Categorical Semantics of Digital Circuits

Ghica, Dan; Jung, Achim

Categorical Semantics of Digital Circuits

Dan R. Ghica
University of Birmingham, UK

Achim Jung
University of Birmingham, UK

***Abstract*—This paper proposes a categorical theory of digital circuits based on monoidal categories and graph rewriting. The main goal of this paper is conceptual: to fill a foundational gap in reasoning about digital circuits, which is currently almost exclusively semantic (simulations). The level of abstraction we target is circuits with discrete signal levels, discrete time, and explicit delays, which is appropriate for modelling a range of components such as boolean gates or transistors working in saturation mode.**

We start with an algebraic signature consisting of the basic electronic components of a given class of circuits and extend it gradually (and in a free way) with further algebraic structure (representing circuit combinations, delays, and feedback), while quotienting it with a notion of equivalence corresponding to input-output observability. Using well-known results about the correspondence between free monoidal categories and graph-like structures we can develop, in a principled way, a graph rewriting system which is shown to be useful in reasoning about such circuits. We illustrate the power of our system by reasoning equationally about a challenging class of circuits: combinational circuits with feedback.

I. Introduction

A. Syntactic vs. semantic modelling

Theories of *programming languages* can be either *syntactic*, formulated equationally, or *semantic*, formulated via translation into a mathematical domain. The former is commonly known as *operational semantics* [1] and the latter as *denotational semantics* [2]. Even though denotational models are attractive conceptually, the difficulties of producing precise ("*fully abstract*") models of realistic programming languages led to a prevalence of operational reasoning methods in practice, for example, when proving compilers correct [3].

In contrast, formal models of digital circuits are[1] virtually exclusively of a denotational nature, where circuits are translated into an executable model (e.g. automata [4]) so that their behaviour can be *simulated*. As far as denotational models go, such translations tend to obscure the structure of the circuit and are more akin to *compilation* than genuine modelling. This is not to say such models are not effective in reasoning about circuits, in fact they are the foundation of a thriving industry which places a premium on correctness. However, the contrast between hardware and software in terms of reasoning techniques is striking.

Our primary motivation is to address a surprising methodological and conceptual gap, the paucity of syntactic models for digital circuits. Our methodology is heavily influenced by developments, in the last decade, in diagrammatic reasoning for a variety of computational models such as quantum computing [5], signal flow [6] or asynchronous circuits [7]. The basic insight of this approach is that the same algebraic structures ("*monoidal categories*") which are very helpful in describing the structure of systems in general [8] also occur in certain diagrams [9]. This approach formalises the intuitive connection between systems and diagrams, also reflected in the rich diversity of graphical environments for circuits and systems (e.g. structural HDLs, Simulink).

B. Methodology and contribution

In terms of level of abstraction we will simply consider digital circuits as broadly construed, i.e. processing a finite number of discrete input and output levels, with explicit and discrete delays down to a smallest observable delay (δ). This level of abstraction applies to Boolean circuits but also to transistor-level modelling, if the transistor operates in non-linear mode.

We will start from a basic algebraic signature meant to represent abstractly the basic components (e.g. transistors) used in the construction of a circuit. Then we will follow a sequence of free categorical constructions and quotientings which will systematically lead to models of digital circuits with delays and feedback. The free constructions represent step-by-step expansions of the algebraic language with new features, whereas the quotientings represent *cutting down* the model by imposing a notion of *observability*. The free constructions have well-known diagrammatic equivalents which will facil-

[1] As far as we are aware.

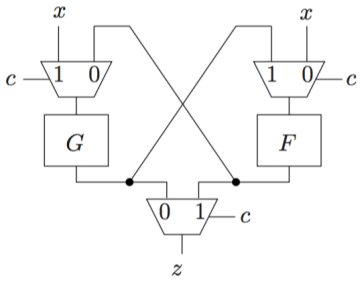


Fig. 1. Combinational circuit with (false) feedback ([10, Fig. 1])

itate reasoning and calculations via diagrammatic graph rewriting.

Our most important guiding principle is the use of "*small*" axioms, i.e. axioms describing the *local* interactions of components, rather than "*coherences*" describing the interplay of complex sub-circuits. We believe this approach to be more physically realistic and more promising for a potential efficient implementation. Coherences, wherever needed, will need to be derived as theorems. The main technical result of the paper is that diagrams of circuits with feedback (*trace*, categorically) have products and therefore feedback can be treated as iteration.

This work was inspired by Berry et. al.'s semantic treatment of cyclic combinational circuits [10], a class of circuits which straddles the combinational-sequential boundary. We aim to provide an equational counterpart to their approach.

