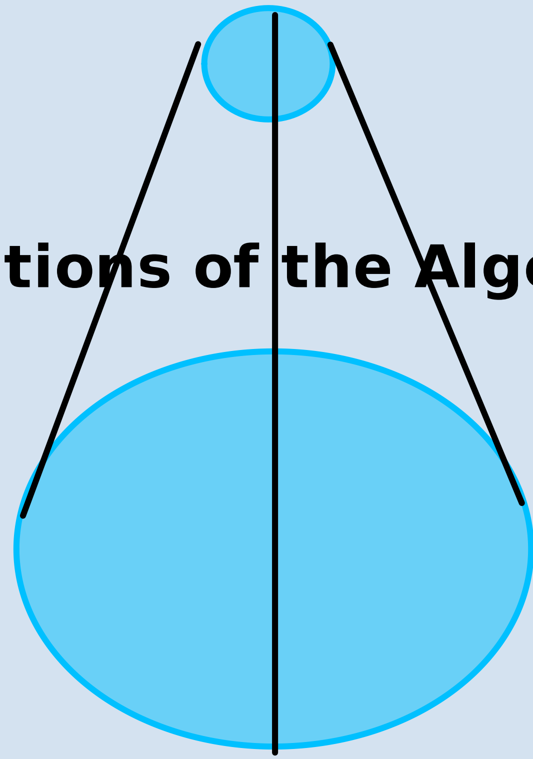


Composing Solutions of the Algebraic Path Problem and More

Jade Master
SYCO10 17/12/22

Joint work with
Ben Bumpus and
Zoltan Kocsis

Overview: I started out researching structured cospans.

$$C \xrightarrow{F} D$$

cat of networks

a computation

cat of network semantics

Which lift to double functors between double categories of structured cospans. At some point I became obsessed with the idea of a "compositional formula".
This is my story...

Table of Contents

Section 1: Enriched Graphs and Their Decompositions

Defn: An R-graph is a function $G : X \times X \longrightarrow R$ and a morphism of R-graphs is a diagram

$$\begin{tikzcd}
X \times X \arrow[rd] \arrow[dd, "f \times f"'] & \\
\arrow[r, phantom, "g \Downarrow"] & R \\
Y \times Y \arrow[ru] &
\end{tikzcd}$$

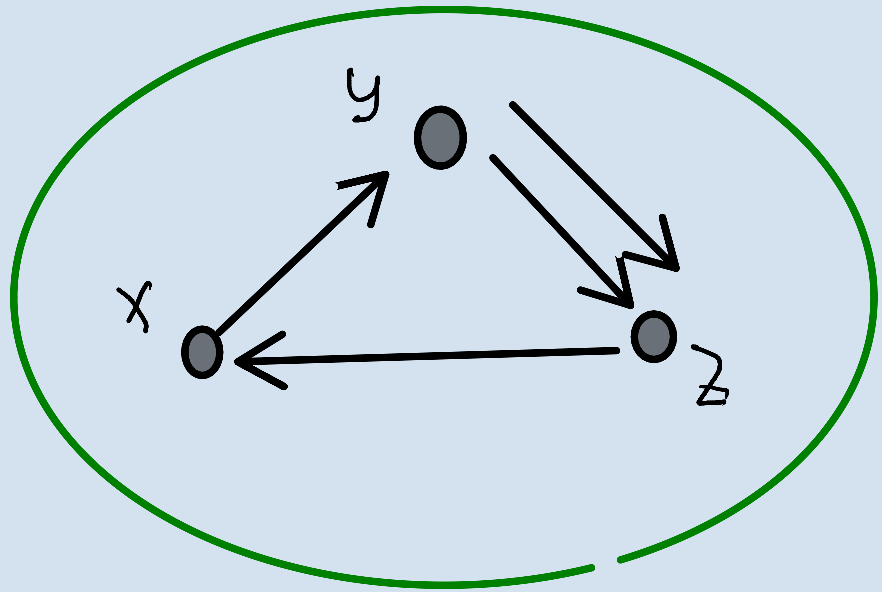
R is a monoidal closed category $(R, +, \otimes)$ e.g. $(Set, +, \times)$ $([0, \infty], \min, +)$ etc.

