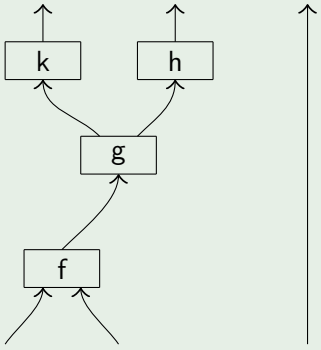
!-graphs with trivial overlap are context-free

Aleks Kissinger Vladimir Zamdzhiev

Department of Computer Science,
University of Oxford

April 13, 2015

Example



- First introduced by Roger Penrose in 1971 as alternative to the tensor-index notation used in theoretical physics.
- (Typed) nodes connected via (typed) wires
- Wires do not have to be connected to nodes at either end
- Open-ended wires serve as inputs/outputs
- Emphasis on compositionality

String diagram applications

Applications in:

- Monoidal category theory (sound and complete categorical reasoning)

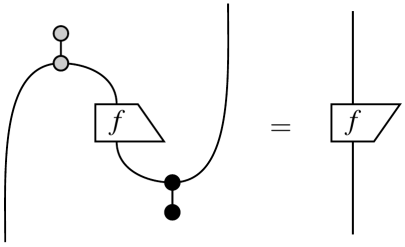


Figure: J. Vicary, W. Zeng (2014)

- Quantum computation and information (graphical calculi, e.g. ZX-calculus)

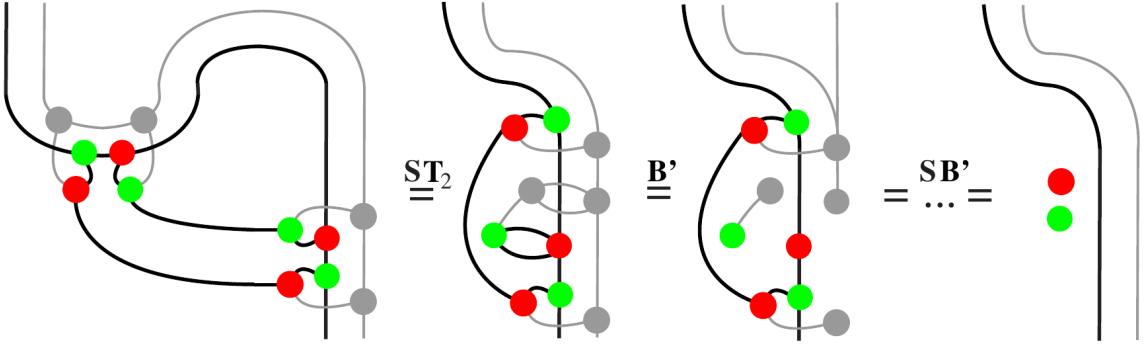


Figure: B. Coecke, R. Duncan (2011)

- Concurrency (Petri nets)

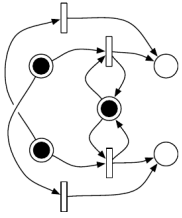


Figure: P. Sobocinski (2010)

- Computational linguistics (compositional semantics)

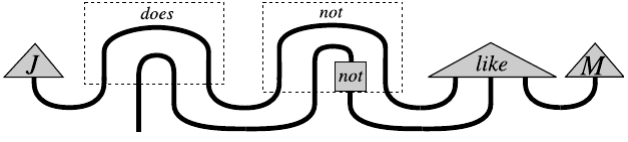


Figure: B. Coecke, E. Grefenstette, M. Sadrzadeh (2013)

- Control theory (signal-flow diagrams)

