Full Abstraction for Signal Flow Graphs

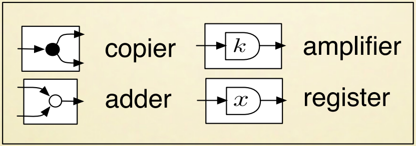
Filippo Bonchi, Paweł Sobociński, **Fabio Zanasi**

ENS DE LYON

ICE 2015

Signal Flow Graphs

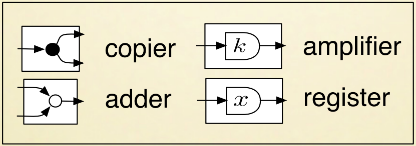
- Signal Flow Graphs are **stream processing circuits** studied in Control Theory since the 1950s.
- Constructed combining four kinds of gate



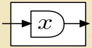
$k \in \mathsf{k}$

Signal Flow Graphs

- Signal Flow Graphs are **stream processing circuits** studied in Control Theory since the 1950s.
- Constructed combining four kinds of gate

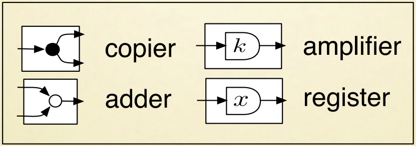


$k \in \mathsf{k}$

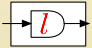


Signal Flow Graphs

- Signal Flow Graphs are **stream processing circuits** studied in Control Theory since the 1950s.
- Constructed combining four kinds of gate

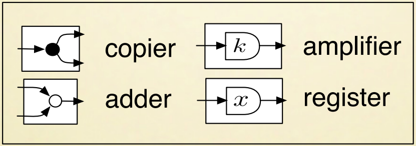


$k \in \mathsf{k}$

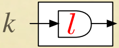


Signal Flow Graphs

- Signal Flow Graphs are **stream processing circuits** studied in Control Theory since the 1950s.
- Constructed combining four kinds of gate

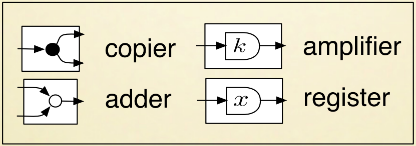


$k \in \mathsf{k}$

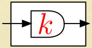


Signal Flow Graphs

- Signal Flow Graphs are **stream processing circuits** studied in Control Theory since the 1950s.
- Constructed combining four kinds of gate



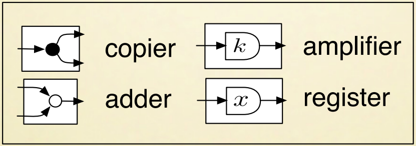
$k \in \mathsf{k}$



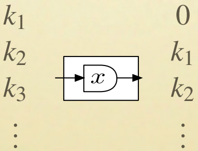
l

Signal Flow Graphs

- Signal Flow Graphs are **stream processing circuits** studied in Control Theory since the 1950s.
- Constructed combining four kinds of gate

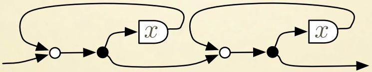


$k \in \mathsf{k}$



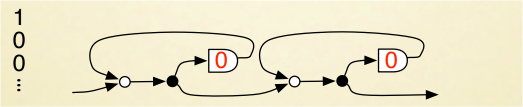
Signal Flow Graphs

An example:



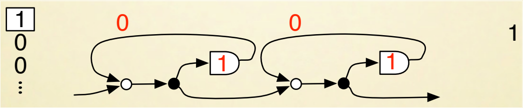
Signal Flow Graphs

An example:



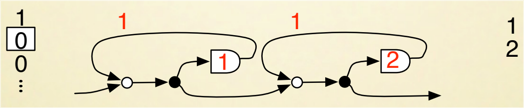
Signal Flow Graphs

An example:



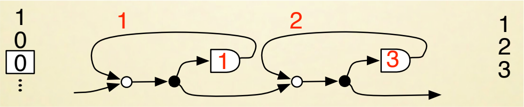
Signal Flow Graphs

An example:



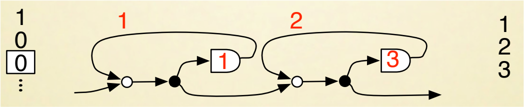
Signal Flow Graphs

An example:



Signal Flow Graphs

An example:



Input 1000… produces 1234….