This defines $RGrph$

Denote Set-Grph by Grph

Defn: Let $\mathrm{Mat}(R)$ be the category where

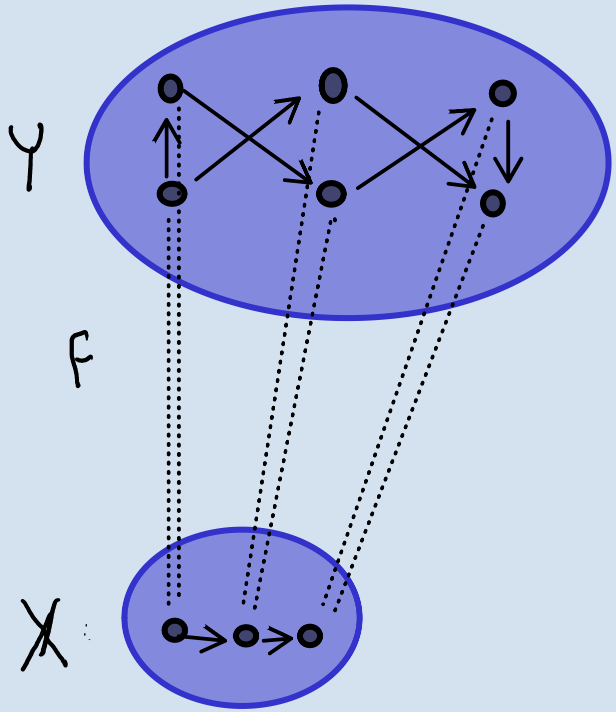
- objects are sets
- a matrix $M : X \times Y \to R$ is a morphism from X to Y
- Composition is matrix multiplication



$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix}$$

What is a Graph Decomposition?

Defn: A Set-graph decomposition is just a morphism of set-graphs $Y \xrightarrow{F} X$.
such that each vertex $x \in X$ has a unique self-edge $r_x : x \to x$



- X is the shape graph
- Y is the total graph
- $F^{-1}(r_x)$ is the local graph at x

Shift Your Perspective

There is an equivalence

$$\mathrm{Grph}^{\to} \xrightarrow{\sim} \mathrm{Grph}/\mathrm{UMat}(\mathrm{Set})$$

i.e. for a graph morphism $f: Y \to X$, its fibers collect into a graph morphism

$$f^{-1}: X \longrightarrow \mathrm{UMat}(\mathrm{Set})$$

mapping vertices to their fibers and an edge

$$e: x \to y \quad \longmapsto \quad f^{-1}(e): f^{-1}(x) \times f^{-1}(y) \longrightarrow \mathrm{Set}$$

$$(a,b) \longmapsto Y(a,b)$$

The weak inverse $\mathrm{Grph}/\mathrm{UMat}(\mathrm{Set}) \xrightarrow{\sim} \mathrm{Grph}^{\rightarrow}$ is a Grothendieck construction sending

$A : G \longrightarrow \mathrm{UMat}(\mathrm{Set}) \longmapsto \int A \xrightarrow{\pi} G$ where

$\int A$ is the graph where

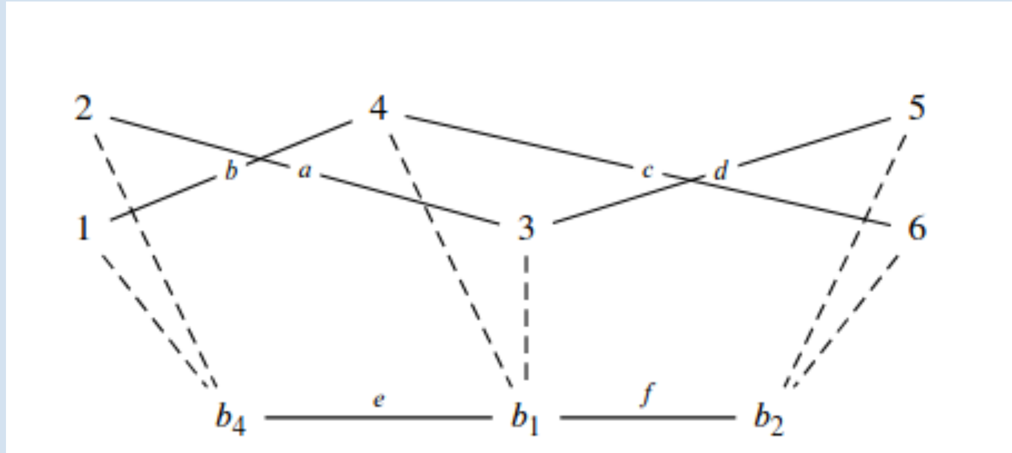
Global coordinate

Local coordinate

- vertices are pairs $(x \in V(G), a \in A(x))$
- edges are pairs $(e \in G(x,y), s \in A(e)(a,b))$ from (x,a) to (y,b)

