

Thin spans and their modelling of rigid intersection types

Pierre Clairambault¹ Simon Forest²

¹LIS, CNRS

²LIS, Aix-Marseille Université

November 16, 2023

Semantic models

What is a semantic model?

program \mapsto *Some Mathematical Object*

Thin spans

What are thin spans?

- ▶ a semantic model which represents programs through **witnesses of computation**

$$A \vdash s : B \quad \rightsquigarrow \quad \begin{array}{ccc} & S & \\ \swarrow & & \searrow \\ !A & & B \end{array}$$

- ▶ a (bi)categorical abstraction of concurrent game semantics
- ▶ a proof-relevant refinement of the **relational model** of linear logic

Thin spans

What are thin spans?

- ▶ a semantic model which represents programs through **witnesses of computation**

$$A \vdash s : B \quad \rightsquigarrow \quad \begin{array}{ccc} & S & \\ \swarrow & & \searrow \\ !A & & B \end{array}$$

- ▶ a (bi)categorical abstraction of concurrent game semantics
- ▶ a proof-relevant refinement of the **relational model** of linear logic

Thin spans

What are thin spans?

- ▶ a semantic model which represents programs through **witnesses of computation**

$$A \vdash s : B \quad \rightsquigarrow \quad \begin{array}{ccc} & S & \\ \swarrow & & \searrow \\ !A & & B \end{array}$$

- ▶ a (bi)categorical abstraction of concurrent game semantics
- ▶ a proof-relevant refinement of the **relational model** of linear logic

Outline

From relations to spans

Interpreting programs

A rigid intersection type system

Outline

From relations to spans

Interpreting programs

A rigid intersection type system

CBN λ -calculus with effects

Consider your favorite λ -calculus and add to it an effectful operator, like a **non-deterministic operator** \bigvee

$$s, t, u, \dots ::= x \in \text{Var} \mid t \ u \mid \lambda x. t \mid n \mid \dots \mid s \bigvee t$$

so that the same program can reduce to different values:

$$3 \bigvee 4 \rightarrow 3 \qquad 3 \bigvee 4 \rightarrow 4$$

CBN λ -calculus with effects

A program like $\vdash \lambda x.x * x : \mathbf{Nat} \rightarrow \mathbf{Nat}$ can reduce to 9, 12 or 16 on the input $3 \textcircled{\vee} 4$:

$$(\lambda x.x * x) (3 \textcircled{\vee} 4) \rightarrow (3 \textcircled{\vee} 4) * (3 \textcircled{\vee} 4) \rightarrow^* 9 \text{ or } 12 \text{ or } 16$$

How to describe the semantics of a CBN program p ?

Idea: use “bags” to represent the outcomes of arguments of programs.

For the program $\lambda x.x * x$:

$$\begin{array}{lll} (x \leftarrow [3, 3]) & \mapsto & 9 \\ (x \leftarrow [3, 4]) & \mapsto & 12 \\ (x \leftarrow [4, 4]) & \mapsto & 16 \end{array}$$

CBN λ -calculus with effects

A program like $\vdash \lambda x. x * x : \mathbf{Nat} \rightarrow \mathbf{Nat}$ can reduce to 9, 12 or 16 on the input $3 \textcircled{\vee} 4$:

$$(\lambda x. x * x) (3 \textcircled{\vee} 4) \rightarrow (3 \textcircled{\vee} 4) * (3 \textcircled{\vee} 4) \rightarrow^* 9 \text{ or } 12 \text{ or } 16$$

How to describe the semantics of a CBN program p ?

Idea: use “**bags**” to represent the outcomes of arguments of programs.

For the program $\lambda x. x * x$:

$$\begin{array}{lll} (x \leftarrow [3, 3]) & \mapsto & 9 \\ (x \leftarrow [3, 4]) & \mapsto & 12 \\ (x \leftarrow [4, 4]) & \mapsto & 16 \end{array}$$

CBN λ -calculus with effects

A program like $\vdash \lambda x. x * x : \mathbf{Nat} \rightarrow \mathbf{Nat}$ can reduce to 9, 12 or 16 on the input $3 \vee 4$:

$$(\lambda x. x * x) (3 \vee 4) \rightarrow (3 \vee 4) * (3 \vee 4) \rightarrow^* 9 \text{ or } 12 \text{ or } 16$$

How to describe the semantics of a CBN program p ?

Idea: use “**bags**” to represent the outcomes of arguments of programs.

For the program $\lambda x. x * x$:

$$\begin{array}{lll} (x \leftarrow [3, 3]) & \mapsto & 9 \\ (x \leftarrow [3, 4]) & \mapsto & 12 \\ (x \leftarrow [4, 4]) & \mapsto & 16 \end{array}$$

CBN λ -calculus with effects

More generally, the outputs of $\lambda x.x * x$ can be correctly described by a (partial) function

$$f: !\mathbb{N} \rightarrow \mathbb{N}$$

where $!\mathbb{N}$ is the set of “bags” on \mathbb{N} .

But more general terms of type $\mathbf{Nat} \rightarrow \mathbf{Nat}$ can involve non-determinism, so that their interpretation should be a function

$$f: !\mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$$

or, equivalently, a **relation** $f \subseteq !\mathbb{N} \times \mathbb{N} \rightsquigarrow$ the relational model **Rel**

CBN λ -calculus with effects

More generally, the outputs of $\lambda x.x * x$ can be correctly described by a (partial) function

$$f : !\mathbb{N} \rightarrow \mathbb{N}$$

where $!\mathbb{N}$ is the set of “bags” on \mathbb{N} .

But more general terms of type $\mathbf{Nat} \rightarrow \mathbf{Nat}$ can involve non-determinism, so that their interpretation should be a function

$$f : !\mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$$

or, equivalently, a **relation** $f \subseteq !\mathbb{N} \times \mathbb{N} \rightsquigarrow$ the relational model **Rel**

The model **Rel** of **LL**

Objects: sets A, B, C , etc.

