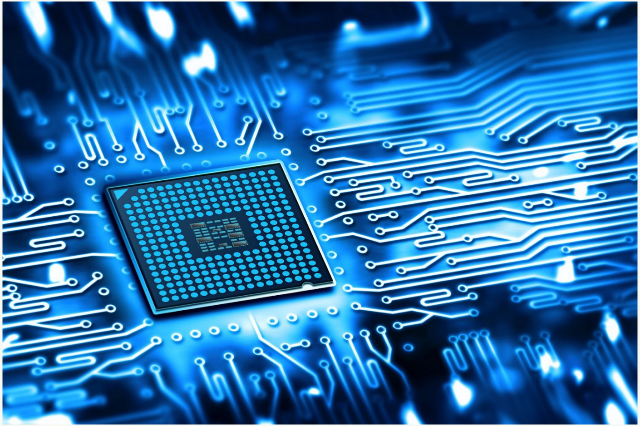
A Fully Compositional Theory of Digital Circuits

George Kaye

University of Birmingham

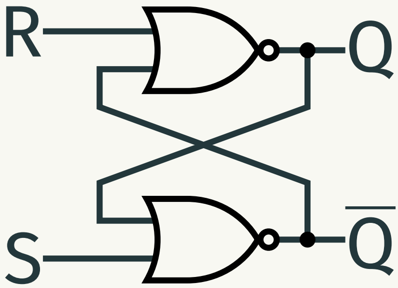
27 October 2023 – SYNCHRON 2023

Digital circuits!



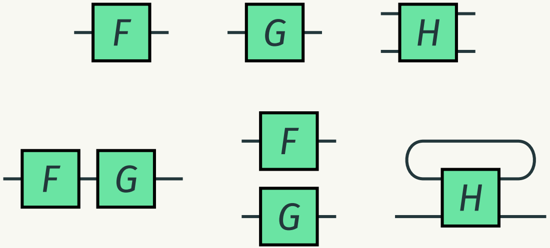
What are we going to be talking about?

Digital circuits!



What are we going to be talking about?

We want a compositional theory of digital circuits.



Why all the pictures?

We want to reason equationally about circuits.

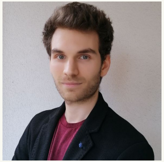
Using string diagrams removes much of the bureacracy

What came before

Lafont (2003) *'Towards an algebraic theory of Boolean circuits'*

Ghica, Jung, Lopez (2017) *'Diagrammatic semantics for digital circuits'*





Joint work with...

Dan Ghica

University of Birmingham

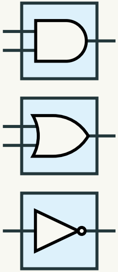
David Sprunger

Indiana State University

Syntax

Combinational circuit components

gates

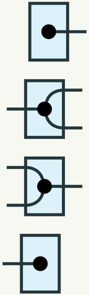


AND gate

OR gate

NOT gate

(co)monoid structure



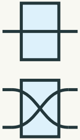
disconnected

fork

join

stub

categorical structure



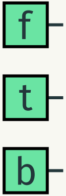
identity

symmetry

Light circuits  only contain gates and structure.

Sequential circuit components

Values



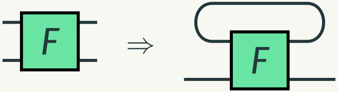
false

true

short circuit

Delay

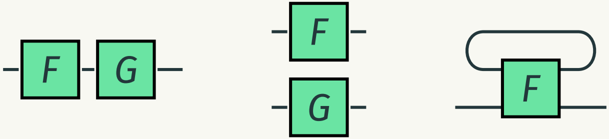
Feedback



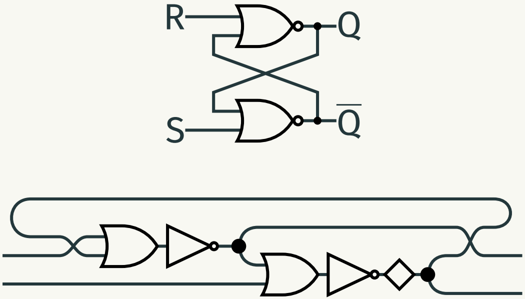
Dark circuits 

may contain delay or feedback.

Circuits are morphisms in a freely generated symmetric traced monoidal category (STMC).



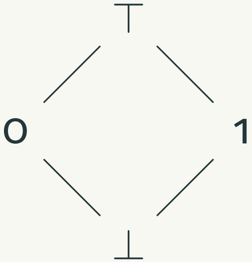
Need an example?