The dependent and the fibrational perspectives are equivalent

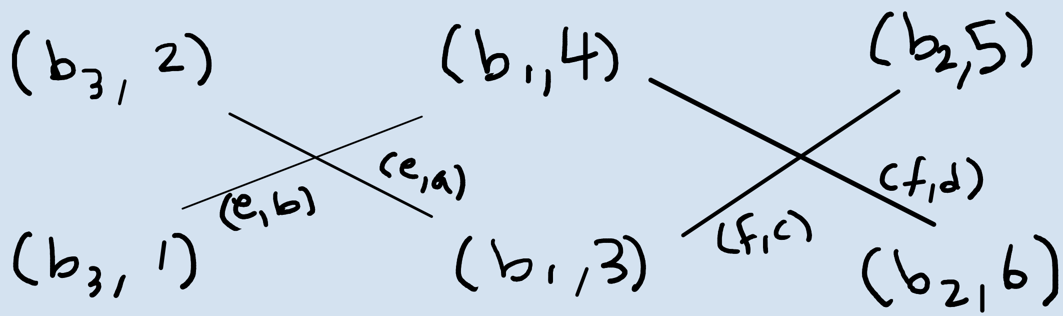
$$\begin{tikzcd} H \arrow[d, "A"] \\ G \end{tikzcd}$$



$$A^{-1} : G \longrightarrow UMat(Set)$$

$$\{2,1\} \xrightarrow{\begin{bmatrix} \phi & \{a\} \\ \{b\} & \phi \end{bmatrix}} \{4,3\} \xrightarrow{\begin{bmatrix} \phi & \{c\} \\ \{d\} & 0 \end{bmatrix}} \{5,6\}$$

$$\int_{Grph} A^{-1}$$



A decomposition $\mathrm{D}\colon \mathrm{G} \to \mathsf{UMat}(\mathrm{R})$ may be extended to sets of edges $\{\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_\mathrm{n} \in \mathrm{G}(\mathrm{x}, \mathrm{y})\}$ by defining

$$D(\{f_1, f_2, \ldots, f_n\})\colon D(x) \times D(y) \to R \text{ by } (a, b) \mapsto \sum_{j=1}^{n} D(f_j)(a, b.)$$

D(G(x,x)) is the local graph at x

General Graph Decompositions are Defined Dependently

Defn: An R-graph decomposition is a morphism of R-graphs $A : G \longrightarrow U\,\mathrm{Mat}(R)$

But they also have a fibrational perspective.

. *For a 2-rig* R*, there is an adjunction*

$$\begin{tikzcd} \mathsf{SetGraph} \arrow[rr, bend left, "(-)_{\mathrm{R}}"] \arrow[rr, phantom, "\perp"] & & \mathsf{RGraph} \arrow[ll, bend left, "(-)_0"] \end{tikzcd}$$

$$G : X \times X \to \mathsf{Set} \mapsto G_R \text{ with } G_R(x, y) = \sum_{f \in G(x,y)} 1_R$$

$$H : Y \times Y \to R \mapsto H_0 \text{ with } H_0(x, y) = R(1_R, H(x, y))$$

There is an equivalence of categories

$$\int_{Grph} : Grph/UMat(R) \xrightarrow{\sim} RGrph^{\to}$$

$$D : G \longrightarrow UMat(R) \longmapsto \int_{Grph} D \longrightarrow G_R$$

where G_R is the free R-graph $G_R(x,y) = \sum_{f \in G(x,y)} 1$

$$\int_{Grph} D((x,a),(y,b)) = \sum_{e \in G(x,y)} D(e)(a,b)$$

Section 2: Free Enriched and Dependent Categories

There is an adjunction

$$\begin{tikzcd} R\mathrm{Grph} \arrow[rr, bend left, "F_R"] \arrow[rr, phantom, "\perp"] & & R\mathrm{Cat} \arrow[ll, bend left, "U_R"] \end{tikzcd}$$

sending an R-enriched graph to its free R-category $F_R(M) = \sum_{n \geq 0} M^n$ and with U_R forgetful. When $R = (\mathrm{Set}, +, \times)$ this is the usual free category adjunction

$$\begin{tikzcd} \mathrm{Grph} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathrm{Cat} \arrow[l, bend left, "U"] \end{tikzcd}$$

Finding $F_R(M)$ is called the <u>algebraic path problem</u> when R is an arbitrary semiring. These are interesting algorithmic problems. For example:

poset	join	multiplication	solution of path problem
$([0,\infty], \geq)$	inf	$+$	shortest paths in a weighted graph
$([0,\infty], \leq)$	sup	inf	maximum capacity in the tunnel problem
$([0,1], \leq)$	sup	$\times$	most likely paths in a Markov process
$\{T, F\}$	OR	AND	transitive closure of a directed graph
$(\mathcal{P}(\Sigma^*), \subseteq)$	$\bigcup$	concatenation	decidable language of a NFA

How do we extend F_R to graph decompositions?