Morphisms $A \rightarrow B$: **relations** $R \subseteq A \times B$, i.e., sets of elements

$$a \multimap b$$

Exponential: $!A$ is $\mathcal{M}_{\text{fin}}(A)$, the set of **finite multisets** on A

(co)Kleisli category **Rel**_!: morphisms $A \rightarrow B$ are morphisms $!A \rightarrow B$ of **Rel**, that is, sets of elements

$$[a_1, \dots, a_n] \multimap b$$

Interpreting programs in \mathbf{Rel}_I

Since \mathbf{Rel}_I is cartesian closed, one can interpret **programs** inside it.

$x : \mathbf{Bool} \vdash \text{if } x \text{ then ff else tt} : \mathbf{Bool}$

interpreted as

$$\{ [\mathbf{tt}] \multimap \mathbf{ff}, \quad [\mathbf{ff}] \multimap \mathbf{tt} \} \quad (\subseteq \mathcal{M}_{\text{fin}}(\mathbf{Bool}) \times \mathbf{Bool})$$

Interpreting programs in \mathbf{Rel}_I

Since \mathbf{Rel}_I is **cartesian closed**, one can interpret **programs** inside it.

$x : \mathbf{Bool} \vdash \text{if } x \text{ then } (\text{if } x \text{ then ff else tt}) \text{ else } (\text{if } x \text{ then tt else ff}) : \mathbf{Bool}$

interpreted as

$$\{ [\mathbf{tt}, \mathbf{tt}] \multimap \mathbf{ff}, \quad [\mathbf{tt}, \mathbf{ff}] \multimap \mathbf{tt}, \quad [\mathbf{ff}, \mathbf{ff}] \multimap \mathbf{ff} \}$$

Interpreting programs in \mathbf{Rel}_l

Since \mathbf{Rel}_l is cartesian closed, one can interpret programs inside it.

$x : \mathbf{Bool} \vdash \text{if } x \text{ then (if } x \text{ then ff else tt) else (if } x \text{ then tt else ff)} : \mathbf{Bool}$

interpreted as

$$\{ [\mathbf{tt}, \mathbf{tt}] \multimap \mathbf{ff}, \quad [\mathbf{tt}, \mathbf{ff}] \multimap \mathbf{tt}, \quad [\mathbf{ff}, \mathbf{ff}] \multimap \mathbf{ff} \}$$

Here, two different executions **get identified** in the interpretation.

Interpreting programs in \mathbf{Rel}_l

Since \mathbf{Rel}_l is cartesian closed, one can interpret programs inside it.

$x : \mathbf{Bool} \vdash \text{if } x \text{ then (if } x \text{ then ff else tt) else (if } x \text{ then tt else ff)} : \mathbf{Bool}$

interpreted as

$$\{ [\text{tt}, \text{tt}] \multimap \text{ff}, \quad [\text{tt}, \text{ff}] \multimap \text{tt}, \quad [\text{ff}, \text{ff}] \multimap \text{ff} \}$$

Here, two different executions get identified in the interpretation.

Hence, \mathbf{Rel}_l aggregates different executions.

Witnesses of executions

How do we represent the different possible executions of a program?

Example of a program with non-determinism:

$$x : \mathbf{Nat}, y : \mathbf{Nat} \vdash x \oplus y : \mathbf{Nat}$$

has the executions

$$\begin{array}{ccc} x \leftarrow 40, & y \leftarrow 2 & \rightsquigarrow \text{output} = 40 \\ & & \rightsquigarrow \text{output} = 2 \end{array}$$

Executions are described by **witnesses**: triples (inputs, outputs, **reason**).

Example: $([x \leftarrow 40, y \leftarrow 2], 40, \oplus \text{ chose left})$ for the first execution.

Witnesses of executions

How do we represent the different possible executions of a program?

Example of a program with non-determinism:

$$x : \mathbf{Nat}, y : \mathbf{Nat} \vdash x \oplus y : \mathbf{Nat}$$

has the executions

$$\begin{array}{ccc} x \leftarrow 40, & y \leftarrow 2 & \rightsquigarrow \text{output} = 40 \\ & & \rightsquigarrow \text{output} = 2 \end{array}$$

Executions are described by **witnesses**: triples (inputs, outputs, **reason**).

Example: $([x \leftarrow 40, y \leftarrow 2], 40, \oplus \text{ chose left})$ for the first execution.

Witnesses of executions

How do we represent the different possible executions of a program?

Example of a program with non-determinism:

$$x : \mathbf{Nat}, y : \mathbf{Nat} \vdash x \oplus y : \mathbf{Nat}$$

has the executions

$$\begin{array}{ll} x \leftarrow 40, \quad y \leftarrow 2 & \rightsquigarrow \quad \text{output} = 40 \text{ because } \oplus \text{ chose left} \\ & \rightsquigarrow \quad \text{output} = 2 \text{ because } \oplus \text{ chose right} \end{array}$$

Executions are described by **witnesses**: triples (inputs, outputs, **reason**).

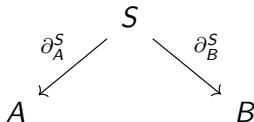
Example: $([x \leftarrow 40, y \leftarrow 2], 40, \oplus \text{ chose left})$ for the first execution.

Witnesses as spans

Witnesses of executions of a program $A \rightarrow B$: set of triples

$$S = \{ (a_i, b_i, r_i) \mid i \in \mathcal{I} \}.$$

There are canonical projections to A and B , so that S is in fact a **span**



Spans can be seen as **generalized relations**: 0, 1 or several “proofs” that $a \in A$ and $b \in B$ are related.

Spans and composition

What about composition?

program s :

$x : \mathbf{Nat} \vdash x \oplus_1 (x + 2) : \mathbf{Nat}$

program t :

$y : \mathbf{Nat} \vdash y \oplus_2 (2y) : \mathbf{Nat}$

Spans and composition

What about composition?

program s :

$x : \mathbf{Nat} \vdash x \textcircled{\vee}_1 (x + 2) : \mathbf{Nat}$

program t :

$y : \mathbf{Nat} \vdash y \textcircled{\vee}_2 (2y) : \mathbf{Nat}$

Witnesses between 0 and 2 of “ $t \circ s$ ”:

► $\textcircled{\vee}_1$ chose right ($s[x \leftarrow 0] \rightsquigarrow 2$) and $\textcircled{\vee}_2$ chose left ($t[y \leftarrow 2] \rightsquigarrow 2$).