C. A motivating class of examples

A particularly intriguing class of circuits are combinational circuits with feedback, as identified by Malik [11]. The presence of feedback suggests that these circuits are *not* combinational yet their behaviour does not lead to any latching effects. Moreover, the feedback loop in purely combinational circuits is banned by the usual synchronous design methodology and cannot be handled by conventional tools. A *semantic* characterisation of such circuits is given in [10]. Our aim is provide an equational method to complement the designer's toolkit of reasoning methodologies.

A simple such circuit is presented in Fig. 1, where a (false) feedback loop is used to create shared data-path resources. The circuit computes $z = \texttt{if } c \texttt{ then } F(G(x)) \texttt{ else } G(F(x))$, which could also be implemented in a cycle-free fashion by duplicating the sub-circuits F and G. Being able to handle such circuits with zero-delay cycles directly has many benefits, as explained in *loc. cit.*, but we target them not particularly for their benefit but for the technical challenge they pose to conventional modelling frameworks.

II. Combinational digital circuits

Our formal language of circuits is based on *category theory*, and in general requires two sorts of variables: *object variables* for labelling (collections of) wires and *morphism variables* for labelling boxes (e.g., gates and circuits). Since we use only one kind of wire we can do away with wire labels and, instead, use just natural numbers to indicate the width of the wire collection (bus). We obtain what is usually called a category of PROducts and Permutations, or PROP for short [12].

Definition 1. *Let* $\mathbf{Circ}$ *be a categorical signature with objects the natural numbers* $\mathbb{N}$ *and a (typically finite) set of morphisms which may be grouped into the following three classes:*

- levels $v : 0 \to 1$*;*
- gates $k : m \to 1$*; and*
- *the special morphisms* join $\mathsf{j} : 2 \to 1$*,* fork $\mathsf{f} : 1 \to 2$*, and* stub $\mathsf{w} : 1 \to 0$*.*

We further assume that there are only finitely many levels and that they form a lattice $(\mathbf{V}, \sqsubseteq)$*. Instead of "level" we will also use "value".*

All circuit signatures include combinators for joining two outputs (*join*) and duplicating an input (*fork*), as well as the ability to discard an output (*stub*). What varies from signature to signature is the number of signal levels we consider, as well as the sets of gates we want to model. Since they form a lattice, the levels must always include a smallest element ($\bot$), corresponding to a disconnected input, and a top element ($\top$) corresponding to an illegal output ("short circuit"). In the simplest and most common instance, the set of level has two other elements, *high* and *low*, but it can go beyond that. For example, in the case of metal-oxide-semiconductor field-effect transistors (MOSFET) it makes sense, in certain designs, to model the diode properties of the transistor by taking into account four levels (strong and weak high and low voltage).

Circuits, in the sense of this paper, are the morphisms of a free categorical construction over their signature. Beginning with *combinational circuits*, the free construction is as follows:

Definition 2. *Let* $\mathbf{CCirc}$ *be the free symmetric*

monoidal category over $\mathbf{Circ}$*, subject to the following additional equations:*

Input-output characterisation of gates *(*extensional completeness*): For any gate* $k\colon m \to 1$*, for any values* $v_i, 1 \leq i \leq m$*, there exists a unique value* v' *such that* $k \circ \bigotimes_{i=1,m} v_i = v'$.

Monotonicity*: For any gate* $k : m \to 1$*, for any values* $v_i, v_i', 1 \leq i \leq m$*, if* $v_i \sqsupseteq v_i'$ *then* $k \circ \bigotimes_{i=1,m} v_i \sqsupseteq k \circ \bigotimes_{i=1,m} v_i'$.

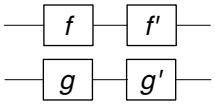
We also require the following equations:

Fork: $\mathsf{f} \circ v = v \otimes v$.
Join*:* $\mathsf{j} \circ (v \otimes v') = v \sqcup v'$.
Stub*:* $\mathsf{w} \circ v = 0$.

The last three encapsulate the understanding that a fork duplicates a value, a join coalesces two values, and a stub discards anything it receives (0 being the identity on the object 0, representing an absence of a wire).

It is known that, in a formal sense, the equality of morphisms in a free symmetric monoidal category (SMC) corresponds to graph isomorphisms in the diagrammatic language [13], where diagrams are created by the operations of sequential composition ($\circ$), parallel composition ($\otimes$) and symmetry ($\mathsf{x}_{m,n}$, the swapping of two buses with m and n wires, respectively), governed by coherence equations. We will usually write composition in diagrammatical order $f \cdot g = g \circ f$. We write the identity (bus of width m) $\mathrm{id}_m : m \to m$ as simply m. For simplicity we also write $\bigotimes_{i=1,m} f = f^m$, $\bigotimes_{i=1,m} f_i = \mathbf{f}$ and $\bigotimes_{i=1,m} v_i = \mathbf{v}$.