Signal Flow Graphs

The orthodoxy

- SFGs are not treated as interesting mathematical objects per se.
- Formal analysis typically mean translation into a “lower-level” formalism like systems of linear equations.

Signal Flow Graphs

The orthodoxy

- SFGs are not treated as interesting mathematical objects per se.
- Formal analysis typically mean translation into a “lower-level” formalism like systems of linear equations.

In this work

- An high-level formalism where SFGs are first-class objects:

the calculus of signal flow diagrams

Signal Flow Graphs

The orthodoxy

- SFGs are not treated as interesting mathematical objects per se.
- Formal analysis typically mean translation into a “lower-level” formalism like systems of linear equations.

In this work

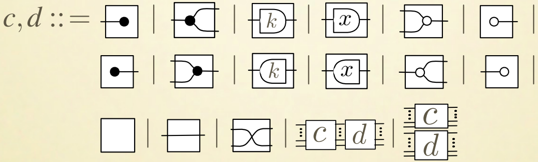
- An high-level formalism where SFGs are first-class objects:

  the calculus of signal flow diagrams

  - String diagrammatic (=graphical) syntax
  - Structural Operational Semantics
  - Denotational semantics
  - Sound and complete axiomatisation
  - Full Abstraction
  - Realisability

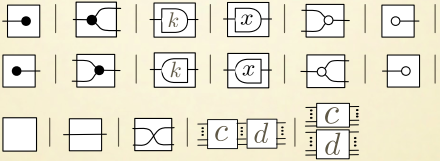
The Calculus of SF Diagrams

Circuit diagrams of Circ are generated by the grammar

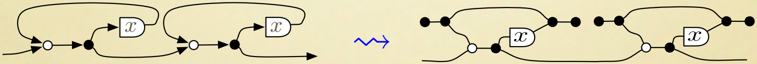


The Calculus of SF Diagrams

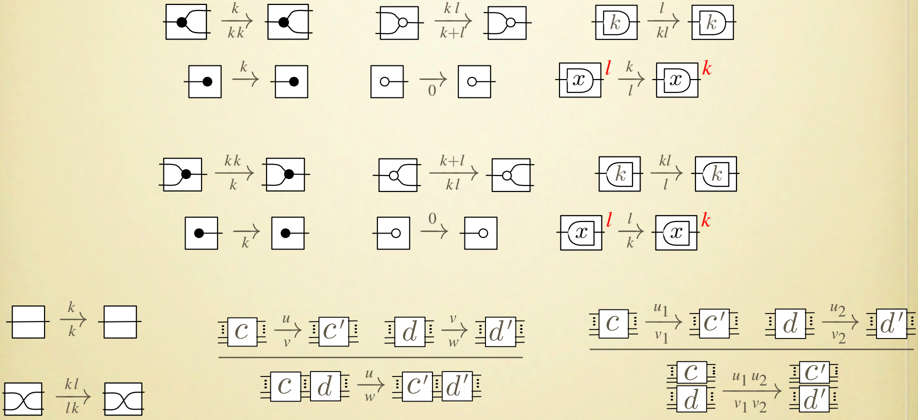
Circuit diagrams of Circ are generated by the grammar

$c, d ::=$ 

We can represent (orthodox) signal flow graphs as circuit diagrams:

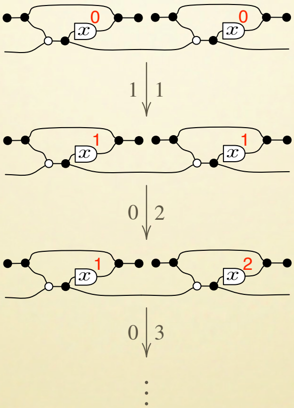


Structural Operational Semantics



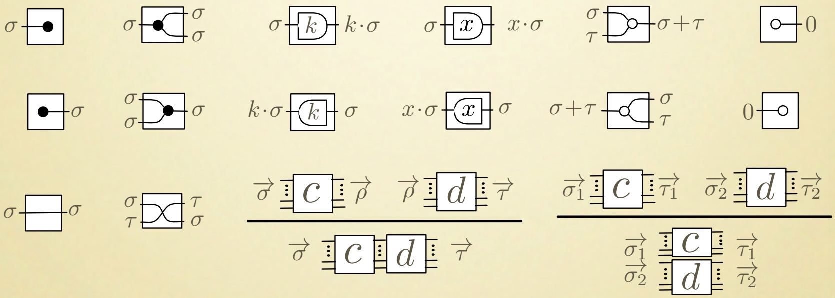
The operational semantics $\langle c \rangle$ is the set of all traces starting from an initial state for c (i.e. one where all the registers are labeled with 0).

Example



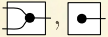
Denotational Semantics

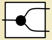
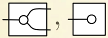
The semantics $[\![\cdot]\!]$ maps a circuit to a linear relation between stream vectors



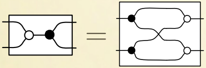
Axiomatisation of $[\![\cdot]\!]$

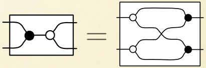
The equational theory of *interacting Hopf algebras* ($\mathbb{IH}$):

– {} and {} form two commutative monoids.

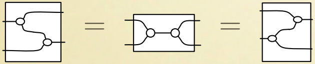
– {} and {} form two commutative comonoids.

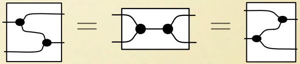
– monoid-comonoid pairs of different colors form Hopf algebras.



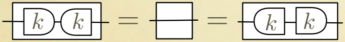


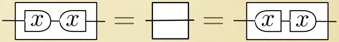
– monoid-comonoid pairs of the same color form Frobenius algebras.





– scalars and delays have formal inverses.





Soundness and Completeness

$$[\![c]\!] = [\![d]\!] \iff c \stackrel{\mathbb{IH}}{=} d$$

Full Abstraction

Theorem (?)

For any c and d in Circ

$$[\![c]\!] = [\![d]\!] \iff \langle c \rangle = \langle d \rangle$$

Full Abstraction

Theorem (?)

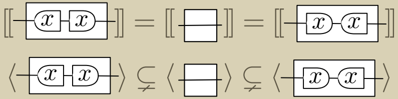
For any c and d in Circ

$$[\![c]\!] = [\![d]\!] \iff \langle c \rangle = \langle d \rangle$$

Not true in general.
The denotational semantics is *coarser* than the operational semantics.