Spans and composition

What about composition?

program s :

$x : \mathbf{Nat} \vdash x \oplus_1 (x + 2) : \mathbf{Nat}$

program t :

$y : \mathbf{Nat} \vdash y \oplus_2 (2y) : \mathbf{Nat}$

Witnesses between 0 and 0 of “ $t \circ s$ ”:

- ▶ \oplus_1 chose left ($s[x \leftarrow 0] \rightsquigarrow 0$) and \oplus_2 chose left ($t[y \leftarrow 0] \rightsquigarrow 0$); or
- ▶ \oplus_1 chose left ($s[x \leftarrow 0] \rightsquigarrow 0$) and \oplus_2 chose right ($t[y \leftarrow 0] \rightsquigarrow 0$).

Spans and composition

What about composition?

program s :

$x : \mathbf{Nat} \vdash x \textcircled{V}_1 (x + 2) : \mathbf{Nat}$

program t :

$y : \mathbf{Nat} \vdash y \textcircled{V}_2 (2y) : \mathbf{Nat}$

Witnesses between 2 and 4 of “ $t \circ s$ ”:

- ▶ \textcircled{V}_1 chose right ($s[x \leftarrow 2] \rightsquigarrow 4$) and \textcircled{V}_2 chose left ($t[y \leftarrow 4] \rightsquigarrow 4$); or
- ▶ \textcircled{V}_1 chose left ($s[x \leftarrow 2] \rightsquigarrow 2$) and \textcircled{V}_2 chose right ($t[y \leftarrow 2] \rightsquigarrow 4$).

Spans and composition

What about composition?

$$x : A \vdash s : B$$

$$y : B \vdash t : C$$

$$S = \{ (a_i, b_i, r_i) \mid i \in \mathcal{I} \} \quad T = \{ (b'_j, c_j, s_j) \mid j \in \mathcal{J} \}.$$

Spans and composition

What about composition?

$$x : A \vdash s : B$$

$$y : B \vdash t : C$$

$$S = \{ (a_i, b_i, r_i) \mid i \in \mathcal{I} \} \quad T = \{ (b'_j, c_j, s_j) \mid j \in \mathcal{J} \}.$$

Witnesses of “ $t \circ s$ ”:

$$T \odot S = \{ (a_i, c_j, (r_i, s_j)) \mid (a_i, b_i, r_i) \in S, (b'_j, c_j, s_j) \in T, b_i = b'_j \}$$

Spans and composition

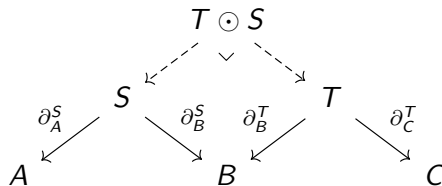
What about composition?

$$x : A \vdash s : B$$

$$y : B \vdash t : C$$

$$S = \{ (a_i, b_i, r_i) \mid i \in \mathcal{I} \} \quad T = \{ (b'_j, c_j, s_j) \mid j \in \mathcal{J} \}.$$

Witnesses of “ $t \circ s$ ”:



Towards a semantic model of spans

Spans can be used to describe the semantics of toy examples and compose them.

But what about more complex examples?

- ▶ CBN and effects, lambda abstractions, higher-order functions. . .

Idea: follow the constructions on **Rel**

- ▶ define a model of linear logic based on spans
- ▶ derive a cartesian closed (bi)category, in which we can interpret programs

A first bicategory of spans

Before defining a model of **LL**, we must start with some categorical structure.

Pullbacks are defined up to isomorphism

\rightsquigarrow associativity of composition \odot is expressed by a 2-dimensional structure

Given two spans $S, T: A \rightarrow B$, a **morphism** between S and T is $m: S \rightarrow T$ such that

$$\begin{array}{ccc} S & \xrightarrow{m} & T \\ \searrow \partial_A^S & = & \swarrow \partial_A^T \\ & A & \end{array} \quad \text{and} \quad \begin{array}{ccc} S & \xrightarrow{m} & T \\ \searrow \partial_B^S & = & \swarrow \partial_B^T \\ & B & \end{array} .$$

One gets a bicategory **Span** = **Span(Set)** of sets, spans and morphisms of spans.

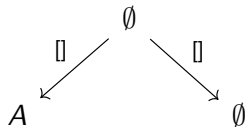
Some structure on **Span**

The cocartesian structure of **Set** translates to a cartesian structure on **Span**.

Some structure on **Span**

The cocartesian structure of **Set** translates to a cartesian structure on **Span**.

$\top \hat{=} \emptyset$ is the terminal object of **Span**.



Some structure on **Span**

The cocartesian structure of **Set** translates to a cartesian structure on **Span**.

$A \& B \triangleq A \sqcup B$ is the cartesian product on **Span**.

$$\begin{array}{ccc} \begin{array}{c} S \\ \swarrow \partial_X^S \quad \searrow \partial_A^S \\ X \qquad A \\ S: X \rightrightarrows A \end{array} & \begin{array}{c} T \\ \swarrow \partial_X^T \quad \searrow \partial_B^T \\ X \qquad B \\ T: X \rightrightarrows B \end{array} & \rightsquigarrow \begin{array}{c} S \sqcup T \\ \swarrow [\partial_X^S, \partial_X^T] \quad \searrow \partial_{A \sqcup B}^S \\ X \qquad A \sqcup B \\ \langle S, T \rangle: X \rightrightarrows A \& B \end{array} \end{array}$$

Some structure on **Span**

The cartesian structure of **Set** translates to a monoidal structure on **Span**.

$A \otimes B \triangleq A \times B$ gives a tensor product on **Span**.

$$\begin{array}{ccc} \begin{array}{c} S \\ \swarrow \partial_A^S \quad \searrow \partial_{A'}^S \\ A \qquad \qquad A' \\ S \end{array} & \begin{array}{c} T \\ \swarrow \partial_B^T \quad \searrow \partial_{B'}^T \\ B \qquad \qquad B' \\ T \end{array} & \rightsquigarrow \begin{array}{c} S \times T \\ \swarrow \partial_A^S \times \partial_B^T \quad \searrow \partial_{A'}^S \times \partial_{B'}^T \\ A \times B \qquad \qquad A' \times B' \\ S \otimes T \end{array} \end{array}$$

The exponential issue

An ingredient of an **LL** model: the exponential modality

Do we still have an exponential for **Span**?