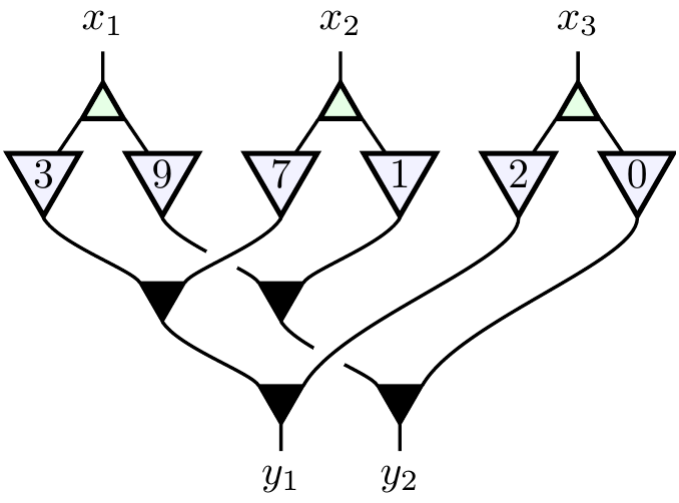


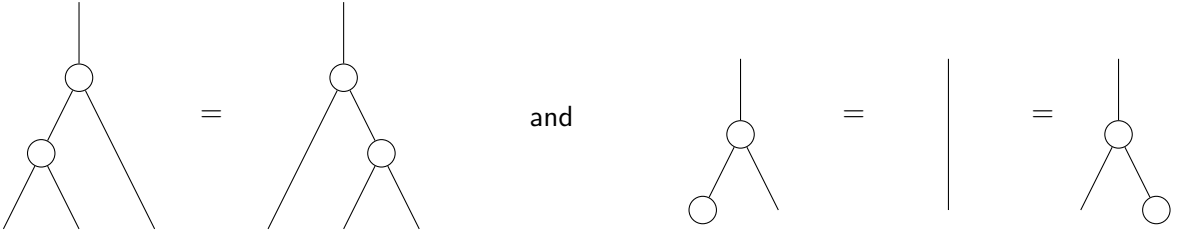
Figure: J. Baez, J. Erbele (2014)

String Diagram Example

A monoid is a triple $(A, \cdot, 1)$, such that:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \quad \text{and} \quad 1 \cdot a = a = a \cdot 1$$

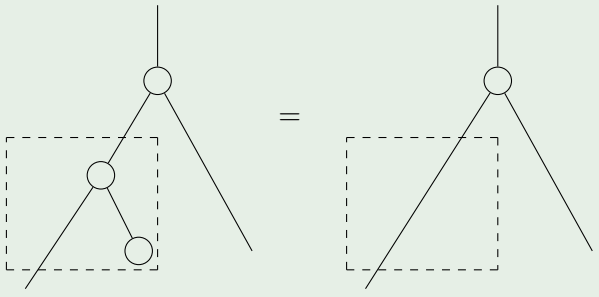
We can model this by setting the binary operation to be  and the unit to be  . Then, the equations become:



String Diagram Example

Equational reasoning is performed by replacing subdiagrams:

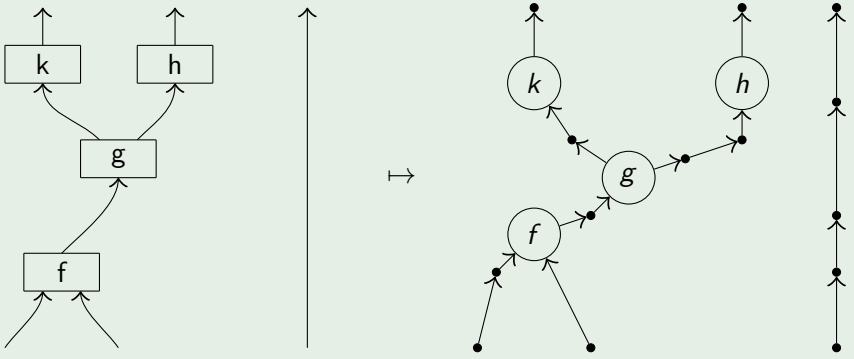
Example



String Graphs

- String diagrams are formally described using (non-discrete) topological notions
- This is problematic for computer implementations
- Discrete representation exists in the form of *String Graphs*
- String graphs are typed (directed) graphs, such that:
  - Every vertex is either a *node-vertex* or a *wire-vertex*
  - No edges between node-vertices
  - In-degree of every wire-vertex is at most one
  - Out-degree of every wire-vertex is at most one