Without enumerating them, the equations of SMCs reflect either the fact that the same diagram can be described in two ways or the fact that two diagrams are graph isomorphic. An example of the first case is the *functoriality* of $\otimes$, which gives two equal ways of describing the diagram below.



$$(f \cdot f') \otimes (g \cdot g') = (f \otimes g) \cdot (f' \otimes g').$$

An example of the second kind is the fact that bus-swapping is *involutive*:

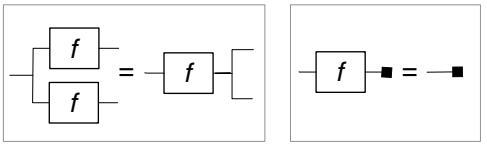
$$\mathsf{x}_{m,m} \cdot \mathsf{x}_{m,m} = 2m$$

The *extensional completeness* principle states that in a circuit the behaviour of a gate must be determined by its input values only. This means that we will deliberately ignore global interactions between components such as electromagnetic interference or quantum tunnelling. *Monotonicity* says that gates must behave monotonically with respect to the ordering of the levels.[2]

We shall see later the importance of these equations. Finally, the last equation represents our commitment to an input-output based notion of equivalence: no matter what value we plug into an unobserved output (*stub*) we get equivalent circuits, so that all circuits with no inputs and no outputs end up being equal.

By simple inductive arguments on the structure of morphisms we can establish that all circuits are extensionally complete, i.e. for any circuit (not just gates) $f : m \to n$, for any values $v_i, 1 \leq i \leq m$, there exists unique values $v_j', 1 \leq j \leq n$ such that $f \circ \bigotimes_{i=1,m} v_i = \bigotimes_{i=1,n} v_i'$. We can further say that two circuits with the same input-output behaviour are *extensionally equivalent*, and we can easily prove, also by structural induction, that this is a *congruence*, i.e. it is an equivalence preserved by sequential and parallel composition. Therefore it makes sense to *quotient* our category $\mathbf{CCirc}$ and create a new category $\mathbf{ECCirc}$ in which equivalent circuits are made equal.

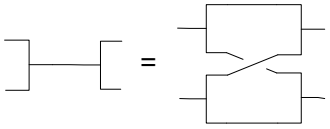
$\mathbf{ECCirc}$ has interesting additional categorical properties which aid reasoning, but two are of particular importance. The first one is that $\mathbf{ECCirc}$ is *Cartesian*, i.e. it has a notion of *product*. The *diagonal* circuit is defined by $\Delta_0 = 0$ and $\Delta_{n+1} = (\Delta_n \otimes \mathsf{f}) \cdot (n \otimes \mathsf{x}_{(1,n)} \otimes 1)$ and it represents the forking of a bus of width n. The diagonal has two important *coherences* represented by the following diagram equalities valid for any diagram $f : n \to m$.



$$\langle f, f \rangle = \Delta_n \cdot (f \otimes f) = f \cdot \Delta_m \qquad f \cdot \mathsf{w}^m = \mathsf{w}^m.$$

[2]Regarding the earlier example of the four values $\{\bot, \mathrm{low}, \mathrm{high}, \top\}$, note that we would not order *low* below *high*; both are valid levels on an equal footing.

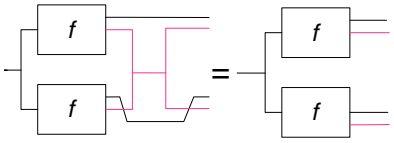
These coherences are immediate by structural induction over f using extensionality. Also using extensionality we can easily show that $(\mathsf{f}, \mathsf{j}, \mathsf{w}, \bot)$ forms what is known as a *Frobenius monoid*, i.e. an algebraic structure in which $(\mathsf{j}, \bot)$ is a commutative monoid, (f, w) is a co-commutative co-monoid, interacting subject to the following law:



$$\mathsf{j} \cdot \mathsf{f} = \mathsf{f}^2 \cdot (1 \otimes \mathsf{x}_{1,1} \otimes 1) \cdot \mathsf{j}^2.$$

In a special context, which will prove to be useful, join and fork behave as if they are inverses.

Proposition 1. *For any $f : m \to n + 1$.*



$$\Delta_m \cdot f^2 \cdot (n \otimes (\mathsf{j} \cdot \mathsf{f}) \otimes n) = \Delta_m \cdot f^2 \cdot (n \otimes 1)^2.$$

Of course, composed the other way round, fork and join are always inverses of each other:

$$\mathsf{f} \cdot \mathsf{j} = 1.$$

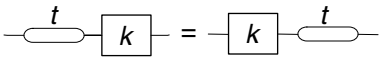
Finally, it will be useful to use a *co-diagonal* which is the joining of two buses of width n, defined as $\nabla_0 = 0$ and $\nabla_{n+1} = (n \otimes \mathsf{x}_{1,n} \otimes 1) \cdot (\nabla_n \otimes \mathsf{j})$. Note that $\nabla_1 = \mathsf{j}$.