There is an adjunction

$$\begin{tikzcd} \mathsf{Grph}/U\mathsf{Mat}(R) \arrow[r, bend left, "F^i"] \arrow[r, phantom, "\perp"] & \mathsf{Cat}/\mathsf{Mat}(R) \arrow[l, bend left, "U^i"] \end{tikzcd}$$

$$D : G \to U\mathsf{Mat}(R) \quad \mapsto \quad F^i D : FG \to \mathsf{Mat}(R)$$

defined by

$$F^i D(e_1, e_2, \ldots, e_n) = D(e_1) \circ D(e_2) \cdots D(e_n)$$

for a sequence of edges $\vec{e} \in \mathrm{Mor}\, FG$

$\mathrm{F^iD}$ may be extended to sets of morphisms $\{\mathrm{f_1, f_2, \ldots, f_n \in FG(x, y)}\}$ by defining

$$F^iD(\{f_1, f_2, \ldots, f_n\})\colon D(x) \times D(y) \to R \text{ by } (a, b) \mapsto \sum_{j=1}^{n} F^i(f_j)(a, b)$$

Then

$$F^iD(G(x, x)^*) \cong \sum_{n \geq 0} F^iD(G(x, x)^n) \cong \sum_{n \geq 0} D(G(x, x))^n \cong F(D(G(x, x))).$$

is the local solution at x

There is an adjunction

$$\begin{tikzcd} \mathcal{R}\mathrm{Grph}^{\to} \arrow[r, bend left, "F^{\to}"] \arrow[r, phantom, "\perp"] & \mathcal{R}\mathrm{Cat}^{\to} \arrow[l, bend left, "U^{\to}"] \end{tikzcd}$$

given by pointwise application ie.

$$\begin{tikzcd} H \arrow[d, "F"'] \\ G \end{tikzcd} \longmapsto \begin{tikzcd} FH \arrow[d, "FF"'] \\ FG \end{tikzcd} \qquad \begin{tikzcd} C \arrow[d, "A"'] \\ D \end{tikzcd} \longmapsto \begin{tikzcd} UC \arrow[d, "UA"'] \\ UD \end{tikzcd}$$

Section 3: The Compositional Theorem

General Compositional Formula for the APP

1. *For an* R*-graph decomposition* D: G $\to$ UMat(R) *there is an isomorphism*

$$F\int_{Gr} D((x,a),(y,b)) \cong \sum_{(\bar{x},\bar{e})\in FG^{\bullet}(x,y)} \left(\prod_{i=1}^{n} FD(G(x_i,x_i))D(e_i)\right)FD(G(x_n,x_n))(a,b)$$

$FG^0(x,y) = \{ \text{paths } x = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} x_n = y \text{ of } G^0 \}$

G^0 is G with all self-loops removed

In practice the sum over $P_G(x,y)$ may be truncated at a path length n equal to the size of the largest local graph.

For example, when $R = ([0,\infty], \min, +)$ any term larger than this n will contain a loop and can be made shorter.

Section 3: Computational Results on an Example

An Algorithm

$$F\int_{Grph} D\big((x,a)(y,b)\big) \simeq \sum_{\substack{(\bar{x},\bar{e}) \in \\ P_G(x,y)}} \left(\prod_{i=1}^{n-1} F^i D\left(G(x_i,x_i)^*\right) \cdot D(e_i) \right) F^i D\left(G(x_n,x_n)^*\right)(a,b)$$

These terms are local solutions

these terms are connecting edges

To find the global solution

$$F\int_{Grph} D\big((x,a),(y,b)\big)$$

1. Find the set $P_G(x,y)$, this may be precompiled and truncated
2. Find the local solutions $F^i D\left(G(x_i,x_i)^*\right)$ for each connected vertex
3. Plug your results into this Formula

Example

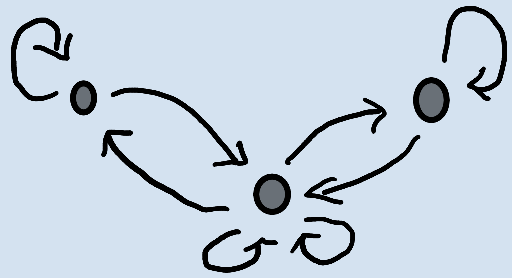
If $G = \begin{tikzcd} x \arrow[loop above, "e"] \end{tikzcd}$ then $F\int_{G} D \simeq F(D(e))$