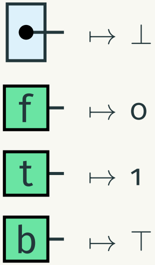


Semantics

We need some meaning

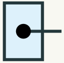
Values are interpreted in a lattice:



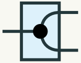


Let's make everything a function

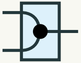
monotone functions $\overline{g} : \mathbf{V}^m \to \mathbf{V}$



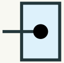
initialise $() \mapsto (\bot)$



copy $x \mapsto (x, x)$



join in the lattice $(x, y) \mapsto x \sqcup y$



discard $x \mapsto ()$

Feedback is interpreted as the least fixed point.

Functions are not enough

How do we model delay?

Streams!

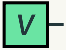
Streams

A stream $\mathbf{V}^\omega$ is an infinite sequence of values.

$$v_0 :: v_1 :: v_2 :: v_3 :: v_4 :: v_5 :: v_6 :: v_7 :: \cdots$$

A stream function $\mathbf{V}^\omega \to \mathbf{V}^\omega$ consumes and produces streams.

$$f(v_0 :: v_1 :: v_2 :: v_3 :: v_4 :: \cdots) = w_0 :: w_1 :: w_2 :: w_3 :: w_4 :: \cdots$$



$$() := v :: \perp :: \perp :: \perp :: \cdots$$

$$(v_0 :: v_1 :: v_2 :: \cdots) := \perp :: v_0 :: v_1 :: v_2 :: \cdots$$

Maybe there are too many streams

Does every circuit correspond to a stream function

$$(\mathbf{V}^m)^\omega \to (\mathbf{V}^n)^\omega?$$

No.

(but this is to be expected!)

Restricting the stream functions

Circuits are causal.

They can only depend what they've seen so far.

Circuits are monotone.

They are constructed from monotone functions.

Is that all? Not quite... (but we'll get there)

Some operations on stream functions

Given a causal stream function $f : (\mathbf{V}^m)^\omega \to (\mathbf{V}^n)^\omega$ and an element $a \in \mathbf{V}^m$...

initial output $\quad f[a] \in \mathbf{V}^n$

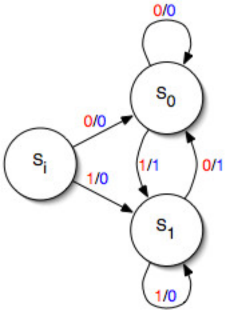
'the first thing f produces given a'

stream derivative $\quad f_a \in (\mathbf{V}^m)^\omega \to (\mathbf{V}^n)^\omega$

'how f behaves after seeing a first'

Hold on, these look familiar...

An old friend



Mealy machines!

Stream functions are the *states* in a Mealy machine.

Circuits have finitely many behaviours

Circuits have a finite number of components.

So there are finite number of states in the Mealy machine.

So the outputs of streams given some input must be periodic.

(There are finitely many stream derivatives).

These are the streams we're looking for

Theorem

A stream function is the interpretation of a sequential circuit if and only if it is ***causal, monotone*** *and has* ***finitely many stream derivatives****.*

The correspondence

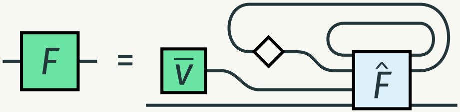
$$
\begin{tikzcd}
& \text{Circuits} \arrow[dl] \arrow[dr] & \\
\text{Stream functions} \arrow[ur, bend left] \arrow[rr] & & \text{Mealy machines} \arrow[ul, bend right] \arrow[ll, bend left]
\end{tikzcd}
$$

Operational semantics

Doing something useful

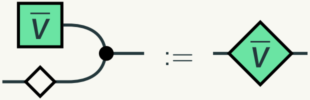
Equational reasoning	**Unstructured pen and paper proofs**
Operational semantics	**Mechanical step-by-step reduction**

Going global



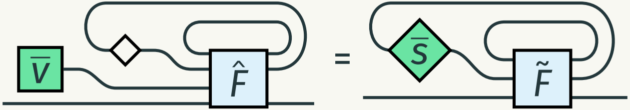
by moving boxes and wires around

Some notation:



'Register'

By adding equations  and 

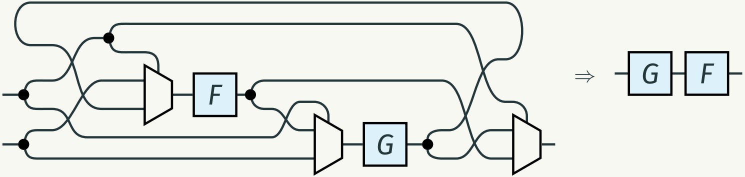


What are we going to do with the non-delay-guarded trace?

Do we even need it?

In industry, normally circuits must be delay-guarded.

But this rules out some clever circuits!