III. Circuits with discrete delays

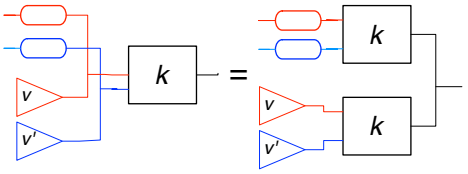
The next step is the free introduction of *delays* which we represent diagrammatically as an elongated oval.

Definition 3. *Let $\mathbf{CCirc}_\delta$ be the category obtained by freely extending $\mathbf{ECCirc}$ with a $\mathbb{Z}$-indexed family of morphisms $\delta_t : 1 \to 1$ such that $\delta_0 = 1$, $\delta_{t+t'} = \delta_t \cdot \delta_{t'}$ and:*

***Timelessness**: For any gate or structural morphism $k : m \to n$ and delay $t \in \mathbb{Z}$, $\delta_t \cdot k = k \cdot \delta_t$:*



***Streaming**: For any levels $\mathbf{v} = v \otimes v'$ and gate k, $(\delta^2 \otimes \mathbf{v}) \cdot \nabla_2 \cdot k = ((\delta^2 \cdot k) \otimes (\mathbf{v} \cdot k)) \cdot \nabla_1$.*



***Disconnect**: $\bot \cdot \delta = \bot$.*
***Unobservable delay**: $\delta \cdot \mathsf{w} = \mathsf{w}$.*

Timelessness means that for any idealised, instantaneous gate or structural morphism (i.e. wire, fork, join, swap, etc.) delaying the inputs by some value t can be compensated by "anti-delaying" the output, by $-t$ [14]. An immediate consequence is that delays can be propagated through combinational circuits, akin to *retiming* [15]. For simplicity we write $\delta_1 = \delta$. Note that any circuit with negative delays, which are not realistic, can be retimed into a realistic circuit by delaying the output:

Theorem 2. *For any circuit $f : m \to n$, there exists $t \in \mathbb{Z}$ such that $f \cdot \delta_t^n$ has no negative delays.*

The proof is immediate by induction on the structure of f and using retiming. We will call the sub-category of circuits without negative delays $\mathbf{CCirc}_+$

Streaming is used to handle waveforms equationally. A waveform of length n is a sequence of n levels $v_n :: v_{n-1} :: \cdots :: v_1$ created using delays and joins, so they always have the form $s_1 = v_1$, $s_{n+1} = (s_n \cdot \delta \otimes v_n) \cdot \mathsf{j}$. The streaming axiom states that to process a waveform we can create two separate instances of a gate, to process the "head" and the "tail" separately, then join the outputs. As far as we know this is a new axiom.

The final two axioms describe the (trivial) interaction between delays and dangling inputs or outputs.

Circuits with delays can also be described extensionally in terms of their input-output behaviour. Because of the presence of delays, now we must use *finite waveforms*, described above and ranged over by s. As before, we write $\bigotimes_{i=i,m} s = s^m$ and $\bigotimes_{i=i,m} s_i = \mathbf{s}$.

Theorem 3 (Extensionality of waveforms)**.** *For any morphism f in $\mathbf{CCirc}_+$ we have that for any input waveform $\mathbf{s}$ there exists a unique output waveform $\mathbf{s}'$ such that $\mathbf{s} \cdot f = \mathbf{s}'$.*

The proof is by induction on the structure of f and

uses routine calculations and the following lemma.

Lemma 4. *For any waveform* $\mathbf{s}$ *of size* n *and for any* $m \geq n$ *there exists a waveform* $\mathbf{s}'$ *of size* m *such that* $\mathbf{s} = \mathbf{s}'$.

The larger waveform $\mathbf{s}'$ is constructed by adding delayed $\bot$ values wherever required. $s_{n+1} = (s_n \otimes (\bot \cdot \delta_n)) \cdot \mathrm{j} = s_n$ from monoid axioms and the *disconnect* axiom.

As in the case of circuits without delays, we can show that extensionality is a congruence and we can quotient by it, creating an *extensional* category of circuits with delays, $\mathbf{ECCirc}_+$.

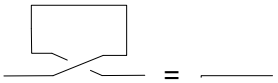
It is a routine exercise to show $\mathbf{ECCirc}_+$ is Cartesian, with the diagonal and terminal object defined the same as in $\mathbf{ECCirc}$, imitating the proof from the previous section.

IV. Circuits with feedback

We can now introduce feedback.

