A Framework for Rewriting Families of String Diagrams

Vladimir Zamdzhiev

Department of Computer Science
Tulane University

TERMGRAPH 2018
7 July 2018

Introduction

- String diagrams have found applications in many areas (quantum computing, petri nets, etc.).
- Equational reasoning with string diagrams may be automated (Quantomatic).
- Reasoning for *families* of string diagrams is sometimes necessary (verifying quantum protocols/algorithms).

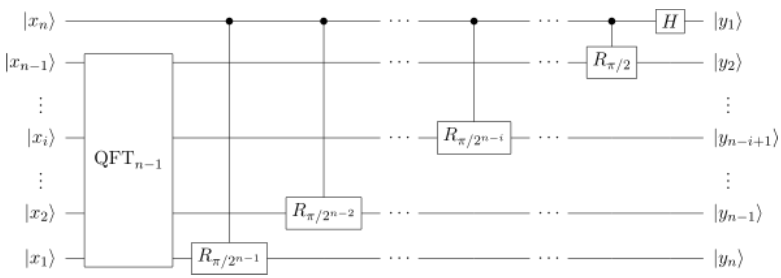
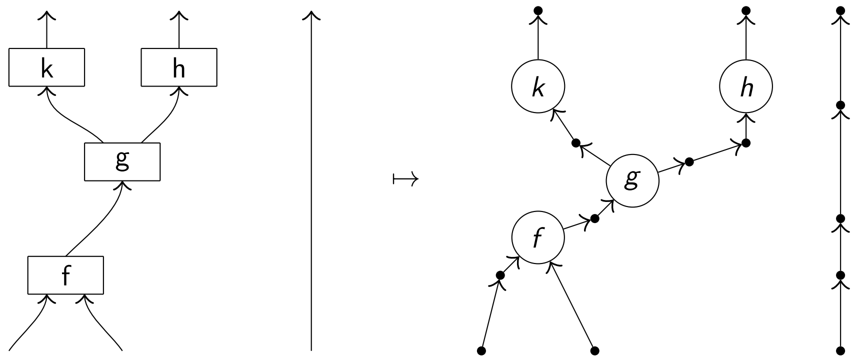


Figure: The Quantum Fourier Transform depicted as a family of quantum circuits.

- In this talk we will describe a framework which allows us to rewrite context-free families of string diagrams.

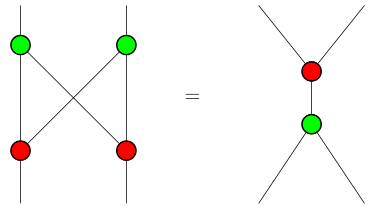
String Diagrams and String Graphs



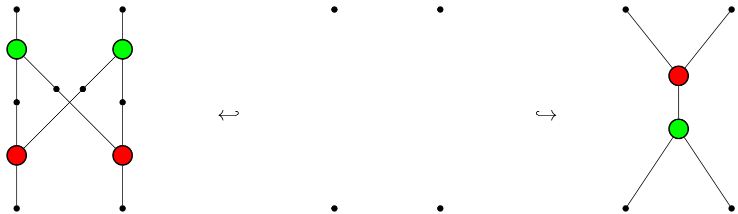
- Discrete representation exists in the form of *String Graphs*.
- String graphs are typed (directed) graphs, such that:
  - Every vertex is either a *node-vertex* or a *wire-vertex*.
  - No edges between node-vertices.
  - In-degree of every wire-vertex is at most one.
  - Out-degree of every wire-vertex is at most one.

String Diagram Equations

In the context of quantum computing and the ZX-calculus, the *Bialgebra rule* is given by the *string diagram equation*:



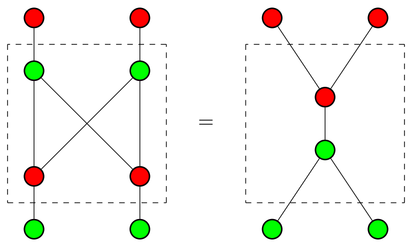
In terms of string graphs, this corresponds to a DPO rewrite rule:



where the interface and its embeddings are determined by the inputs and outputs of the equation.