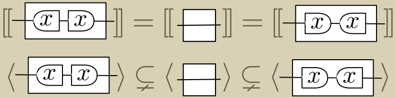
Full Abstraction

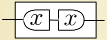
A counterexample

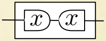


Full Abstraction

A counterexample

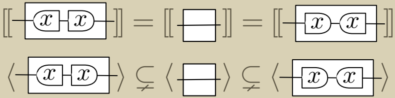


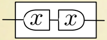


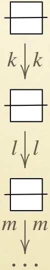


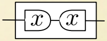
Full Abstraction

A counterexample



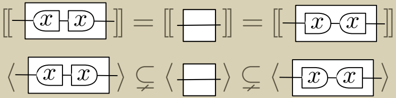


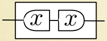


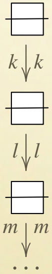


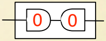
Full Abstraction

A counterexample



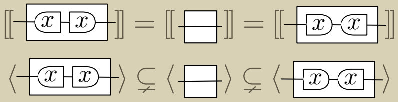


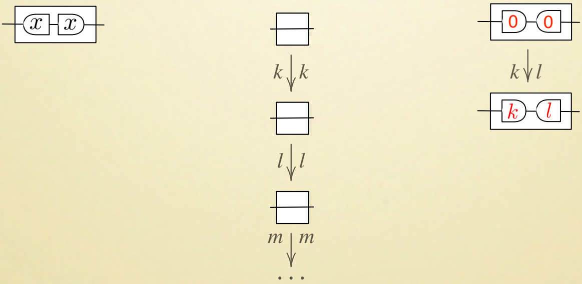




Full Abstraction

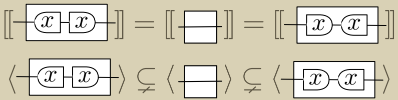
A counterexample

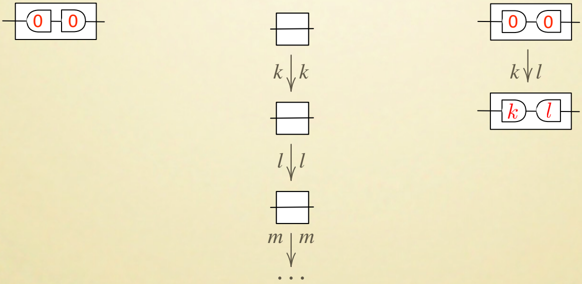




Full Abstraction

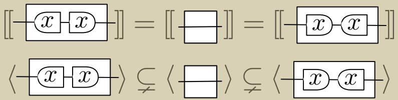
A counterexample

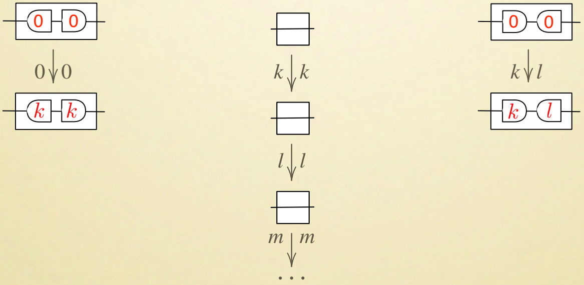




Full Abstraction

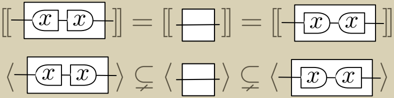
A counterexample

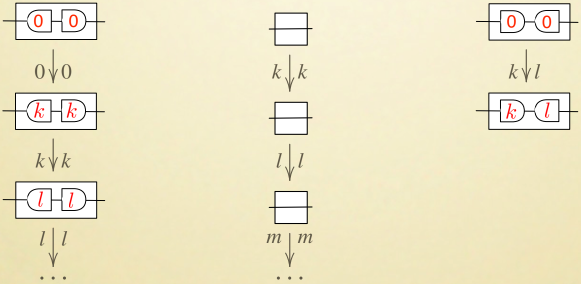




Full Abstraction

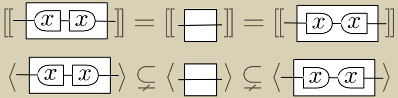
A counterexample

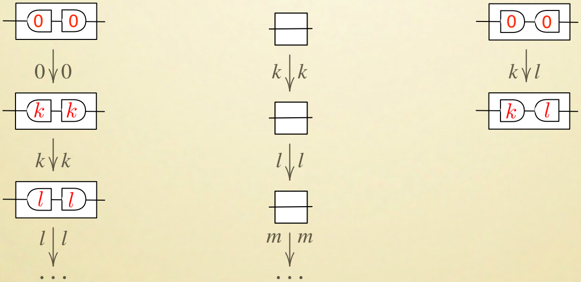


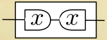
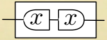


Full Abstraction

A counterexample





We say that  has *deadlocks* and  needs *initialisation*.