Definition 4. Let $\mathbf{CCirc}^*_\delta$ (and $\mathbf{CCirc}^*_+$, respectively) be the category obtained from $\mathbf{ECCirc}_\delta$ ($\mathbf{ECCirc}_+$, respectively) by freely adding a trace operator.

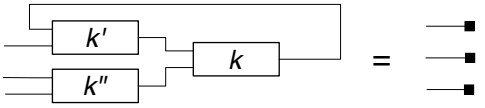
Diagrammatically, the trace operator applied to a diagram $f : m + k \to n + k$ corresponds to a feedback loop of width k, written $\mathrm{Tr}^k(f) : m \to n$. Symmetric traced monoidal categories (STMC) satisfy a number of equations (coherences) which we will not enumerate for lack of space [16]. As before, their interpretation coincides with equality of diagrams (with feedbacks) up to graph isomorphism. For example, of particular interest is the axiom "*yanking*" a loop into a straight line:



$$\mathrm{Tr}^m(\mathsf{x}_{m,m}) = m.$$

This is indeed an axiom that indicates that, conceptually, we are on the right track. The swapping of two wires is a trivial combinational circuit, and applying a trace creates a (false) feedback loop which can be simply eliminated.

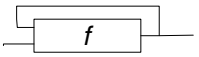
As before, we are committed to an extensional view of circuits where the only observable is the input-output behaviour. In combinational circuits, with or without delays, the only way we can create a circuit with 0 outputs is by explicitly composing a circuit $f : m \to n$ with w^n. However, 0-output circuits can arise in more complicated ways in the presence of feedback, whenever all the outputs are fed back. For example, the diagram on the left can be reduced to just three unobserved inputs:



Such equalities cannot be proved out of local interactions, so we will simply impose the equivalence of all 0-output circuits, an equivalence which is trivially a congruence. The new quotient category is called $\mathbf{OCirc}^*_\delta$. In this category all diagrams of shape $f : m \to 0$ are therefore equal which, categorically speaking, makes 0 a "*terminal object*".

We are now approaching the main result of our paper: reasoning equationally about circuits with feedback. In general, in programs feedback corresponds to recursion and iteration, and syntactic models (operational semantics) of such programs involve creating two copies of the code recursed over. For example, the operational semantics of the Y-combinator as applied to some G is $YG = G(YG)$.

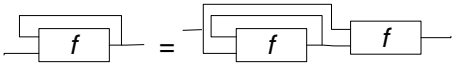
A similar rule does not exist in general for SMTCs unless the category is also Cartesian. Such categories, also called *control-flow categories* [17], admit an *iterator* defined for any $f : m + n \to n$:



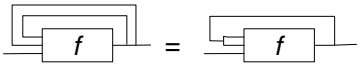
$$\mathrm{iter}^n(f) = \mathrm{Tr}^n(f \cdot (\Delta_n \otimes n)) : m \to n$$

which satisfies the following equations:

Iteration: $\mathrm{iter}(f) = \langle m, \mathrm{iter}(f)\rangle \cdot f$



Diagonal: $\mathrm{iter}^n(\mathrm{iter}^n(f)) = \mathrm{iter}^n((\langle n, n\rangle \otimes m) \cdot f)$.

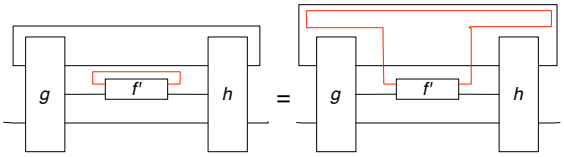


To use this essential axiom we need to first establish that the SMTC of circuits with feedback is Cartesian. This will be the main technical result of this

paper. Before we do that we will establish the following result which holds in general about SMTCs and can be proved by diagrammatic reasoning. Each diagram with feedbacks can be constructed from a feedback-free diagram and one single global trace.

Lemma 5 (Global trace)**.** *For any morphism f in a SMTC PROP there exists a trace-free morphisms $\hat{f}$ such that $f = \mathrm{Tr}^A(\hat{f})$ for some object A.*

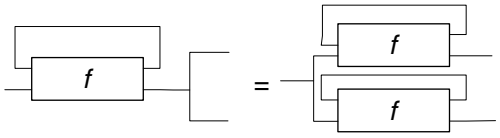
In the case of a PROP the object $A = m \in \mathbb{N}$. The diagram $\hat{f}$ is constructing by "*pulling out*" any internal feedback loop and applying to the whole diagram in a process similar to lambda-lifting. Pictorially, this construction looks as follows:



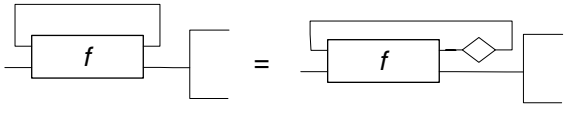
Theorem 6. *The category* $\mathbf{OCirc}_+^*$ *is Cartesian with diagonal* Δ_n*.*

Proof. We need to prove the *naturality* of the diagonal, i.e. for all $f : m \to n$, $f \cdot \Delta_n = \Delta_m \cdot (f \otimes f)$. We use induction on the structure of the diagram f. For all gates and structural morphisms the equation holds because it holds in the category of combinational circuits with delays $\mathbf{ECCirc}_\delta$. Tensor and composition are immediate by induction and simple algebraic calculations.