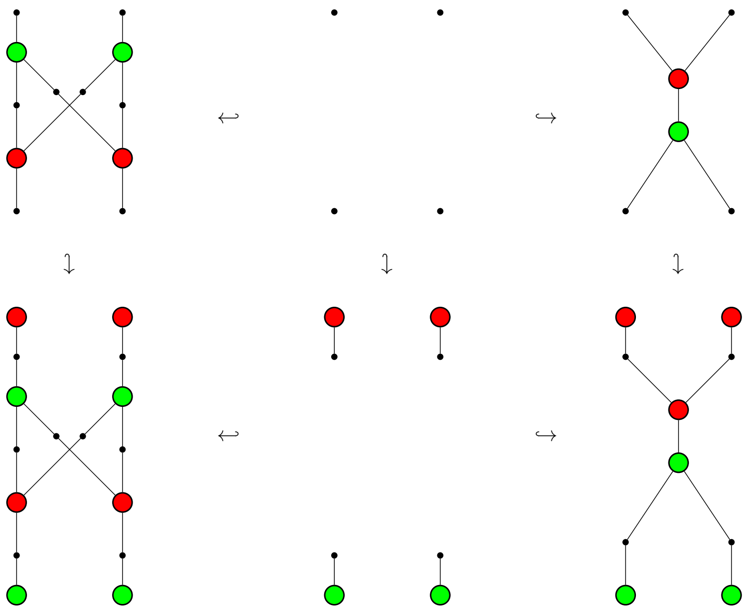
Equational Reasoning with String Diagrams

String diagrams may be used for equational reasoning:



(Bialgebra)

In terms of string graphs, this corresponds to a DPO rewrite:

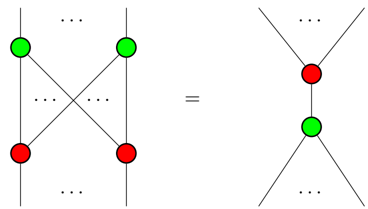


Motivation

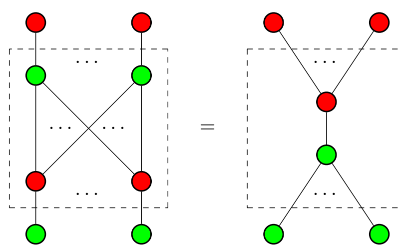
- In the ZX-calculus, the standard axiomatisation is expressed in terms of *families* of diagrams.
- In quantum computing, algorithms and protocols are often described as uniform *families* of diagrams.
- How can we represent *families* of string diagrams and how can we rewrite them?

Example

The *generalised bialgebra rule* is an *equational schema* in the ZX-calculus:



which may also be used for rewriting families of diagrams:



Approach

The main ideas are:

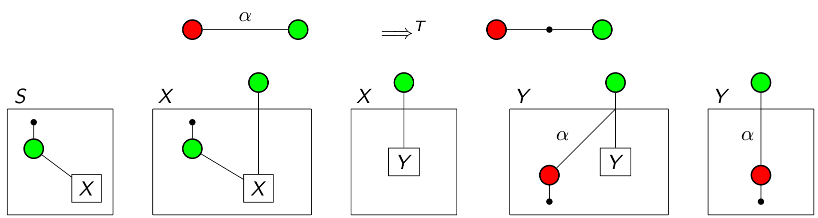
- Context-free graph grammars represent families of graphs (diagrams)
- Grammar DPO rewrite rules represent equational schemas
- Grammar DPO rewriting represents equational reasoning on families of graphs (diagrams)
- Grammar DPO rewriting is admissible (or correct) w.r.t. concrete instantiations

