Foundations of Network Diagrams: Dynamical Systems, Bayesian Networks and Quantum Processes

Filippo Bonchi
University of Pisa
Quantum Teleportation
Quantum Teleportation

1932: von Neumann’s original formulation of quantum theory based on Hilbert spaces
Quantum Teleportation

1932: von Neumann’s original formulation of quantum theory based on Hilbert spaces

Quantum Teleportation

1932: von Neumann’s original formulation of quantum theory based on Hilbert spaces

1935: EPR weirdness of non-locality: "spooky action at distance"

Quantum Teleportation

1932: von Neumann’s original formulation of quantum theory based on Hilbert spaces

1935: EPR weirdness of non-locality: "spooky action at distance"

1993: Bennet et al. conceived the feature of quantum teleportation.

Quantum Teleportation

1932: von Neumann’s original formulation of quantum theory based on Hilbert spaces

1935: EPR \textit{weirdness} of non-locality: "spooky action at distance"

1993: Bennet et al. conceived the \textit{feature} of quantum teleportation.

Why did it take so long?
Quantum Pictorialism
Quantum Pictorialism

Reasoning about quantum systems via Hilbert spaces is rather inconvenient, pretty much like programming a distributed application in Assembly.
Quantum Pictorialism

Reasoning about quantum systems via Hilbert spaces is rather inconvenient, pretty much like programming a distributed application in Assembly.

\[
\frac{1}{4} \begin{pmatrix}
-1+i & 1+i & 1+i & -1+i & 1+i & 1-i & 1-i & 1+i \\
1+i & 1-i & 1-i & 1+i & -1+i & 1+i & 1+i & -1+i \\
1-i & -1-i & 1-i & 1+i & 1-i & -1-i & -1-i & 1-i \\
1+i & 1-i & 1-i & 1+i & 1-i & 1-i & 1-i & 1+i \\
1-i & -1-i & 1-i & 1+i & 1-i & 1-i & 1-i & 1+i \\
-1+i & 1+i & 1+i & -1+i & 1+i & 1-i & 1-i & 1+i \\
1+i & 1-i & 1-i & 1+i & -1+i & 1+i & 1+i & -1+i
\end{pmatrix}
\]

vs.

Figure 1.6 Contrasting a low-level and a high-level language for computer programs. The programs on the left and right perform the same task, but one is written in the low-level x86 assembly language, and one in the high-level language Ruby.

Figure 1.7 Contrasting a low-level and a high-level language for quantum processes, just like we contrasted the low-level and a high-level representation for digital data in Fig. 1.5, and a low-level and a high-level programming language in Fig. 1.6.

as well, for example, modelling meaning in natural language (Fig. 1.8), doing proofs in formal logic, control theory, and modelling electrical circuits. Diagrams are also becoming increasingly important in some fancy research areas of pure mathematics, such as knot theory, representation theory and
Quantum Pictorialism

Reasoning about quantum systems via Hilbert spaces is rather inconvenient, pretty much like programming a distributed application in Assembly.

Developing an high level language for quantum system would boost the discovery of quantum features and the development of quantum technologies.
Quantum Pictorialism

Reasoning about quantum systems via Hilbert spaces is rather inconvenient, pretty much like programming a distributed application in Assembly.

Developing an high level language for quantum system would boost the discovery of quantum features and the development of quantum technologies.
Network diagrams

- Electrical Circuits
- Bayesian Networks
- Quantum Processes
- Petri Nets
- Signal Flow Graphs
Network diagrams

Signal Flow Graphs
Network diagrams

Diagrammatic languages are not really made of syntax.
Network diagrams

Diagrammatic languages are not really made of syntax.

We are able to describe the behaviour of the whole systems
Network diagrams

Diagrammatic languages are not really made of syntax.

We are able to describe the behaviour of the whole systems

But not the behaviour of the single components
Network diagrams

Diagrammatic languages are not really made of syntax.

We are able to describe the behaviour of the whole systems

But not the behaviour of the single components
Network diagrams

Diagrammatic languages are not really made of syntax.

We are able to describe the behaviour of the whole systems

But not the behaviour of the single components

The behaviour of the whole system should be "reducible" to the behaviour of its components
Network diagrams

Diagrammatic languages are not really made of syntax.

We are able to describe the behaviour of the whole systems

But not the behaviour of the single components

The behaviour of the whole system should be "reducible" to the behaviour of its components

https://www.azimuthproject.org/azimuth/show/Network+theory
Compositional Modelling

There is an emerging, multi-disciplinary field aiming at studying different sorts of networks \textit{compositionally}, inspired by the \textbf{algebraic methods} of programming language semantics.

Diagrams are first-class citizens of the theory. The appropriate algebraic setting is \textbf{monoidal} (and not \textbf{cartesian}) categories.
Compositional Modelling

There is an emerging, multi-disciplinary field aiming at studying different sorts of networks \textit{compositionally}, inspired by the \textbf{algebraic methods} of programming language semantics.

Diagrams are first-class citizens of the theory. The appropriate algebraic setting is \textbf{monoidal} (and not \textbf{cartesian}) categories.

Signal Flow Graphs

Signal Flow Graphs are **stream** processing circuits widely adopted in Control Theory and Signal Processing.

Signal Flow Graphs

Signal Flow Graphs are stream processing circuits widely adopted in Control Theory and Signal Processing.

\[ c, d ::= \quad \begin{array}{cccc}
\bullet & \circ & \boxed{k} & \boxed{x} \\
\boxed{c} & \boxed{d} & \boxed{c} & \boxed{d}
\end{array} \]

Sound and Complete Axiomatisation for Signal Flow Graphs
Sound and Complete Axiomatisation for Signal Flow Graphs

https://graphicalinearalgebra.net
These axioms are almost the same as those for Quantum mechanics.
These axioms are almost the same as those for Quantum mechanics.

What is going on?

https://graphcallinearalgebra.net
References


• Bonchi, Gadducci, Kissinger, Sobocinski, Zanasi - *Rewriting modulo symmetric monoidal structure* - LICS 2016.