The exponential issue

An ingredient of an **LL** model: the exponential modality

Do we still have an exponential for **Span**?

- First try: can we use $\mathcal{M}_{\text{fin}}(-)$ of **Rel** as exponential for **Span**?

Given $S \in \mathbf{Span}$, define

$$\mathcal{M}_{\text{fin}} \left(\begin{array}{ccc} & S & \\ \partial_A^S \swarrow & & \searrow \partial_B^S \\ A & & B \end{array} \right) = \begin{array}{ccc} & \mathcal{M}_{\text{fin}}(S) & \\ \mathcal{M}_{\text{fin}}(\partial_A^S) \swarrow & & \searrow \mathcal{M}_{\text{fin}}(\partial_B^S) \\ \mathcal{M}_{\text{fin}}(A) & & \mathcal{M}_{\text{fin}}(B) \end{array}$$

Problem: \mathcal{M}_{fin} **does not respect composition**, because pullbacks are not preserved. Thus, not a functor $\mathbf{Span} \rightarrow \mathbf{Span}$.

The exponential issue

An ingredient of an **LL** model: the exponential modality

Do we still have an exponential for **Span**?

- ▶ First try: can we use $\mathcal{M}_{\text{fin}}(-)$ of **Rel** as exponential for **Span**? No.
- ▶ Second try: can we use lists as exponential?

$$a_1, \dots, a_n \in A \quad \rightsquigarrow \quad [a_1; \dots ; a_n] \in \mathbf{List}(A)$$

The exponential issue

An ingredient of an **LL** model: the exponential modality

Do we still have an exponential for **Span**?

- First try: can we use $\mathcal{M}_{\text{fin}}(-)$ of **Rel** as exponential for **Span**? No.
- Second try: can we use lists as exponential?

$$a_1, \dots, a_n \in A \rightsquigarrow [a_1; \dots; a_n] \in \mathbf{List}(A)$$
$$\mathbf{List} \left(\begin{array}{ccc} & S & \\ \partial_A^S \swarrow & & \searrow \partial_B^S \\ A & & B \end{array} \right) = \begin{array}{ccc} & \mathbf{List}(S) & \\ \mathbf{List}(\partial_A^S) \swarrow & & \searrow \mathbf{List}(\partial_B^S) \\ \mathbf{List}(A) & & \mathbf{List}(B) \end{array}$$

The exponential issue

An ingredient of an **LL** model: the exponential modality

Do we still have an exponential for **Span**?

- First try: can we use $\mathcal{M}_{\text{fin}}(-)$ of **Rel** as exponential for **Span**? No.
- Second try: can we use lists as exponential?

$$a_1, \dots, a_n \in A \rightsquigarrow [a_1; \dots; a_n] \in \mathbf{List}(A)$$

$$\mathbf{List} \left(\begin{array}{ccc} & S & \\ \partial_A^S \swarrow & & \searrow \partial_B^S \\ A & & B \end{array} \right) = \begin{array}{ccc} & \mathbf{List}(S) & \\ \mathbf{List}(\partial_A^S) \swarrow & & \searrow \mathbf{List}(\partial_B^S) \\ \mathbf{List}(A) & & \mathbf{List}(B) \end{array}$$

We now have a (pseudo)functor, but no **Seely equivalence**

$$\text{see}_{A,B}: \mathbf{List} A \otimes \mathbf{List} B \begin{array}{c} \xrightarrow{\quad} \\ \simeq \\ \xleftarrow{\quad} \end{array} \mathbf{List}(A \& B) : \overline{\text{see}}_{A,B} \in \mathbf{Span}$$

because $[b_1; a_1; a_2; b_2] \neq [a_1; a_2; b_1; b_2]$: **lack of symmetries**

The exponential issue

An ingredient of an **LL** model: the exponential modality

Do we still have an exponential for **Span**?

- ▶ First try: can we use $\mathcal{M}_{\text{fin}}(-)$ of **Rel** as exponential for **Span**? No.
- ▶ Second try: can we use lists as exponential? Probably no.

$$a_1, \dots, a_n \in A \quad \rightsquigarrow \quad [a_1; \dots ; a_n] \in \mathbf{List}(A)$$

Span is dead, long live Span!

Problem:

- ▶ our spans are **set-based**
- ▶ there is **no adequate Seely equivalence** in this setting

We must change the kind of spans that we use.

Span is dead, long live Span!

Let **Gpd** be the 2-category of groupoids, functors and **natural transformations**.

\rightsquigarrow within groupoids, there are symmetries between objects:

$$[b_1; a_1; a_2; b_2] \cong [a_1; a_2; b_1; b_2] \in \mathbf{List}^*(A \sqcup B)$$

We now have a Seely equivalence

$$\text{see}_{A,B}: \mathbf{List}^* A \times \mathbf{List}^* B \begin{array}{c} \xrightarrow{\quad} \\ \simeq \\ \xleftarrow{\quad} \end{array} \mathbf{List}^*(A \sqcup B) : \overline{\text{see}}_{A,B} \in \mathbf{Gpd}$$

because the symmetries allow us to reindex:

$$\begin{array}{lcl} \overline{\text{see}}_{A,B} \circ \text{see}_{A,B} & = & \text{id}_{\mathbf{List}^* A \times \mathbf{List}^* B} \\ \text{see}_{A,B} \circ \overline{\text{see}}_{A,B} & \cong & \text{id}_{\mathbf{List}^*(A \sqcup B)} \end{array}$$

Span is dead, long live Span!

Let **Gpd** be the 2-category of groupoids, functors and **natural transformations**.