Context-free graph grammars

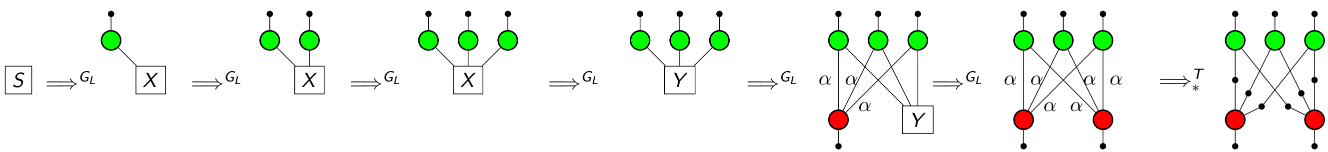
We will be using (slightly modified) context-free graph grammars, subject to some (omitted) conditions, to represent families of string graphs.

Example

The following grammar generates the LHS of the generalised bialgebra rule (represented as string graphs):



A derivation in the grammar of the string graph with three green vertices and two red vertices:



Theorem

These grammars generate only languages of string graphs and the membership problem is decidable.

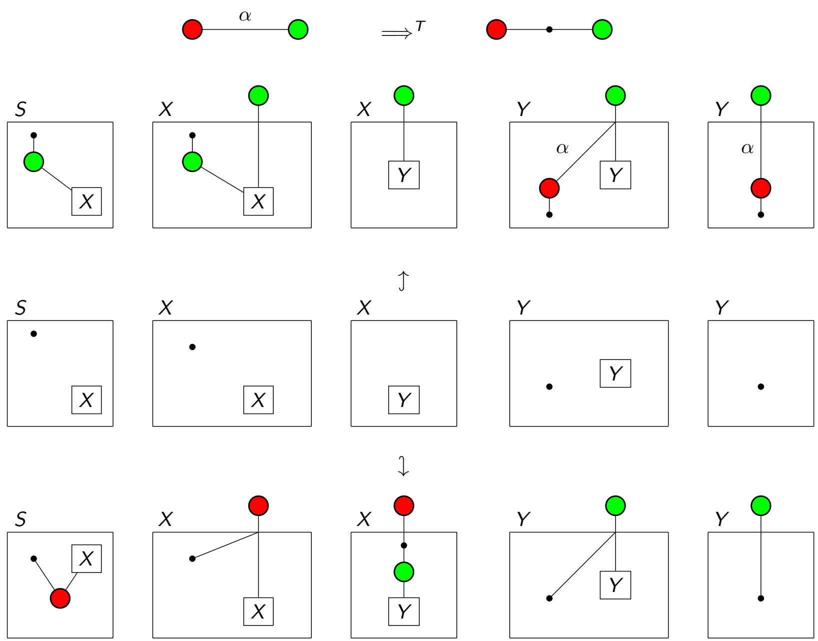
Adhesivity of graph grammars

- The category of context-free grammars **SGram** is a partially adhesive category.
- Suitable for performing DPO rewriting.
- Languages induced by context-free grammars are defined set-theoretically, not algebraically.
- Restrictions on rewrite rules and matchings necessary if we wish rewriting of grammars to make sense w.r.t language generation.

Representing Equational Schemas

Main idea: an equational schema is represented by a *grammar rewrite rule* which is a DPO rewrite rule in **SGram**, where productions (and their corresponding nonterminal vertices) are in bijective correspondance.

Example

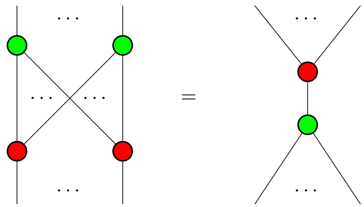


Equational Schemas and Instantiations

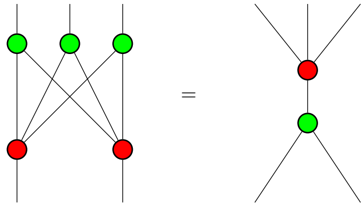
An equational schema can always be instantiated to produce specific string diagram equations.