If $G = \begin{tikzcd} x \arrow[loop left, "c"] \arrow[r, bend left, "a"] & y \arrow[l, bend left, "b"] \arrow[loop right, "d"] \end{tikzcd}$ then $FG^{\bullet}(x,y) = a(ba)^{*}$

and

$$F\int_{G} D \simeq \sum_{n \geq 0} FD(c) \cdot D(a) \big(FD(d)\, D(b) \big)^{n}$$

Example

$G =$ 

Let the shape graph be

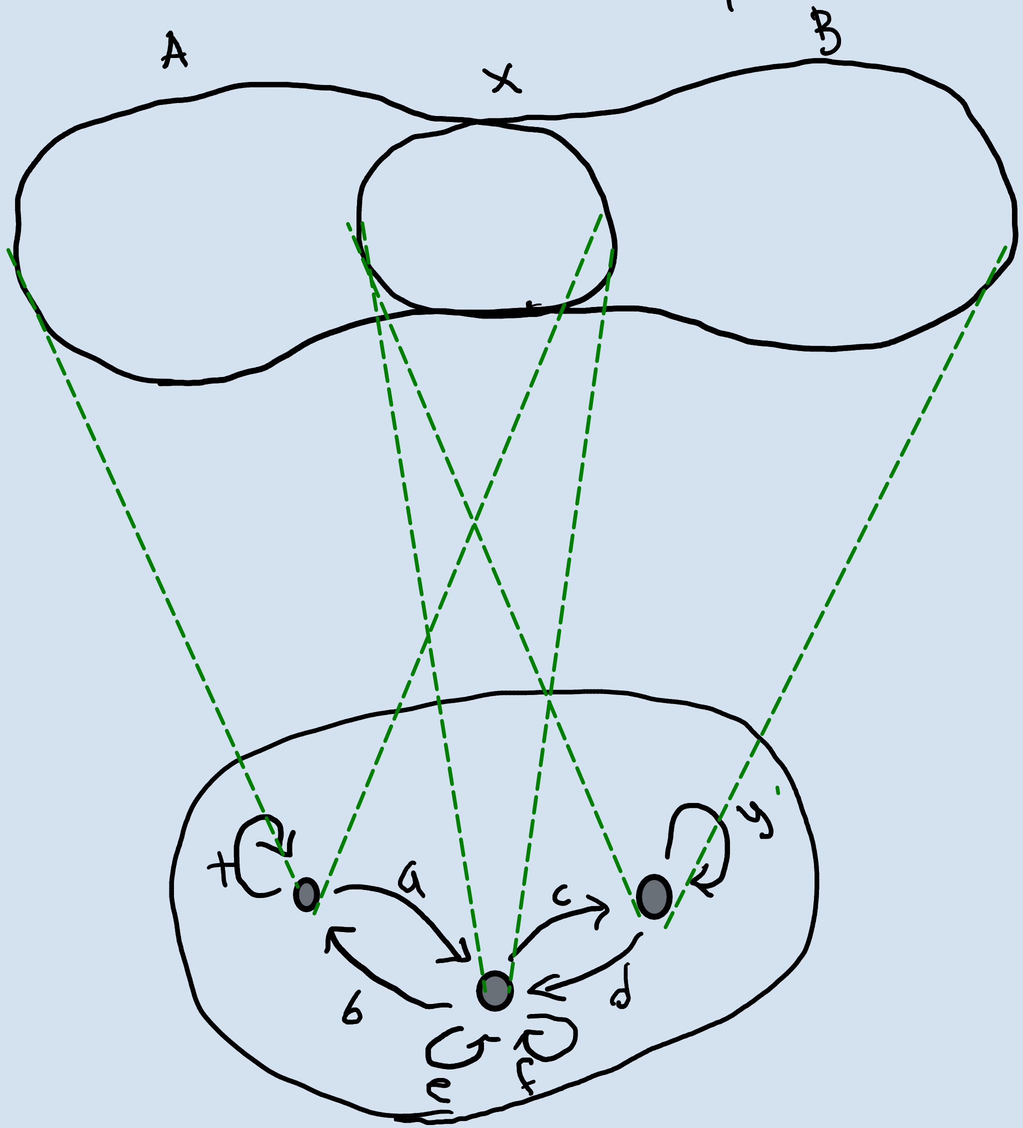
This decomposition comes from the pushout

$$\begin{tikzcd} & A +_{LX} B & \\ A \arrow[ur] & & B \arrow[ul] \\ & LX \arrow[ul] \arrow[ur] & \end{tikzcd}$$

over a discrete graph LX.