\rightsquigarrow within groupoids, there are symmetries between objects:

$$[b_1; a_1; a_2; b_2] \cong [a_1; a_2; b_1; b_2] \in \mathbf{List}^*(A \sqcup B)$$

We now have a Seely equivalence

$$\mathrm{see}_{A,B}: \mathbf{List}^* A \times \mathbf{List}^* B \begin{array}{c} \xrightarrow{\quad} \\ \simeq \\ \xleftarrow{\quad} \end{array} \mathbf{List}^*(A \sqcup B) : \overline{\mathrm{see}}_{A,B} \in \mathbf{Gpd}$$

because the symmetries allow us to reindex:

$$\begin{array}{ccc} \overline{\mathrm{see}}_{A,B} \circ \mathrm{see}_{A,B} & = & \mathrm{id}_{\mathbf{List}^* A \times \mathbf{List}^* B} \\ \mathrm{see}_{A,B} \circ \overline{\mathrm{see}}_{A,B} & \cong & \mathrm{id}_{\mathbf{List}^*(A \sqcup B)} \end{array}$$

Span is dead, long live **Span**!

We (re)define **Span** as **Span(Gpd)**

Span is dead, long live **Span**!

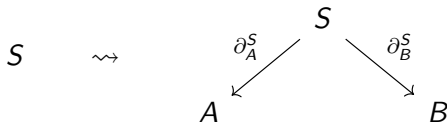
We (re)define **Span** as **Span(Gpd)**

- ▶ objects: groupoids A, B, \dots

Span is dead, long live **Span**!

We (re)define **Span** as **Span(Gpd)**

- ▶ objects: groupoids A, B, \dots
- ▶ 1-morphisms: spans S, T, \dots



Span is dead, long live **Span**!

We (re)define **Span** as **Span**(Gpd)

- ▶ objects: groupoids A, B, \dots
- ▶ 1-morphisms: spans S, T, \dots
- ▶ 2-morphisms: **pseudo-commutative** triangles $(F, \phi), (G, \psi), \dots$

$$(F, \phi): S \Rightarrow T \quad \rightsquigarrow \quad \begin{array}{ccc} S & \xrightarrow{F} & T \\ & \searrow \partial_A^S & \swarrow \partial_A^T \\ & A & \end{array} \quad \text{and} \quad \begin{array}{ccc} S & \xrightarrow{F} & T \\ & \searrow \partial_B^S & \swarrow \partial_B^T \\ & B & \end{array}$$

The diagram shows a 2-morphism $(F, \phi): S \Rightarrow T$ represented by a pseudo-commutative triangle. The triangle has vertices S , T , and A . The top edge is $F: S \rightarrow T$. The bottom-left edge is $\partial_A^S: S \rightarrow A$. The bottom-right edge is $\partial_A^T: T \rightarrow A$. A 2-morphism $\phi^A: \partial_A^S \Rightarrow \partial_A^T$ is shown as a double arrow between these two edges. The word "and" is placed between this triangle and another identical one, but with vertex A replaced by B and the edges labeled ∂_B^S , ∂_B^T , and ϕ^B .

Span is dead, long live Span!

We (re)define **Span** as **Span(Gpd)**

- ▶ objects: groupoids A, B, \dots
- ▶ 1-morphisms: spans S, T, \dots
- ▶ 2-morphisms: **pseudo-commutative** triangles $(F, \phi), (G, \psi), \dots$

$$(F, \phi): S \Rightarrow T \quad \rightsquigarrow \quad \begin{array}{ccc} S & \xrightarrow{F} & T \\ & \searrow \partial_A^S & \swarrow \partial_A^T \\ & A & \end{array} \quad \text{and} \quad \begin{array}{ccc} S & \xrightarrow{F} & T \\ & \searrow \partial_B^S & \swarrow \partial_B^T \\ & B & \end{array}$$

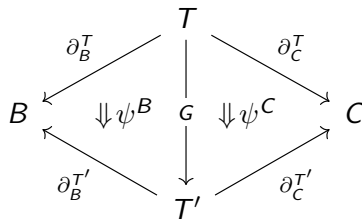
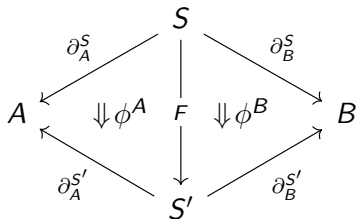
ϕ^A and ϕ^B are represented by double arrows between the bottom nodes in the triangles above.

We can now hope that the Seely equivalence of **Gpd** lifts in **Span(Gpd)**.

The horizontal composition issue

But is $\mathbf{Span} = \mathbf{Span}(\mathbf{Gpd})$ a bicategory?

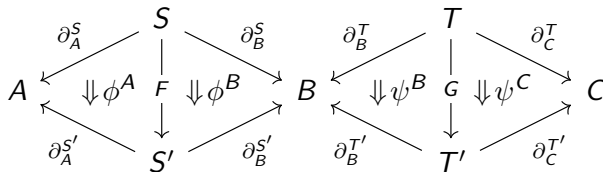
In particular, are we able to define a **horizontal composition**?



The horizontal composition issue

But is $\mathbf{Span} = \mathbf{Span}(\mathbf{Gpd})$ a bicategory?

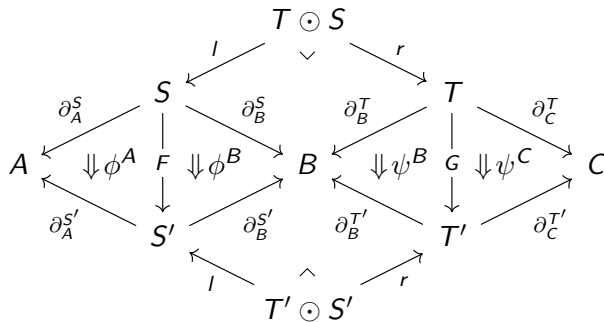
In particular, are we able to define a **horizontal composition**?



The horizontal composition issue

But is $\mathbf{Span} = \mathbf{Span}(\mathbf{Gpd})$ a bicategory?