Example

The generalised bialgebra schema (denoted $K_{m,n} = S_{m,n}$):



is parameterised by two natural numbers m and n. Each pair of natural numbers determines an equality of string diagrams. For example $K_{3,2} = S_{3,2}$ is given by:

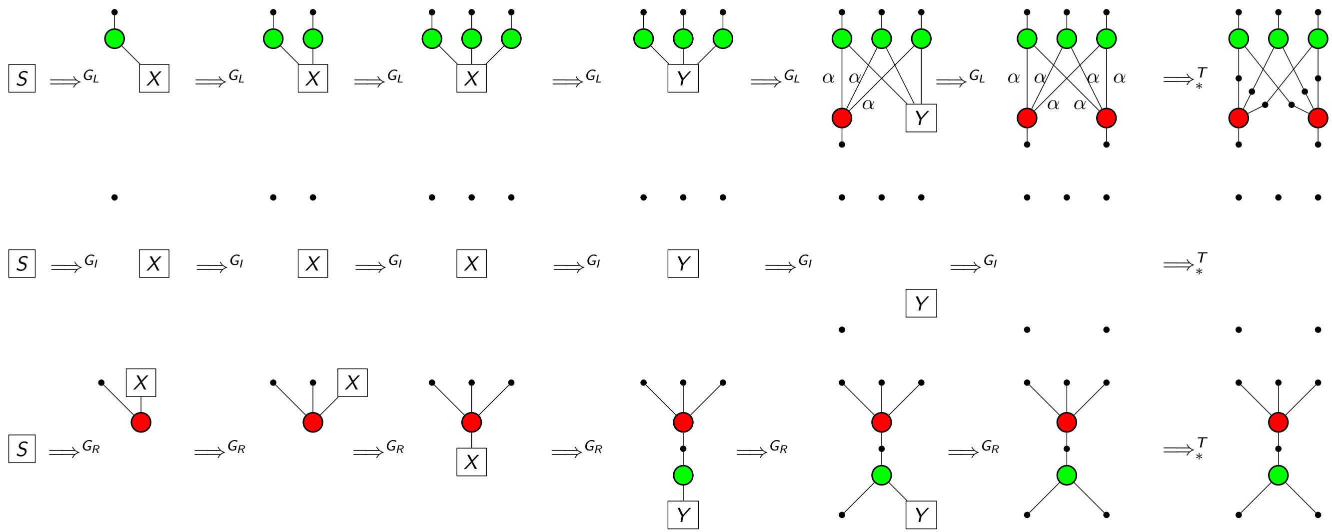


Representing Instantiations

An instantiation of a grammar rewrite rule is given by a triple of parallel derivations, together with their induced embeddings.

Example

The instantiation of $K_{m,n} = S_{m,n}$ to $K_{3,2} = S_{3,2}$ is represented by the parallel derivation:



together with the obvious induced embeddings (vertical from the middle sentential forms).

Theorem

Every grammar rewrite rule instantiation is a DPO rewrite rule on string graphs.

Rewriting in **SGram**

So far:

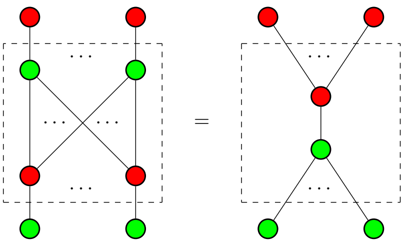
- String diagram $\mapsto$ string graph.
- String diagram equation $\mapsto$ DPO rewrite rule in **SGraph**.
- String diagram equational reasoning $\mapsto$ DPO rewriting in **SGraph**.
- Family of string diagrams $\mapsto$ Graph grammar of string graphs.
- Equational schema of string diagrams $\mapsto$ DPO rewrite rule in **SGram**.