This pushout forms the composition for structured cospan categories

There are three components



The paths are found by taking powers of

$$\begin{bmatrix} \emptyset & \{a\} & \emptyset \\ \{b\} & \emptyset & \{c\} \\ \emptyset & \{d\} & \emptyset \end{bmatrix}$$

i.e. the free category on $G^0 =$
$$\begin{tikzcd} \bullet \arrow[r, bend left, "a"] & \bullet \arrow[l, bend left, "b"] \arrow[r, bend left, "c"] & \bullet \arrow[l, bend left, "d"] \end{tikzcd}$$

The local solutions $F^i D(G(x,x)^*)$ may be found. and for each path $x_1 \xrightarrow{e_1} x_2 \rightarrow \cdots \xrightarrow{e_{n-1}} x_n$ in $FG^0(x,y)$ we obtain a matrix

$$\left(\prod_{i=1}^{n-1} F^i D(G(x_i,x_i)^*) D(e_i)\right) F^i D(G(x_n,x_n)^*)$$

Adding these matrices at a point (a,b) gives the solution of the APP from (x,a) to (y,b)

I coded this example up

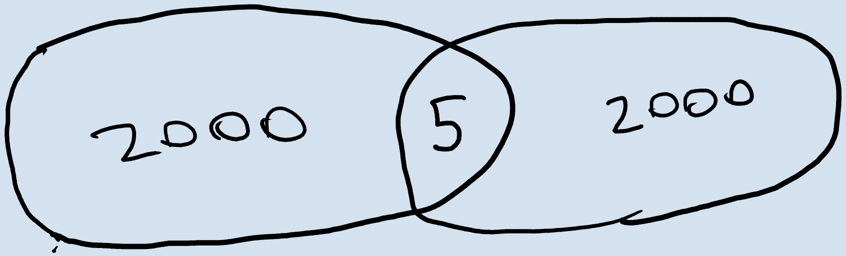
https://github.com/Jademaster/pathcomposer

```python
#change this to arbitrary structured decompositions
class Compositionproblem():

    def __init__(self,g1,g2,intersection,matsemi):
        self.lgraph=g1
        self.rgraph=g2
        self.intersection=intersection
        self.symbols=None
        self.matsemi=matsemi
        self.values=None
        self.joined=None
        self.pushforwards=None
        self.lengths=None

    def precompilesymbols(self):
        maxboundary=len(self.intersection)
        self.symbols=symbols.paths(maxboundary)[-1]

    def precompilematrices(self):
        Fxmat=nx.floyd_warshall_numpy(nx.from_numpy_matrix(self.lgraph.getmatrix()))
        Fymat=nx.floyd_warshall_numpy(nx.from_numpy_matrix(self.rgraph.getmatrix()))
        self.lgraph=graphfrommatrix(Fxmat,self.lgraph.verts)
        self.rgraph=graphfrommatrix(Fymat,self.rgraph.verts)

```

```
def randomcompproblem(graphsize,boundarysize,matsemi):
    x=np.random.randint(1,10,(graphsize,graphsize))
    y=np.random.randint(1,10,(graphsize,graphsize))
    gx=graphfrommatrix(x,[i for i in range(graphsize)])
    gy=graphfrommatrix(y,[i+graphsize for i in range(graphsize)])
    i=dict(zip([i for i in range(graphsize-boundarysize,graphsize)],[j for j in range(graphsize,graphsize+boundarysize)]))
    return Compositionproblem(gx,gy,i,matsemi)
```

```
def shortestpath(self,s,t):
    lengthg,lengthk,lengthh,total=self.lengths
    gstar,hstar = self.pushforwards
    values=self.values
    #find compositional shortest path
    #find s and t
    i=list(self.joined.outedges).index(s)
    j=list(self.joined.outedges).index(t)
    valueg=np.array([gstar[i,j]])
    valueh=np.array([hstar[i,j]])
    #add start and end vectors to value list
    newvalues={
    #start symbols
    symbols.GGs:gstar[i,0:lengthg][None,:],
    symbols.GKs:gstar[i,lengthg:lengthg+lengthk][None,:],
    symbols.KGs:gstar[i,0:lengthg][None,:],
    symbols.gKKs:gstar[i,lengthg:lengthg+lengthk][None,:],
    symbols.hKKs:hstar[i,lengthg:lengthg+lengthk][None,:],
    symbols.KHs:hstar[i,total-lengthh:total][None,:],
    symbols.HKs:hstar[i,lengthg:lengthg+lengthk][None,:],
    symbols.HHs:hstar[i,total-lengthh:total][None,:],
    #final symbols
    symbols.GGt:gstar[0:lengthg,j][:,None],
    symbols.GKt:gstar[0:lengthg,j][:,None],
    symbols.KGt:gstar[lengthg:lengthg+lengthk,j][:,None],
    symbols.gKKt:gstar[lengthg:lengthg+lengthk,j][:,None],
```