Full Abstraction

Theorem

For any c and d in Circ **deadlock and initialisation free**

$$[\![c]\!] = [\![d]\!] \iff \langle c \rangle = \langle d \rangle$$

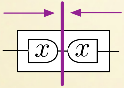
Realisability

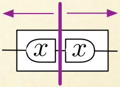
In presence of deadlocks or initialisation, we cannot determine directionality of the flow.

A trace for these circuits cannot be thought as the execution of a state-machine.

Realisability

In presence of deadlocks or initialisation, we cannot determine directionality of the flow.





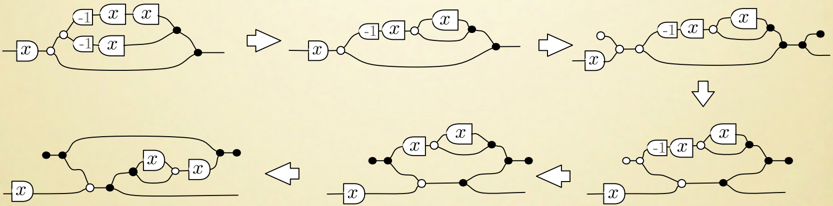
A trace for these circuits cannot be thought as the execution of a state-machine.
However, all the circuit diagrams can be put into an executable form using the equational theory $\stackrel{\mathbb{IH}}{=}$.

Realisability Theorem

For any circuit c of Circ there exists
d deadlock and initialisation free such that $c \stackrel{\mathbb{IH}}{=} d$.

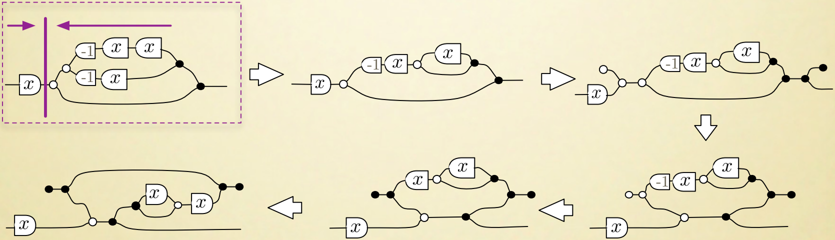
Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



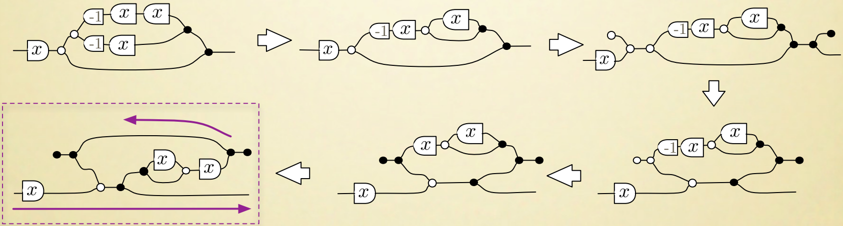
Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



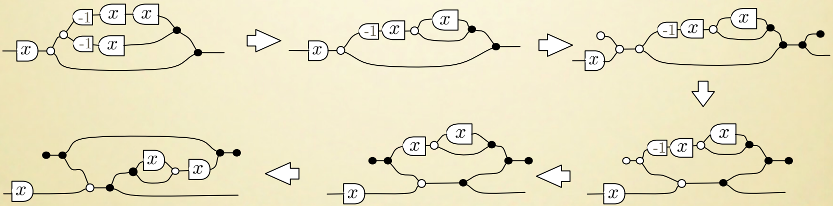
Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



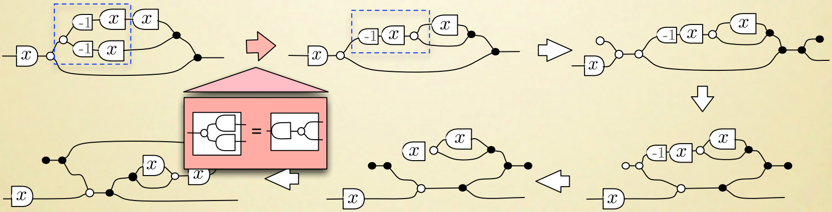
Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