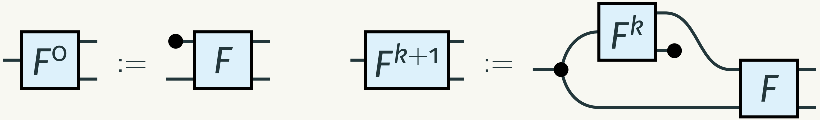
(And also it would be cheating)

Getting rid of non-delay-guarded feedback

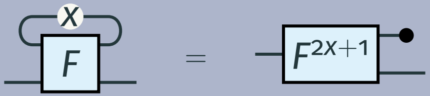
V is a finite lattice...

The functions are monotone...

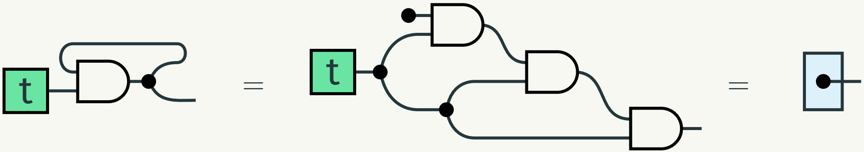
We can compute the least fixed point in finite iterations!



Axiom(s)



Getting rid of non-delay-guarded feedback



Here's Mealy

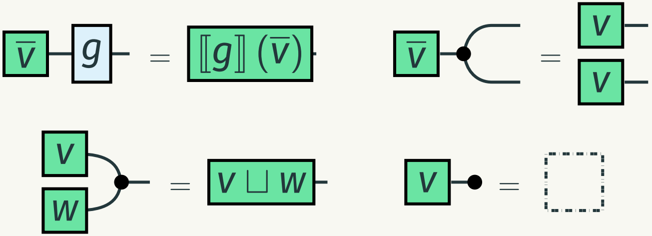
For any circuit

What is the goal

We want to compute the outputs of circuits given some inputs

How does a circuit process a value?

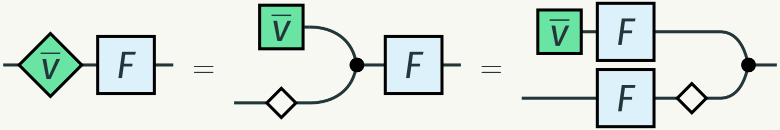
Reducing values



Lemma

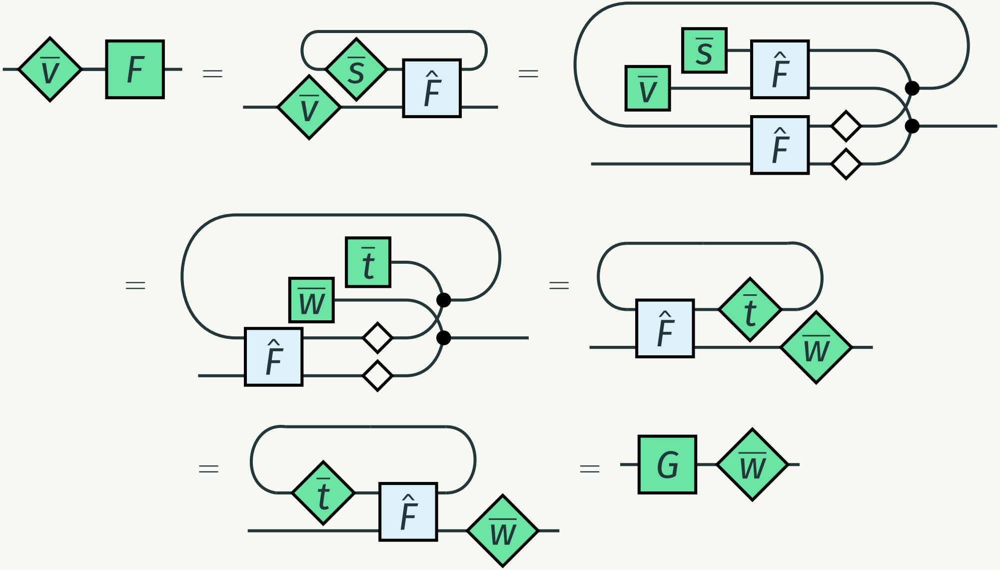
For every $-\boxed{F}-$ *there exists* $\boxed{\overline{w}}-$ *s.t.* $\boxed{\overline{v}}-\boxed{F}- = \boxed{\overline{w}}-$.

What about delays?