random comp. problem



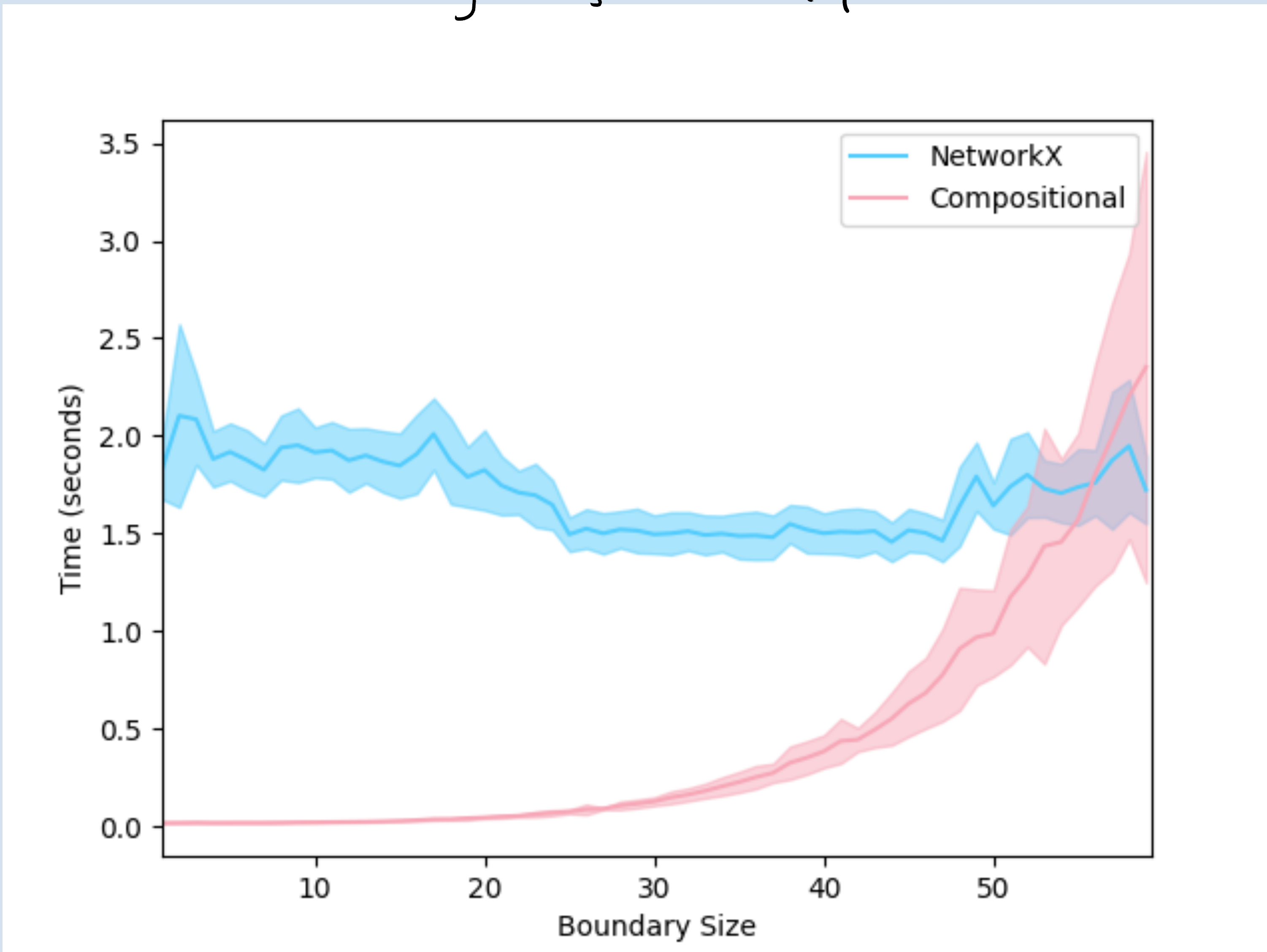
compositional:
$.1602 \pm .0169$

Djikstra:
39.7804 ± 3.3561

But precompilation took ~93 seconds

So we are trading space for time

The speed up is most dramatic when the boundary is small



Size 500 graphs, 50 runs at each boundary

Section 5: Proof of Theorem

To prove the compositional formula we need two lemmas. First we need another Grothendieck construction

$$\int_{Cat} : RCat/_{Mat(R)} \xrightarrow{\sim} RCat^{\to}$$

between indexed R-categories and R-functoss

"The R-enriched displayed category construction"