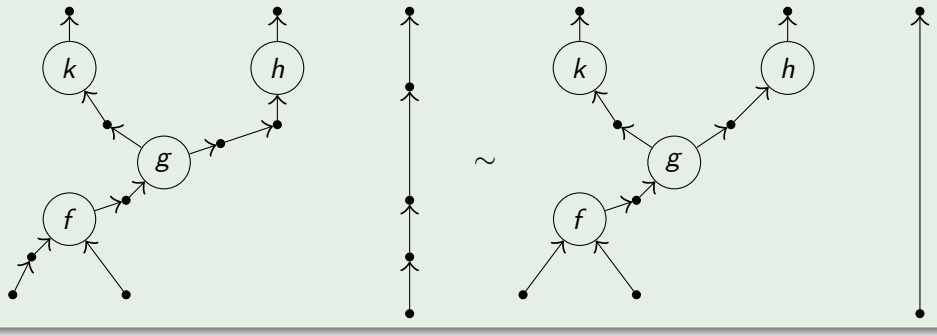
Example



Wire-homeomorphism

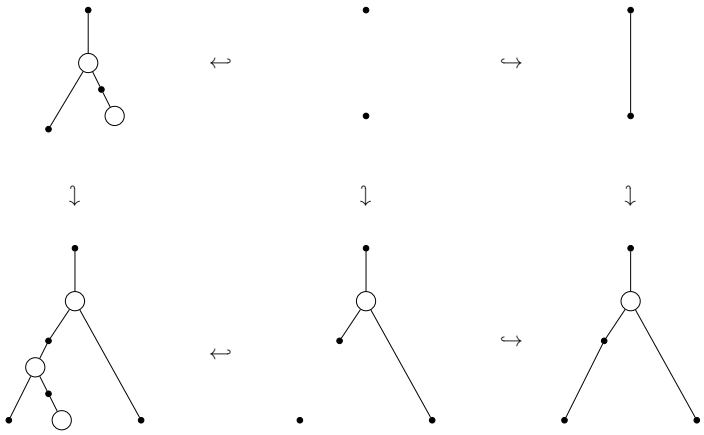
We consider two string graphs to be equal if they are *wire-homeomorphic*, that is, we can obtain one from the other by increasing or decreasing the length of chains of wire-vertices.

Example



By utilising wire-homeomorphism we can simulate string diagram matching and rewriting.

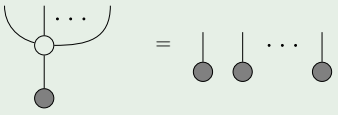
We use double-pushout (DPO) rewriting on string graphs to represent string diagram rewriting:



Families of string diagrams

- String diagrams (and string graphs) can be used to establish equalities between pairs of objects, one at a time.
- Proving infinitely many equalities simultaneously is only possible using metalogical arguments.

Example

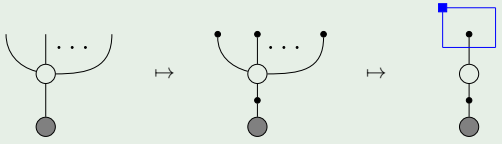


- However, this is imprecise and implementing software support for it would be very difficult.

!-graphs

- A *!-graph* is a generalised string graph which allows us to represent an infinite family of string graphs in a formal way.
- Marked subgraphs called *!-boxes* can be repeated any number of times.

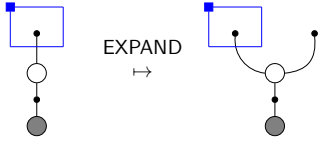
Example



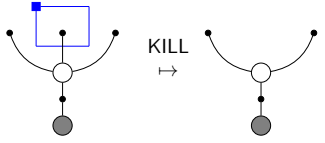
- Semantically, a !-graph should be thought of as an infinite set of concrete string graphs, each of which is obtained after a finite application of two different operations on the !-boxes of the graph.

