

Graphical reasoning in symmetric monoidal categories

Lucas Dixon, University of Edinburgh

Joint work with: Ross Duncan and Aleks Kissinger, University of Oxford

5 Nov 2009



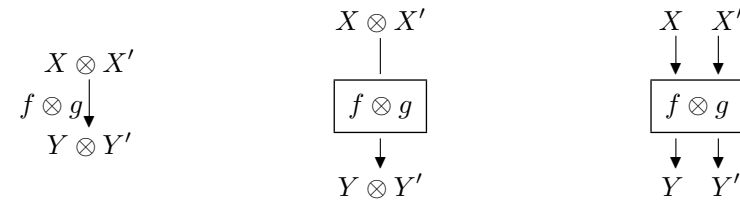
Outline

- Motivation: characterise processes (quantum computation)
- Symmetric Monoidal Categories and Graphs
- Example with boolean circuits
- Extended graphs, Matching and Plugging
- Inductive patterns of graphs with !-boxes

Symmetric Monoidal Categories (SMC)

- C is a monoidal category: it has associative and unital bifunctor \otimes :
 - \otimes operation on objects: $X \otimes Y$; and specific identity object I (\otimes is associative and has I as identity)
 - \otimes operation on morphisms: if $f : X \rightarrow Y$ and $g : X' \rightarrow Y'$ then $(f \otimes g) : (X \otimes X') \rightarrow (Y \otimes Y')$ (associative and has identity id)
- *Braided*: has 'braiding' isomorphisms: $\sigma_{X,Y} : X \otimes Y \rightarrow Y \otimes X$.
- *Symmetric*: $\sigma_{X,Y} \circ \sigma_{Y,X} = id$.

Typed Graphs = SMC



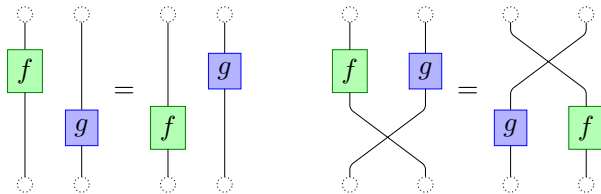
Category Theory \Rightarrow swap edges and vertices \Rightarrow tensor is spacial

- already the generic way to draw processes, e.g. circuits: **Vertices are operations** and **Edges are objects**,
- Coherence conditions provide correctness for graphical notation: equality for graph = equality for SMC

Graphical Representation

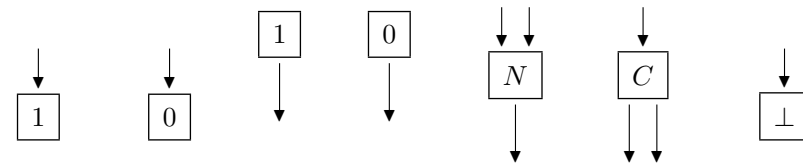


- We can express the bifunctionality of \otimes and the symmetric braiding of σ as:



Example: Boolean Circuits

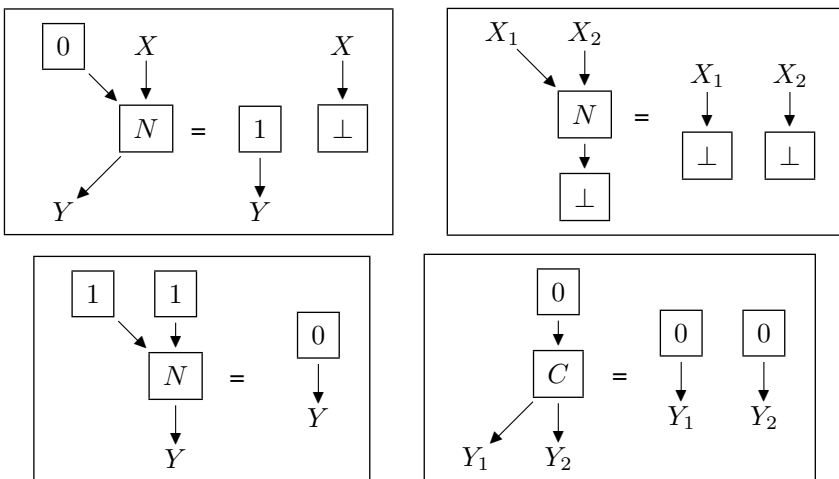
Values: $B = \{0, 1\}$; Operations: $N : B \otimes B \rightarrow B$, $C : B \rightarrow B \otimes B$, $\perp : B \rightarrow 1$



Out 1 Out 0 In 1 In 0 Nand Copy Ignore

Symmetric monoidal categories composition of diagrams; need additional equational structure to describe equivalences between circuits...