The equivalence

$$\int_{Cat} : RCat/_{Mat(R)} \xrightarrow{\sim} RCat^{\to}$$

is similar to the Grothendieck construction from before

$$\int_{Cat} (A : C \longrightarrow Mat\,R) \begin{cases} \text{Objects: } (x \in C,\ a \in A(x)) \\ \text{Morphisms:} \end{cases}$$

$$\int_{Cat} A((x,a),(y,b)) = \sum_{f \in C(x,y)} A(f)(a,b)$$

$$\mathsf{RGraph}^{\rightarrow} \cong \mathsf{Graph}/U\mathsf{Mat}(R)$$

$$f : H \to G \mapsto f^{-1} : G_0 \to U\mathsf{Mat}(R)$$

$$D\colon G \to U\mathsf{Mat}(R) \mapsto p\colon \int_{Gr} D \to G_R$$

$$\int_{Gr} D(x,y) = \sum_{e \in G(x,y)} D(e)$$

Lemma: (Linearity) there is a square

$$\begin{tikzcd} R\mathrm{Grph}^{\to} \arrow[d, "F^{\to}"'] & & \mathrm{Grph}/_{\mathrm{U}}\mathrm{Mat}(R) \arrow[ll, "\int_{\mathrm{Grph}}"'] \arrow[d, "F^{\mathrm{ind}}"] \\ R\mathrm{Cat}^{\to} \arrow[rru, phantom, "\cong"] & & \mathrm{Cat}/\mathrm{Mat}(R) \arrow[ll, "\int_{\mathrm{Cat}}"] \end{tikzcd}$$

commuting up to natural isomorphism

Explicitly, for a graph decomposition $D : G \to U\,\mathrm{Mat}(R)$ there is an isomorphism

$$F^{\to} \int_{\mathrm{Grph}} D\big((x,a),(y,b)\big) \simeq \int_{\mathrm{Cat}} F^{\mathrm{ind}}(D)\big((x,a),(y,b)\big)$$

$$\xrightarrow{\sim} \sum_{f \in FG(x,y)} F^{\mathrm{ind}} D(f)(a,b)$$

So with some eye squinting we have that "F preserves sums"

$$F^{\to} \sum_{e \in G(x,y)} D(e) \xrightarrow{\sim} \sum_{f \in FG(x,y)} F^{ind} D(f)$$

The left hand side is solution to the APP on the total graph.

This is already a compositional formula but there's no precompilation yet.

Lemma 2: There is an isomorphism

$$FG(x,y) \xrightarrow{\sim} \sum_{\substack{(\vec{x},\vec{e}) \in \\ FG^{0}(x,y)}} \left(\prod_{i=1}^{n-1} F^{i} D(G(x_i,x_i)^*) \, D(e_i) \right) F^{i} D(G(x_n,x_n)^*)$$

where FG^0 is the free category on G^0, the graph obtained by removing all self-edges from G

Idea: Consecutive morphisms on the same vertex may be factored out

Proof of Theorem

$$F\int_{Gr} D((x,a),(y,b)) \cong \int_{Cat} F^i D((x,a),(y,b))$$ **(Linearity)**

$$\cong \sum_{f\in FG(x,y)} F^i D(f)(a,b)$$ **(Definition)**

$$\cong \sum_{(\bar{x},\bar{e})\in FG^{\bullet}(x,y)} F^i D\Big(\Big(\prod_{i=1}^{n} G(x_i,x_i)^*\{e_{x_i}\}\Big)G(x_n,x_n)^*\Big)(a,b)$$

(Factorization)

$$\cong \sum_{(\bar{x},\bar{e})\in FG^{\bullet}(x,y)} \Big(\prod_{i=1}^{n} F^i D(G(x_i,x_i)^*)D(e_i)\Big)F^i D(G(x_n,x_n)^*)(a,b)$$

(Functoriality)

$$\cong \sum_{(\bar{x},\bar{e})\in FG^{\bullet}(x,y)} \Big(\prod_{i=1}^{n} FD(G(x_i,x_i))D(e_i)\Big)FD(G(x_n,x_n))(a,b)$$

Conclusion and Further Directions

Structured Decompositions

Definition 3.6 (Structured Decomposition)**.** Fix a base category K with pullbacks and a graph G. A *K-valued structured decomposition of shape G* is a dagger functor $D : \mathbf{F}_{\dagger}(G) \to \mathbf{Span}(\mathbf{K})$ from the free dagger category on G to $\mathbf{Span}(\mathbf{K})$. Given any vertex v in G, we call the object Dv in $\mathbf{K}$ the *bag of D indexed by v*. Given any edge $e = xy$ of G, we call the span $De := x \leftarrow a_e \rightarrow y$ the *adhesion indexed by e*.

We started from

a "spine" $F : K \to C$

Make the software better?

Complexity?

Other Algorithms?

Model Checking?

Thanks for listening!