!-box operations

Applying an EXPAND operation creates a new copy of the subgraph in the !-box which is connected in the same way to the neighbourhood of the !-box:

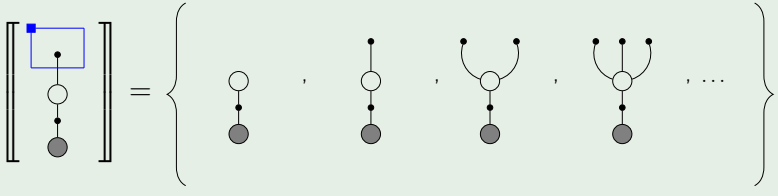


Applying a KILL operation removes a !-box and its contents:



Semantically, a !-graph represents the infinite set of concrete string graphs obtained after applying all possible sequences of !-box operations.

Example



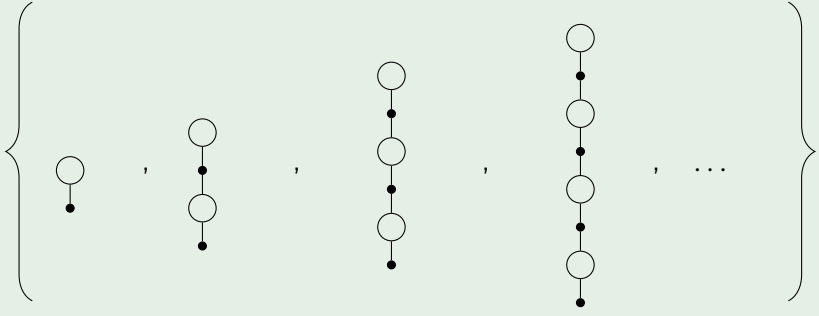
Proposition

The language induced by any !-graph is of bounded diameter.

This limits the expressiveness of !-graphs and there are families of string graphs of interest which we cannot represent using !-graphs.

Example

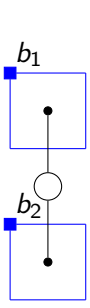
The following language is not induced by any !-graph:



Because of the limitations in expressiveness, we consider alternative language generating mechanisms and try to establish the relationship between them.

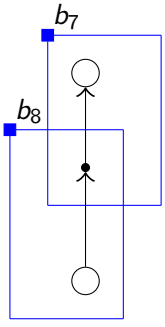
!-box relationships

A !-graph can have multiple !-boxes. The possible relationships between a pair of !-boxes are the following:



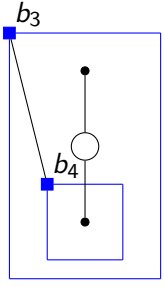
trivial overlap

✓



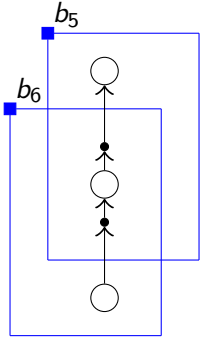
trivial overlap

✓



nested

✓



non-trivial overlap

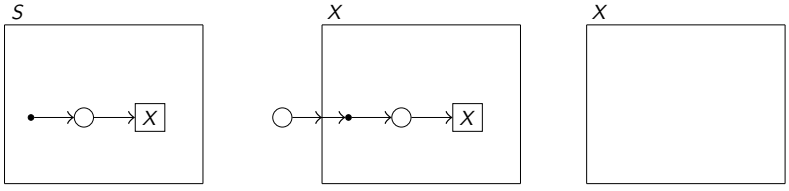
✗

Context-free graph grammars

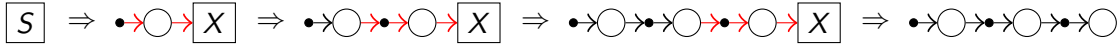
- We investigate context-free graph grammars first, as they have better structural, complexity and decidability properties compared to other more expressive graph grammars.
- Most studied context-free graph grammars are:
  - Hyperedge replacement grammars (HR)
  - Vertex replacement grammars (VR)
- Large body of literature available for both VR and HR grammars
- VR grammars (also known as C-edNCE grammars) are more expressive than HR grammars in general
- We will be working with VR grammars only, in particular linear VR grammars (LIN-edNCE)