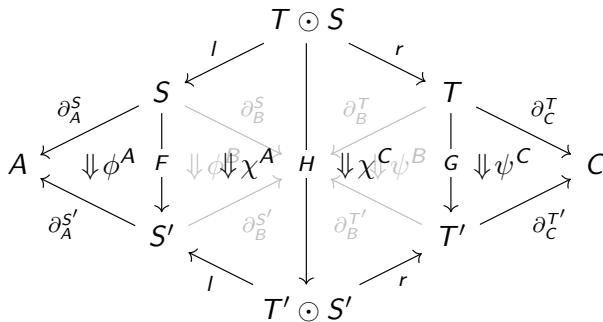
In particular, are we able to define a **horizontal composition**?



The horizontal composition issue

But is $\mathbf{Span} = \mathbf{Span}(\mathbf{Gpd})$ a bicategory?

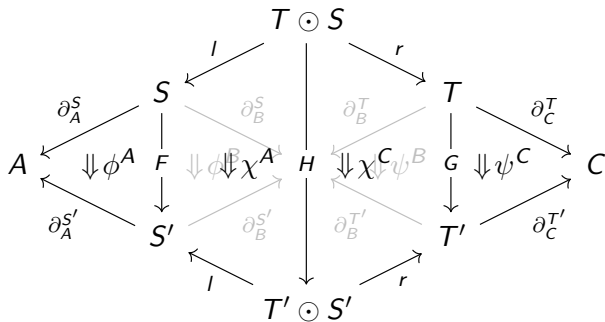
In particular, are we able to define a **horizontal composition**?



The horizontal composition issue

But is $\mathbf{Span} = \mathbf{Span}(\mathbf{Gpd})$ a bicategory?

In particular, are we able to define a **horizontal composition**?



\leadsto not always possible to find such H, χ^A, χ^C ! So **Span** is not a bicategory!

Taking into account symmetries

This problem actually arises in other proof-relevant bicategorical models and needs to be addressed.

- ▶ generalized species of structures [**fiore2008cartesian**]: quotient of the witnesses through a **coend**
- ▶ template games [**mellies2019template**]: use of **deformations** to correctly align the witnesses

Span is dead (again), long live Thin!

Our solution: we add structures to constrain the spans and the morphisms of spans, so that horizontal composition exists.

A **thin span** is a tuple

$$\mathcal{A} = (A, A_-, A_+, \mathcal{U}_A, T_A)$$

where A is a groupoid and the remainder is the data associated with two orthogonality relations.

Span is dead (again), long live Thin!

Our solution: we add structures to constrain the spans and the morphisms of spans, so that horizontal composition exists.

A **thin span** is a tuple

$$\mathcal{A} = (A, \dots)$$

where A is a groupoid and the remainder is ~~the data associated with two orthogonality relations~~ **black box magic**.

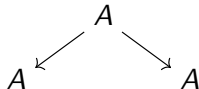
Span is dead (again), long live **Thin**!

Our solution: we add structures to constrain the spans and the morphisms of spans, so that horizontal composition exists.

Theorem (C., F.)

We get a bicategory **Thin**:

- ▶ objects: *thin groupoids* $\mathcal{A}, \mathcal{B}, \dots$
- ▶ 1-morphisms: **thin spans** (i.e. spans compatible with the black box magic)
- ▶ 2-morphisms: **positive morphisms** (i.e. morphisms of spans compatible with the black box magic)
- ▶ 1-identity on \mathcal{A} :



- ▶ 1-composition: *pullbacks*

The exponential modality

Recall that the exponential $! : \mathbf{Rel} \rightarrow \mathbf{Rel}$ is derived from the monad $\mathcal{M}_{\text{fin}} : \mathbf{Set} \rightarrow \mathbf{Set}$.

We derive an exponential $! : \mathbf{Thin} \rightarrow \mathbf{Thin}$ from the monad $\mathbf{List}^* : \mathbf{Gpd} \rightarrow \mathbf{Gpd}$.

The exponential modality

The monad $\mathbf{List}^*: \mathbf{Gpd} \rightarrow \mathbf{Gpd}$? The “free strict symmetric monoidal construction”.

To $A \in \mathbf{Gpd}$, associates $\mathbf{List}^*(A) \in \mathbf{Gpd}$:

- ▶ objects: lists $[a_1; \dots; a_n] \in \mathbf{List}(\mathbf{Ob}(A))$;
- ▶ morphisms $[a_1; \dots; a_n] \rightarrow [a'_1; \dots; a'_m]$: pairs $(\pi, (f_i)_{i \in I})$ where
 - ▶ π is a bijection $\{1, \dots, n\} \rightarrow \{1, \dots, m\}$;
 - ▶ f_i is a morphism $a_i \rightarrow a'_{\pi(i)} \in A$.

The unit $\eta_A: A \rightarrow \mathbf{List}^*(A)$: maps $a \in A$ to $[a]$;

The multiplication $\mu_A: \mathbf{List}^*(\mathbf{List}^*(A)) \rightarrow \mathbf{List}^*(A)$: merges lists of lists into lists.

The exponential modality

The monad $\mathbf{List}^*: \mathbf{Gpd} \rightarrow \mathbf{Gpd}$? The “free strict symmetric monoidal construction”.

To $A \in \mathbf{Gpd}$, associates $\mathbf{List}^*(A) \in \mathbf{Gpd}$:

- ▶ objects: lists $[a_1; \dots; a_n] \in \mathbf{List}(\mathbf{Ob}(A))$;
- ▶ morphisms $[a_1; \dots; a_n] \rightarrow [a'_1; \dots; a'_m]$: pairs $(\pi, (f_i)_{i \in I})$ where
 - ▶ π is a bijection $\{1, \dots, n\} \rightarrow \{1, \dots, m\}$;
 - ▶ f_i is a morphism $a_i \rightarrow a'_{\pi(i)} \in A$.

The unit $\eta_A: A \rightarrow \mathbf{List}^*(A)$: maps $a \in A$ to $[a]$;

The multiplication $\mu_A: \mathbf{List}^*(\mathbf{List}^*(A)) \rightarrow \mathbf{List}^*(A)$: merges lists of lists into lists.

The exponential modality

The monad $\mathbf{List}^*: \mathbf{Gpd} \rightarrow \mathbf{Gpd}$? The “free strict symmetric monoidal construction”.