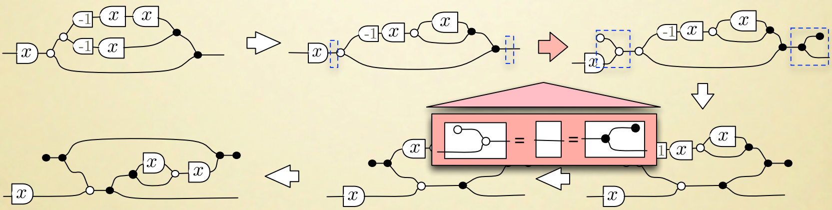
Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



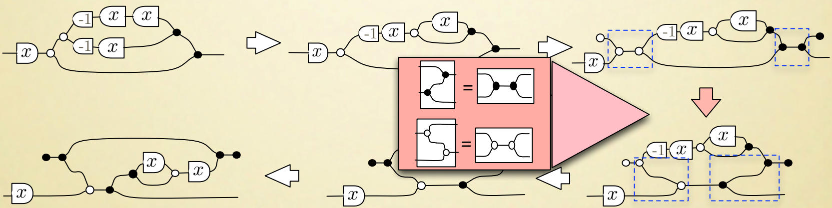
Realisation via IH-rewriting

Implementing the Fibonacci circuit



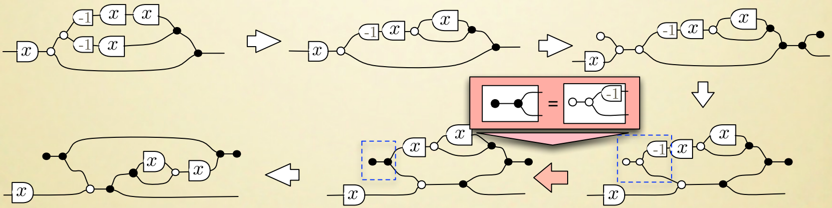
Realisation via IH-rewriting

Implementing the Fibonacci circuit



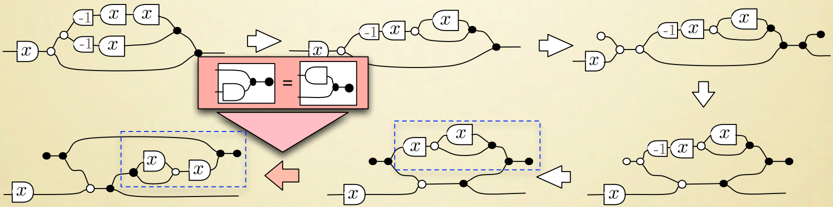
Realisation via IH-rewriting

Implementing the Fibonacci circuit



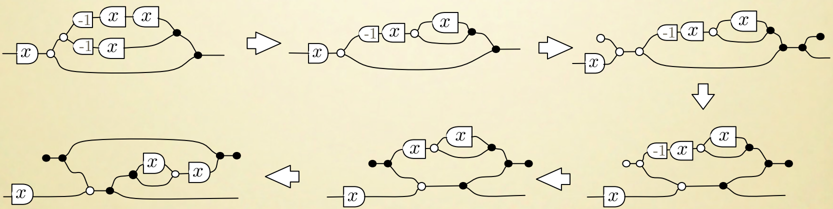
Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



Realisation via $\mathbb{IH}$-rewriting

Implementing the Fibonacci circuit



Conclusions

- The calculus of signal flow diagrams does not rely on flow directionality as a primitive.

  The reason why physics has ceased to look for causes is that in fact there are no such things. The law of causality, I believe, like much that passes muster among philosophers, is a relic of a bygone age, surviving, like the monarchy, only because it is erroneously supposed to do no harm.

  (Bertrand Russell -1913)

- This allows for a more flexible syntax, disclosing a rich and elegant mathematical playground: $\mathbb{IH}$.
- Whenever flow directionality matters, the realisability theorem allows us rewrite any circuit diagram into an executable form.