VR grammar example

The following grammar generates the set of all chains of node vertices with an input and no outputs:



A derivation in the above grammar of the string graph with three node vertices:



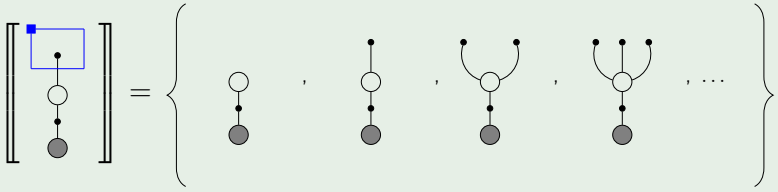
where we color the newly established edges in red.

Theorem

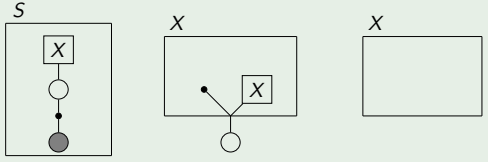
The language induced by any !-graph with no overlapping !-boxes can be generated by a LIN-edNCE grammar. Moreover, this grammar can be constructed effectively.

Example

The language induced by the following !-graph:



can be generated by the following LIN-edNCE grammar:



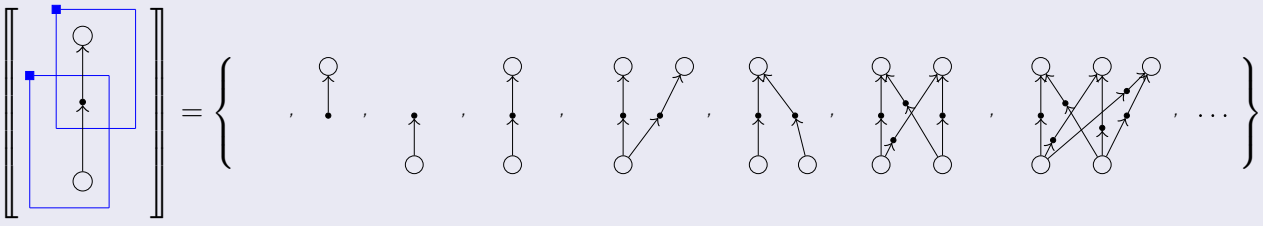
What happens when we allow overlapping !-boxes?

Proposition

A string graph language L can be generated by a VR grammar iff it can be generated by an HR grammar.

Corollary

The language induced by the following !-graph:



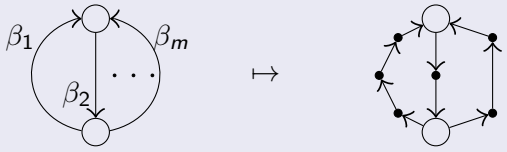
with (trivially) overlapping !-boxes cannot be directly generated by any VR grammar.

Proof.

Follows from a simple application of the pumping lemma for HR grammars. □

Definition (Wire-encoding)

We say that two graphs H and H' are equal up to *wire-encoding*, if we can get one from the other by replacing every edge with special label β_k by a closed wire with endpoints the source and target of the original edge.