Example: Boolean Circuit Graphical Equations



Graphical Reasoning:

- Goal: *to develop suitable formalism for reasoning about equational structure in symmetric monoidal categories.*
- Based on *SMC as graphs*.
- Incident edges to a vertex define its *type*
- 'subject reduction': *rewriting preserves types*
- rewriting and plugging commute* (plugging doesn't break matching)
- reason with common *inductive structures*

Graphs

- Directed graph: $E \begin{matrix} \xrightarrow{s} \\ \xrightarrow{t} \end{matrix} V$

Any number of edges are allowed between vertices (not a binary relation)

- $G = (G_E, G_V, s, t); E = G_E; V = G_V; \text{in}(v) := t^{-1}(v); \text{out}(v) := s^{-1}(v)$
- graph morphism (graphs: G, H) $f_E : E_G \rightarrow E_H$ and $f_V : V_G \rightarrow V_H$ where:

$$s_H \circ f_E = f_V \circ s_G$$

$$t_H \circ f_E = f_V \circ t_G$$

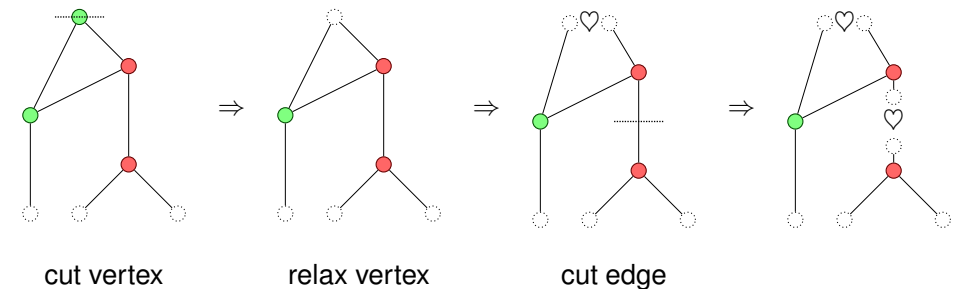
Extended Open Graphs

- Extended open graph: (G, X)
 $X \subseteq V$ (exterior); $\text{Int } G = V \setminus X$ (interior)
- Exterior vertices define an interface (hierarchical)
a subgraph has the same character as a vertex
- Morphism of open graphs: $f : (G, G_X) \rightarrow (H, H_X)$ (only map to H_X from G_X)
 $\forall v \in V_G. f_V(v) \in \partial H \Rightarrow v \in \partial G$
- Strict Morphism: $f : (G, G_X) \rightarrow (H, H_X)$ (no extra interior edges)
 $\forall e \in E_H. s_H(e) \in f_V(\text{Int } G) \vee t_H(e) \in f_V(\text{Int } G) \Rightarrow \exists e' \in E_G. f_E(e') = e.$
- There is also a topological interpretation: morphisms as continuous maps

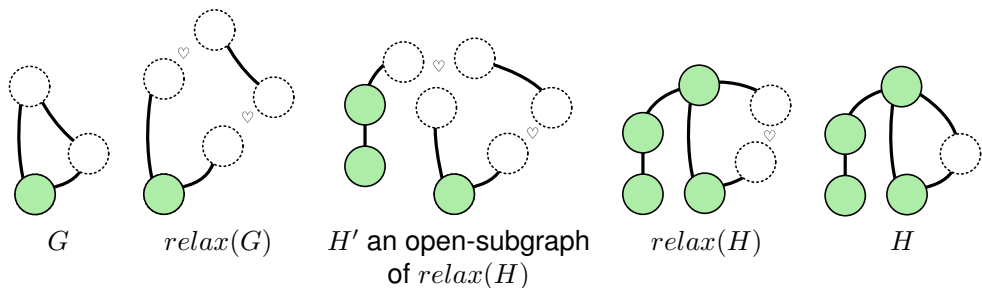
Matching for Extended Graphs

- Relaxed subgraph: cut and relaxed
 - cut an edge: introduce two-clique new exterior vertices
 - cut a vertex: throw away data, make exterior
 - relax a vertex: makes incidence 1 'loving' vertex-cliques of exterior vertex
 - love: relation between cliques of exterior vertices
- $G \leq H$ (G matches H) = $\exists f$ which is an open graph morphism from a relaxed G to a relaxed subgraph of H , such that (it is an exact embedding):
 - f is a strict love morphism; (locally preserves type)
 - f_E and f_V are injective; (mapped 1-1 in subgraph)
 - $\forall v \in V_G. f_V(v) \in \partial H \Leftrightarrow v \in \partial G$ (exact X map)