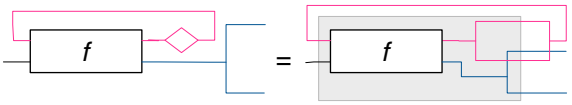
The most interesting case is that of the trace, where we need to show that assuming $f \cdot \Delta_m = \Delta_n \cdot (f \otimes f)$ we have that $\mathrm{Tr}(f) \cdot \Delta_m = \Delta_n \cdot (\mathrm{Tr}(f) \otimes \mathrm{Tr}(f))$:



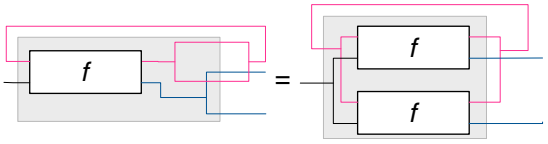
Using the Global Trace Lemma, we can assume that f is trace-free otherwise we can simply incorporate all the internal traces into the global trace. Since $\mathsf{f} \cdot \mathsf{j} = 1$ we have the following equality of diagrams:



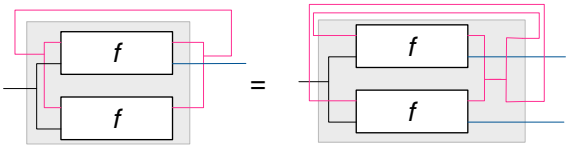
Using coherences (graph isomorphisms) of the SMTC we can rearrange the diagram as follows:



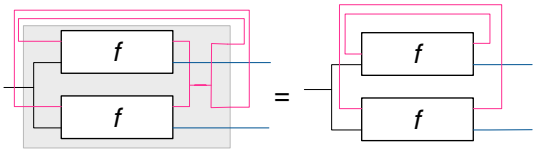
Note that now in the greyed diagram we have the trace-free morphism $f \cdot \Delta_n = \Delta_m \cdot (f \otimes f)$, so by induction hypothesis:



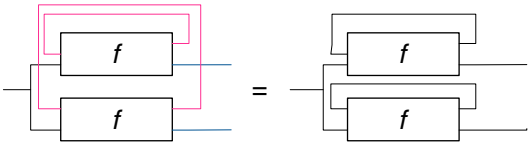
We note the following circuits are graph-isomorphic so they represent equal diagrams in the SMTC:



Noting that the grey sub-diagram is still trace-free we can apply Prop. 1 and obtain the following equal diagram:



The final step is again purely diagrammatic, involving two graph-isomorphic circuits:



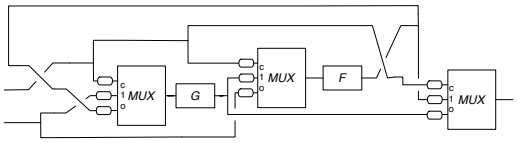
□

V. Equational Reasoning

We have established a comprehensive equational theory which allows us to reason purely syntactically about digital circuits with feedback and discrete delays. We will now apply it to reason about circuits such as the one in Fig. 1. Since that circuit is built using only multiplexers as constants (*gates*, as broadly construed) we are going to consider a categorical signature consisting of one gate (m) and two levels, high (h) and low (l) in addition to the low-impedance ($\bot$) and illegal ($\top$) values. This is for the sake of simplicity, as we could go down to standard boolean gates or even transistors.

The equations describing the multiplexer are the standard ones.

In our categorical notation, the circuit diagram is represented by the following term: $M = \mathrm{Tr}^1\big((\mathsf{x}_{1,1}\otimes\mathsf{f})\cdot(\mathsf{f}\otimes\mathsf{x}_{1,1}\otimes 1)\cdot(\mathsf{f}\otimes(\mathsf{m}\cdot G\cdot\mathsf{f})\otimes 1)\cdot(3\otimes\mathsf{x}_{1,1})\cdot(1\otimes(\mathsf{m}\cdot F)\otimes i)\cdot(\mathsf{x}_{1,1}\otimes 1)\big)\cdot(\mathsf{x}_{1,1}\otimes 1)\cdot\mathsf{m}$. We will also consider that each wire segment has some delay δ_t. By applying retiming we can reduce the number of required delays and obtain the following diagram (the delays may all be different):

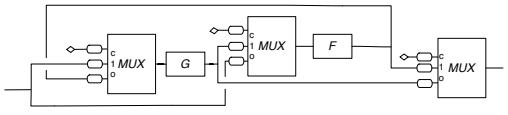


We will prove that if we apply high or low to the input that connects to the control port of the MUX, the resulting circuit is combinational. Given a value v let us define the constant waveform $v^\omega = \mathrm{Tr}((1\otimes\mathsf{f})\cdot(\delta\otimes 1))$.