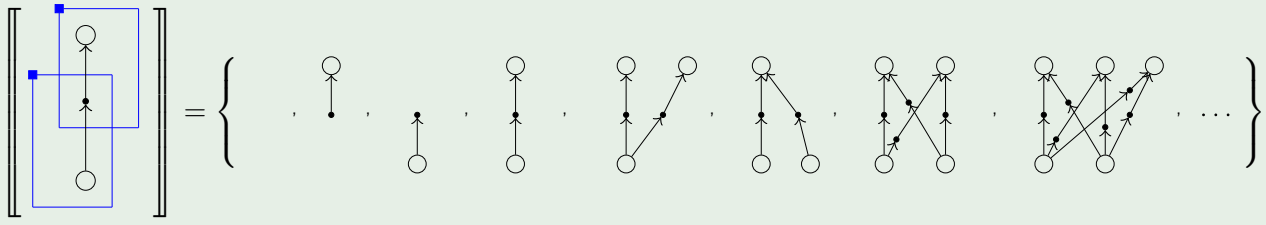
The decoding in the above definition can be formally achieved using a very simple system of DPO rewrite rules.

Theorem

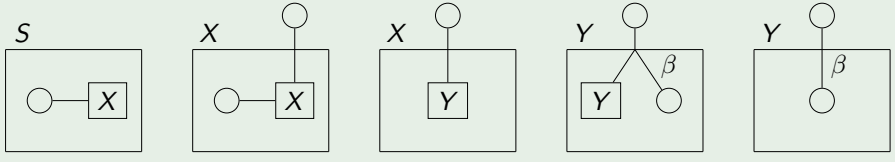
Given a !-graph H such that the only overlap between !-boxes in H is trivial, then there exists a LIN-edNCE grammar which generates the same language as H, up to wire-encoding. Moreover, this grammar can be effectively generated.

Example

The language induced by the following !-graph:

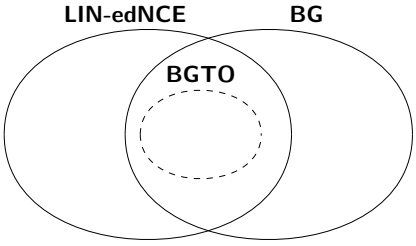


with (trivially) overlapping !-boxes is generated up to wire-encoding by the following LIN-edNCE grammar:



Conclusion

- We have shown the following relationship between !-graphs and LIN-edNCE grammars:

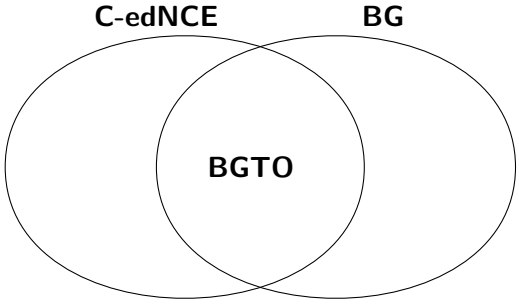


- Moreover, the proofs of the theorems are constructive and translating a !-graph into a LIN-edNCE grammar can be automated.

Conjecture

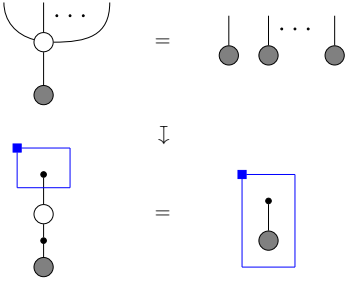
The language induced by any !-graph which contains !-boxes whose overlap is non-trivial cannot be described by a C-edNCE grammar, even up to wire-encoding.

If the conjecture is true, then the relationship simplifies to:

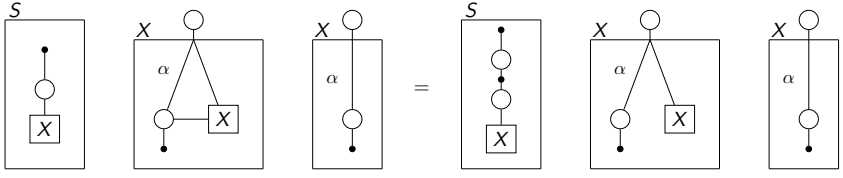


Future work

- !-graphs can be used to formally establish infinitely many equalities via !-graph rewrite rules:



- In recent work, we showed that VR grammars can also be used to achieve the same goal:



- Future work will involve combining the two results in the hope of showing that VR grammars can be used to represent all !-graph rewrite rules