To $A \in \mathbf{Gpd}$, associates $\mathbf{List}^*(A) \in \mathbf{Gpd}$:

- ▶ objects: lists $[a_1; \dots; a_n] \in \mathbf{List}(\mathbf{Ob}(A))$;
- ▶ morphisms $[a_1; \dots; a_n] \rightarrow [a'_1; \dots; a'_m]$: pairs $(\pi, (f_i)_{i \in I})$ where
 - ▶ π is a bijection $\{1, \dots, n\} \rightarrow \{1, \dots, m\}$;
 - ▶ f_i is a morphism $a_i \rightarrow a'_{\pi(i)} \in A$.

The unit $\eta_A: A \rightarrow \mathbf{List}^*(A)$: maps $a \in A$ to $[a]$;

The multiplication $\mu_A: \mathbf{List}^*(\mathbf{List}^*(A)) \rightarrow \mathbf{List}^*(A)$: merges lists of lists into lists.

The exponential modality

The monad $\mathbf{List}^*: \mathbf{Gpd} \rightarrow \mathbf{Gpd}$? The “free strict symmetric monoidal construction”.

To $A \in \mathbf{Gpd}$, associates $\mathbf{List}^*(A) \in \mathbf{Gpd}$:

- ▶ objects: lists $[a_1; \dots; a_n] \in \mathbf{List}(\mathbf{Ob}(A))$;
- ▶ morphisms $[a_1; \dots; a_n] \rightarrow [a'_1; \dots; a'_m]$: pairs $(\pi, (f_i)_{i \in I})$ where
 - ▶ π is a bijection $\{1, \dots, n\} \rightarrow \{1, \dots, m\}$;
 - ▶ f_i is a morphism $a_i \rightarrow a'_{\pi(i)} \in A$.

The unit $\eta_A: A \rightarrow \mathbf{List}^*(A)$: maps $a \in A$ to $[a]$;

The multiplication $\mu_A: \mathbf{List}^*(\mathbf{List}^*(A)) \rightarrow \mathbf{List}^*(A)$: merges lists of lists into lists.

The exponential modality

The monad $\mathbf{List}^*: \mathbf{Gpd} \rightarrow \mathbf{Gpd}$? The “free strict symmetric monoidal construction”.

To $A \in \mathbf{Gpd}$, associates $\mathbf{List}^*(A) \in \mathbf{Gpd}$:

- ▶ objects: lists $[a_1; \dots; a_n] \in \mathbf{List}(\mathbf{Ob}(A))$;
- ▶ morphisms $[a_1; \dots; a_n] \rightarrow [a'_1; \dots; a'_m]$: pairs $(\pi, (f_i)_{i \in I})$ where
 - ▶ π is a bijection $\{1, \dots, n\} \rightarrow \{1, \dots, m\}$;
 - ▶ f_i is a morphism $a_i \rightarrow a'_{\pi(i)} \in A$.

The unit $\eta_A: A \rightarrow \mathbf{List}^*(A)$: maps $a \in A$ to $[a]$;

The multiplication $\mu_A: \mathbf{List}^*(\mathbf{List}^*(A)) \rightarrow \mathbf{List}^*(A)$: merges lists of lists into lists.

The exponential modality

We get an exponential

$$!: \mathbf{Thin} \rightarrow \mathbf{Thin}$$

where

$$!\mathcal{A} \quad \hat{=} \quad (\mathbf{List}^*(A), \dots)$$

for every thin groupoid \mathcal{A} and

$$! \left(\begin{array}{ccc} & S & \\ \partial_A^S \swarrow & & \searrow \partial_B^S \\ A & & B \end{array} \right) \quad \hat{=} \quad \begin{array}{ccc} & \mathbf{List}^*(S) & \\ \mathbf{List}^*(\partial_A^S) \swarrow & & \searrow \mathbf{List}^*(\partial_B^S) \\ \mathbf{List}^*(A) & & \mathbf{List}^*(B) \end{array}$$

for every thin span $S: \mathcal{A} \rightarrow \mathcal{B}$.

The exponential modality

The structure of comonad of $!$ is derived from the monad structure of \mathbf{List}^* .

Given a thin groupoid \mathcal{A} ,

$$\check{\eta}_{\mathcal{A}} = \begin{array}{ccc} & A & \\ \eta_A \swarrow & & \searrow \text{id}_A \\ \mathbf{List}^*(A) & & A \end{array}$$

$$\check{\mu}_{\mathcal{A}} = \begin{array}{ccc} & \mathbf{List}^*(\mathbf{List}^*(A)) & \\ \mu_A \swarrow & & \searrow \text{id}_{!!A} \\ \mathbf{List}^*(A) & & \mathbf{List}^*(\mathbf{List}^*(A)) \end{array} .$$

The Kleisli bicategory

We thus get a Kleisli bicategory $\mathbf{Thin}_!$ with $! = \mathbf{List}^*$, whose 1-cells $\mathcal{A} \rightarrow \mathcal{B}$ are of the form

$$\begin{array}{ccc} & S & \\ \partial_{!A}^S \swarrow & & \searrow \partial_B^S \\ !A & & B \end{array} .$$

The Kleisli bicategory

We thus get a Kleisli bicategory $\mathbf{Thin}_!$ with $! = \mathbf{List}^*$, whose 1-cells $\mathcal{A} \rightarrow \mathcal{B}$ are of the form

$$\begin{array}{ccc} & S & \\ \partial_{!A}^S \swarrow & & \searrow \partial_B^S \\ !A & & B \end{array} .$$

In categorical models of \mathbf{LL} , the Kleisli category is cartesian closed.

The Kleisli bicategory

We thus get a Kleisli bicategory $\mathbf{Thin}_!$ with $! = \mathbf{List}^*$, whose 1-cells $\mathcal{A} \rightarrow \mathcal{B}$ are of the form

$$\begin{array}{ccc} & S & \\ \partial_{!A}^S \swarrow & & \searrow \partial_B^S \\ !A & & B \end{array} .$$

In categorical models of \mathbf{LL} , the Kleisli category is cartesian closed.

Theorem (C., F.)

The bicategory $\mathbf{Thin}_!$ is cartesian closed.