Next:

- Equational reasoning with families of string diagrams $\mapsto$ DPO rewriting in **SGram**.

Example

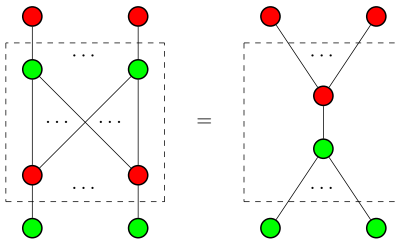
The equational schema:



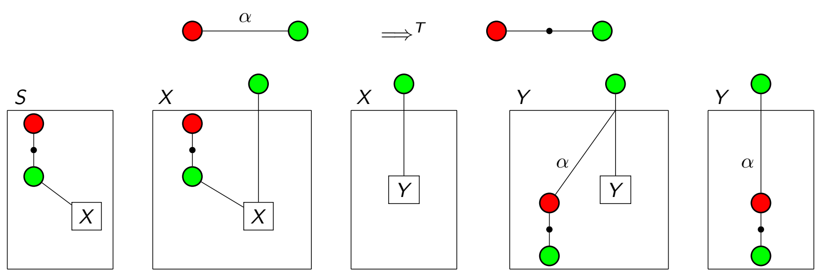
may be obtained by applying the schema $K_{m,n} = S_{m,n}$ to the LHS above.

In general, rewriting of *families* of string diagrams is represented by a DPO rewrite rule in **SGram** subject to some *strong* matching conditions.

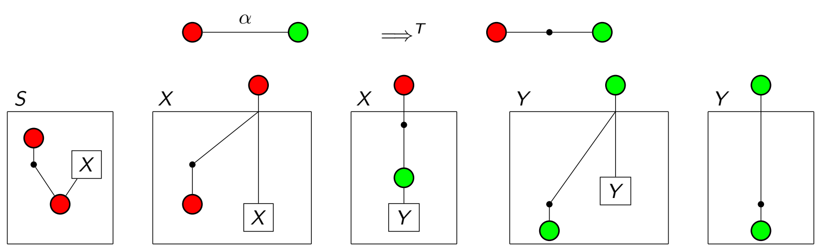
Rewriting in **SGram**



We saw how to reprsent the subschema in the dashed boxes via a DPO rewrite rule in **SGram**. The LHS of the whole schema is represented by the grammar:



Performing the DPO rewrite in **SGram** results in:



which correctly represents the RHS.

Admissibility

- Grammar rewriting as defined is admissible in the sense that it respects the concrete semantics of the grammars (and the equational schemas).
- More formally:
- If a grammar G rewrites into a grammar G' via a grammar rewrite rule B, then:
  - Every concrete instantiation of B is a DPO rewrite rule on string graphs.
  - The language of B, denoted $L(B)$ is the set of all such DPO rewrite rules.
  - For any concrete instantion H of G, a parallel concrete derivation H' exists for G'.
  - Finally, the graph H' can be obtained from the graph H by applying some number of DPO rewrite rules on graphs from $L(B)$ in any order.

Theorem

*Every DPO rewrite in **SGram** subject to our strong matching conditions is admissible in the above sense.*

Conclusion and Future Work

- Basis for formalized equational reasoning for context-free families of string diagrams.
- Framework can handle equational schemas and it can apply them to equationally reason about families of string diagrams.
- Meta-theory mixes categorical (DPO rewriting) and algorithmic (Grammar derivations) rewriting and is rather complicated.
- Future work: consider representing string diagrams as *hypergraphs* and families of string diagrams as *hypergraph grammars*:
  - Lower expressive power.
  - Better categorical properties (e.g. adhesivity vs partial adhesivity).
  - Better structural properties (e.g. no "wire-homeomorphism").
  - Better complexity properties.
  - Grammar derivations can be understood algebraically.
  - Probably cleaner meta-theory.

Thank you for your attention!