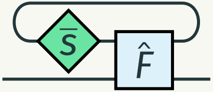
'Streaming'

Catching the jet stream



From diagrams to graphs

Making it combinatorial

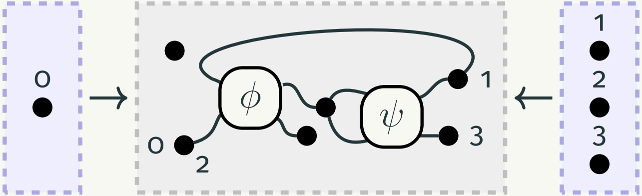


$\Rightarrow$

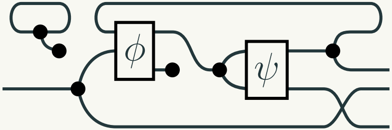
It is hard for computers to work with string diagrams...

...but computers love graphs!

A hyper kind of graph



There are correspondences between certain classes of hypergraphs and circuit string diagrams.



Using the correspondence

$$\begin{tikzcd}
\text{Circuit string diagram} \arrow[r] \arrow[d, equal] & \text{Hypergraph representation} \arrow[d, "\text{Graph rewrite}"] \\
\text{New circuit string diagram} & \text{Rewritten hypergraph} \arrow[l]
\end{tikzcd}$$

Implementing it all

The rewriting framework has been *implemented* in a hardware description language.

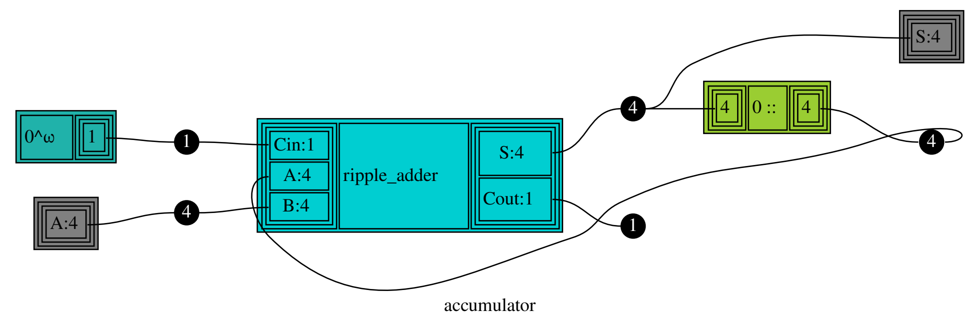
HUAWEI

(the language is still in development)

(so I’ve been warned not to accidentally announce anything)

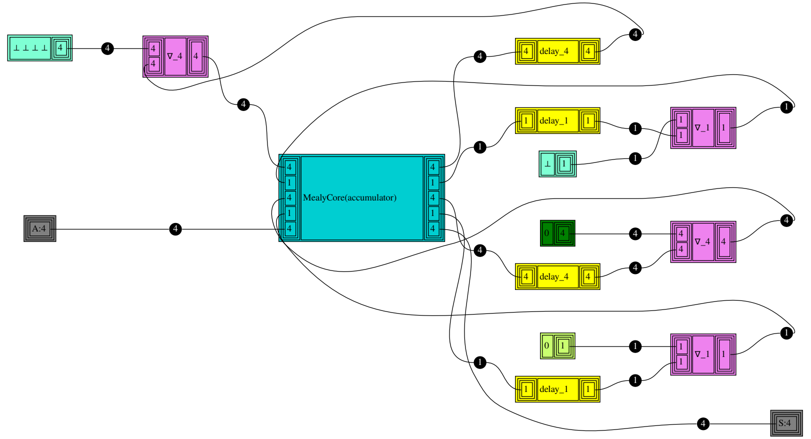
I can still show you something

Create a circuit...



I can still show you something

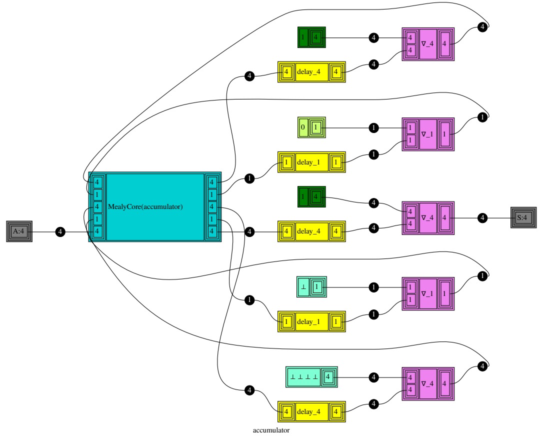
The evaluator converts it into Mealy form...

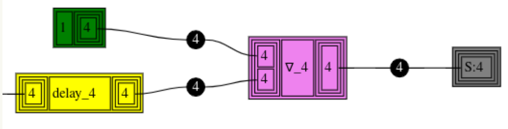


accumulator

I can still show you something

...and then evaluates an input.





Conclusion

We have developed a compositional framework for digital circuits

We have an operational semantics for digital circuits

We can model this using graph rewrites on hypergraphs

This has been implemented in a hardware description language