Example 1. If $s\in\{\mathsf{h}^\omega,\mathsf{l}^\omega\}$, and F, G are combinational circuits then $(v\otimes 1)\cdot M$ is combinational.

Proof. The two cases to consider are all similar. We only show $v=\mathsf{l}$, which is more interesting. We will not show the detailed algebraic calculations but emphasise the diagrammatic reasoning, which is mathematically equivalent and more intuitive. Diagrammatically, we write the constant low (l^ω) as a small diamond.

First we use the axioms for *fork* and *swap* several times, so the diagram reduces to:



At this stage we would like to reason extensionally about the MUXs which receive a low (l^ω) waveform on the control port, but the presence of the delay stops us. We need to use the timelessness of the MUX, and reason as follows:

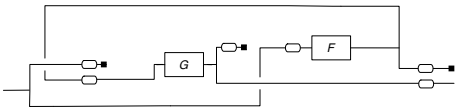
$$\begin{aligned}&(\mathsf{l}^\omega\otimes 2)\cdot(\delta_t\otimes\delta_{t'}\otimes\delta_{t''})\cdot\mathsf{m}\\ =&(\mathsf{l}^\omega\otimes 2)\cdot(\delta_{t-t}\otimes\delta_{t'-t}\otimes\delta_{t''-t})\cdot\mathsf{m}\cdot\delta_t\\ =&(1\otimes\delta_{t'-t}\otimes\delta_{t''-t})\cdot(\mathsf{l}^\omega\otimes 2)\cdot\mathsf{m}\cdot\delta_t\end{aligned}$$

The second step is by functoriality. $(\mathsf{l}^\omega\otimes 2)\cdot\mathsf{m} = \mathsf{w}^2\otimes 1$ by extensional reasoning about trace-free circuits ($\mathbf{ECCirc}_+$). So the above is further equal to

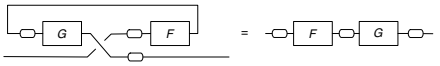
$$\begin{aligned}&=(1\otimes\delta_{t'-t}\otimes\delta_{t''-t})\cdot(\mathsf{w}^2\otimes 1)\cdot\delta_t\\ &=\delta_{t''-t}\cdot\delta_t=\delta_{t''}.\end{aligned}$$

Note that above we may have strayed temporarily outside the safe confines of circuits without negative delays as $t'-t$ and $t''-t$ may be negative! This is not a problem so long as we carefully avoid using properties which only apply to circuits with no negative delays.

Applying this equation for all the MUXs, the diagram becomes:



From here on, using the unobservability of delays on blocked outputs, the fact that (f,w) is a comonoid, and combining delays we get the diagram on the left which can be "yanked" using the STMC coherences:



This is a combinational circuit. The fact that the feedback loop was false is confirmed by the fact that we yanked rather than iterate the trace.

A stronger version of this result is possible, where the input waveforms are arbitrary, but the proof is more complex.

VI. Related and further work

The structured style of presenting digital circuits and some of the equations (e.g. re-timing) have been prefigured by the pioneering work of Sheeran [18], further developed by Luk [19]. Our work represents a categorical systematisation of their approach. Otherwise, there is a dearth of syntactic reasoning methods for digital circuits, but semantic or simulation-based reasoning is extremely broadly studied and very useful.

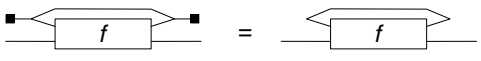
One interesting point of contrast between our approach and more standard approaches is that we introduce the joining of two wires $\mathsf{j}:2\to 1$ as an explicit combinator, which is unusual in Boolean designs but technically essential for us in proving the fact that circuits with feedback have a product and therefore satisfy the iteration equation. Joins are also used in formalising waveforms. The natural interpretation of the wire-join is the value-join in

the lattice of logical levels. Thus, for interesting circuits we require at least four values: $\bot$ (disconnected), h (logical high), l (logical low), $\top$ (illegal). Combining high and low produces an illegal value: $(\mathsf{h} \otimes \mathsf{l}) \cdot \mathsf{j} = \top$. Models of digital circuits requiring ternary logic (unknown, true, false) have been used for a long time [20], but we need the full lattice of values. Having a *join* combinator offers the additional benefit that it allows the representation of sub-logical circuits such as pass-through gates, or even transistors operating in saturation mode. This will be developed in forthcoming papers.

