominique Fober, and Stephane Letz. Syntactical and semantical aspects of Faust. *Soft Computing*, 8(9), July 2004.