Matching Example 1

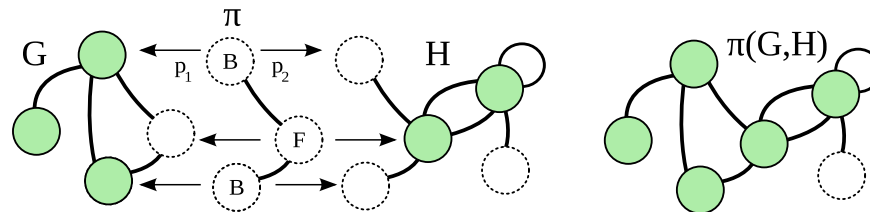


Matching Example 2



- Efficient algorithm by graph traversal:
 - relaxation built in
 - cuts implicit by left-over graph.

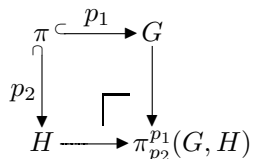
Composing Graphs: a picture



Plugging of G and H via the two-sided e-graph π with embeddings p_1 and p_2 .

Composing Graphs: Plugging

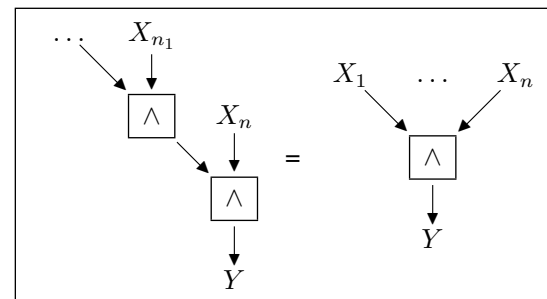
- $((\pi, \pi_X), (F, B))$ a graph, π , with $\pi_V = \pi_X$ and partition of π_V into F and B
- Pair of embeddings: $p_1 : (\pi, \pi_X) \rightarrow (G, G_X)$ and $p_2 : (\pi, \pi_X) \rightarrow (H, H_X)$ such that $p_1(F) \subseteq X$ and $p_2(B) \subseteq Y$
- *plugging*, $\pi_{p_2}^{p_1}(G, H)$, defined by pushout:



(minimal graph matched by both G and H where the two π 's are identified)

- Properties: $\pi(G, H) \cong \pi(H, G)$; $G \leq_e \pi(G, H)$ and $H \leq_e \pi(G, H)$; $K \leq_e G$ implies $K \leq_e \pi(G, H)$;

Representing Inductive Families of Graphs



- Want a higher level language to capture such repeated structure; allow rewriting etc

!-Box Graphs

!-Box Graphs = (G, B) where B is a disjoint set of subsets of G_V
(draw a box around elements of each member of B)

!-Box Matching : G matches H : ($H \in G$ closed under:

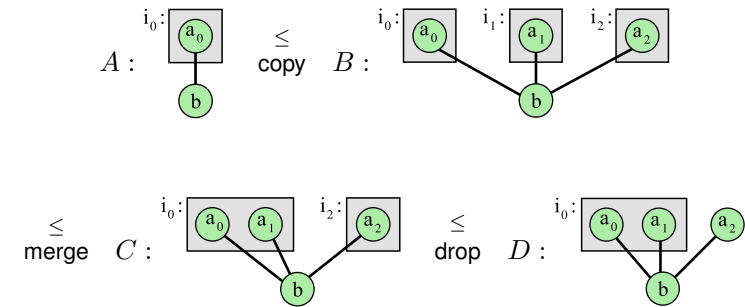
copy : copy subgraph including incident edges some number of times

drop : removes the !-box, keep the contents.

merge : combines two !-boxes: $\{B_1, B_2, \dots\} \rightarrow \{(B_1 \cup B_2), \dots\}$.

Semantics: $\llbracket G \rrbracket_!$, subset of matches that have no !-boxes.

!-Box Graphs: Example



Example showing how A matches D

Conclusions

- Symmetric monoidal categories have a natural graphical presentation
- Many processes form SMCs with extra equational structure
- High level language for processes motivates !-boxes to capture inductive structure (ellipsis notation)
- Initial goal was to reason about quantum information; also has applications to traditional circuits
- Developed a formalism for equational reasoning over graph-based representations of symmetric monoidal categories
- Implementation: <http://dream.inf.ed.ac.uk/projects/quantomatic>