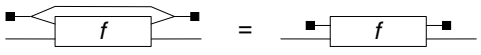
The ternary logic approach is also used by Mendler et. al. [10]. It would be interesting to study whether their semantic model is in fact an alternative (syntax-independent) concrete category of digital circuits, satisfying the same axiom and coherences as ours.

From a mathematical point of view our work is inspired by the deep connections between monoidal categories and diagrams [9] which have been also used in the modelling of quantum protocols [5] and signal-flow graphs [6]. Some contrasts are quite interesting. Unlike in quantum protocols, all digital circuits with no inputs and no outputs are equal whereas in quantum computing they correspond to *scalars*, which allow quantitative aspects to be expressed. Should we have taken a similar direction we could have included quantitative aspects such as power consumption in our formalism, but we would have lost the diagonal property. Obviously, two copies of a circuit will at least sometimes consume more power than one copy!

The signal-flow graph model in [6] is essentially linear and reversible, which is not the case for sequential circuits. Without elaborating the mathematics too much, a key difference between their model and ours can be illustrated by the following equality, involving the interaction between fork, join, and disconnected wires, as a trace can be created out of a fork and a join:



Of course, by comparison, in our setting the directionality of the wires never changes, so the correct equality is:



Acknowledgements: This work was motivated by initial discussions with Michael Mendler. Alex Smith and George Constantinides provided useful comments on preliminary work. Ross Duncan and Peter Selinger helped with technical advice on monoidal categories.

REFERENCES

[1] G. D. Plotkin, "A structural approach to operational semantics," *J. Log. Algebr. Program.*, vol. 60-61, pp. 17–139, 2004.

[2] M. Hennessy, *The Semantics of Programming Languages*. Wiley, 1990.

[3] R. Krebbers, X. Leroy, and F. Wiedijk, "Formal C semantics: Compcert and the C standard," in *5th Int. Conf. Thm. Prov.*, 2014, pp. 543–548.

[4] R. P. Kurshan and K. L. McMillan, "Analysis of digital circuits through symbolic reduction," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1356–1371, 1991.

[5] S. Abramsky and B. Coecke, "A categorical semantics of quantum protocols," in *19th IEEE Symp. Logic in Comp. Sci.*, 2004, pp. 415–425.

[6] F. Bonchi, P. Sobocinski, and F. Zanasi, "Full abstraction for signal flow graphs," in *42nd Ann. ACM Symp. on Princ. of Prog. Lang.*, 2015, pp. 515–526.

[7] D. R. Ghica, "Diagrammatic reasoning for delay-insensitive asynchronous circuits," in *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky*, 2013, pp. 52–68.

[8] J. Baez and M. Stay, *Physics, topology, logic and computation: a Rosetta Stone*. Springer, 2010.

[9] P. Selinger, "A survey of graphical languages for monoidal categories," in *New structures for physics*. Springer, 2010, pp. 289–355.

[10] M. Mendler, T. R. Shiple, and G. Berry, "Constructive boolean circuits and the exactness of timed ternary simulation," *Form. Meth. Syst. Des.*, vol. 40, no. 3, pp. 283–329, 2012.

[11] S. Malik, "Analysis of cyclic combinational circuits," in *Proc. IEEE/ACM Int. Conf. on Comp. Aided Design*, 1993, pp. 618–625.

[12] S. Lack, "Composing PROPs," *Theory and App. of Categories*, vol. 13, no. 9, pp. 147–163, 2004.

[13] A. Joyal and R. Street, "The geometry of tensor calculus, i," *Adv. in Math.*, vol. 88, no. 1, pp. 55–112, 1991.

[14] G. Jones and M. Sheeran, "Timeless truths about sequential circuits," in *Concurrent Computations*. Springer, 1988, pp. 245–259.

[15] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1-6, pp. 5–35, 1991.

[16] A. Joyal, R. Street, and D. Verity, "Traced monoidal categories," in *Math. Proc. of the Cambridge Phil. Soc.*, vol. 119, no. 03. Cambridge Univ. Press, 1996, pp. 447–468.

[17] V. E. Căzănescu and G. Ştefănescu, "Towards a new algebraic foundation of flowchart scheme theory," *Fund. Inf.*, vol. 13, no. 2, pp. 171–210, 1990.

[18] M. Sheeran, "muFP, A language for VLSI design," in *LISP and Func. Prog.*, 1984, pp. 104–112.

[19] W. Luk, "Pipelining and transposing heterogeneous array designs," *J. of VLSI Sig. Proc. Sys.*, vol. 5, no. 1, pp. 7–20, 1993.

[20] M. A. Breuer, "A note on three-valued logic simulation," *IEEE Trans. Comp.*, vol. 21, no. 4, pp. 399–402, 1972.