Outline

From relations to spans

Interpreting programs

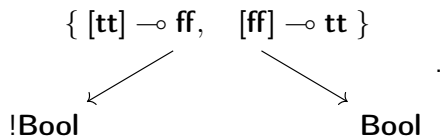
A rigid intersection type system

Examples of interpretations

Example 1:

$x : \mathbf{Bool} \vdash \text{if } x \text{ then ff else tt} : \mathbf{Bool}$

interpreted as the span (which happens to be a relation)



Examples of interpretations

Example 2:

$x : \mathbf{Bool} \vdash \text{if } x \text{ then } (\text{if } x \text{ then ff else tt}) \text{ else } (\text{if } x \text{ then tt else ff}) : \mathbf{Bool}$

interpreted as the span (which happens to be a relation)

$\{ [\text{tt}; \text{tt}] \multimap \text{ff}, [\text{tt}; \text{ff}] \multimap \text{tt}, [\text{ff}; \text{tt}] \multimap \text{tt}, [\text{ff}; \text{ff}] \multimap \text{ff} \}$

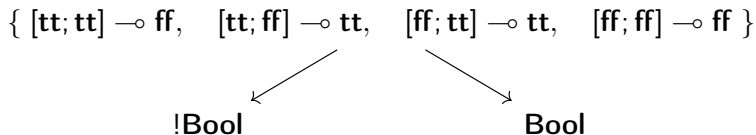


Examples of interpretations

Example 2:

$x : \mathbf{Bool} \vdash \text{if } x \text{ then (if } x \text{ then ff else tt) else (if } x \text{ then tt else ff)} : \mathbf{Bool}$

interpreted as the span (which happens to be a relation)



to compare with the interpretation in $\mathbf{Rel}_!$:

$$\{ [\mathbf{tt}, \mathbf{tt}] \multimap \mathbf{ff}, \quad [\mathbf{tt}, \mathbf{ff}] \multimap \mathbf{tt}, \quad [\mathbf{ff}, \mathbf{ff}] \multimap \mathbf{tt} \}.$$

Categorical interpretation

Since **Thin**_I is cartesian closed, we can interpret simply-typed λ -calculus. What would it look like?

Considered types:

$$A, B, \dots ::= \mathbf{Bool} \mid A \rightarrow B$$

Interpretation $\llbracket A \rrbracket$ of a type A :

$$\llbracket \mathbf{Bool} \rrbracket = 1 \sqcup 1 \qquad \llbracket A \rightarrow B \rrbracket = !\llbracket A \rrbracket \times \llbracket B \rrbracket$$

Interpretation $\llbracket \Gamma \rrbracket$ of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$\llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \sqcup \dots \sqcup \llbracket A_n \rrbracket$$

Categorical interpretation

Since \mathbf{Thin}_1 is cartesian closed, we can interpret simply-typed λ -calculus. What would it look like?

Considered types:

$$A, B, \dots ::= \mathbf{Bool} \mid A \rightarrow B$$

Interpretation $\llbracket A \rrbracket$ of a type A :

$$\llbracket \mathbf{Bool} \rrbracket = 1 \sqcup 1 \qquad \llbracket A \rightarrow B \rrbracket = !\llbracket A \rrbracket \times \llbracket B \rrbracket$$

Interpretation $\llbracket \Gamma \rrbracket$ of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$\llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \sqcup \dots \sqcup \llbracket A_n \rrbracket$$

Categorical interpretation

Since \mathbf{Thin}_I is cartesian closed, we can interpret simply-typed λ -calculus. What would it look like?

Considered types:

$$A, B, \dots ::= \mathbf{Bool} \quad | \quad A \rightarrow B$$

Interpretation $\llbracket A \rrbracket$ of a type A :

$$\llbracket \mathbf{Bool} \rrbracket = 1 \sqcup 1 \qquad \llbracket A \rightarrow B \rrbracket = !\llbracket A \rrbracket \times \llbracket B \rrbracket$$

Interpretation $\llbracket \Gamma \rrbracket$ of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$\llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \sqcup \dots \sqcup \llbracket A_n \rrbracket$$

Categorical interpretation

Since \mathbf{Thin}_I is cartesian closed, we can interpret simply-typed λ -calculus. What would it look like?

Considered types:

$$A, B, \dots ::= \mathbf{Bool} \quad | \quad A \rightarrow B$$

Interpretation $\llbracket A \rrbracket$ of a type A :

$$\llbracket \mathbf{Bool} \rrbracket = 1 \sqcup 1 \qquad \llbracket A \rightarrow B \rrbracket = !\llbracket A \rrbracket \times \llbracket B \rrbracket$$

Interpretation $\llbracket \Gamma \rrbracket$ of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$\llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \sqcup \dots \sqcup \llbracket A_n \rrbracket$$

Categorical interpretation

Given a derivation of $\Gamma \vdash t : A$ in STLC, its **categorical interpretation** is a span

$$\begin{array}{ccc} & \langle t \rangle & \\ \partial_l^{\langle t \rangle} \swarrow & & \searrow \partial_r^{\langle t \rangle} \\ !\langle \Gamma \rangle & & \langle A \rangle \end{array}$$

by induction on the derivation.

- ▶ this is automatically derived from the cartesian closed structure
- ▶ but uneasy to describe syntactically (notably the projection on $!\langle \Gamma \rangle$)

The bagged interpretation

We introduce a more relevant interpretation of contexts.

Bagged interpretation of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$[\![\Gamma]\!] = [\![A_1]\!] \times \dots \times [\![A_n]\!] \in \mathbf{Thin}$$

(Recall that $!(\Gamma) = !((A_1) \sqcup \dots \sqcup (A_n))$)

Proposition

The underlying groupoid of $[\![\Gamma]\!]$ is equipped with a structure of monoid.

► multiplication: given $\gamma = (\gamma_1, \dots, \gamma_n)$ and $\delta = (\delta_1, \dots, \delta_n)$ in $[\![\Gamma]\!]$,

$$\gamma \vec{\oplus} \delta = (\gamma_1 ++ \delta_1, \dots, \gamma_n ++ \delta_n)$$

where $\gamma_i ++ \delta_i$ is the concatenation of lists in $[\![A_i]\!]$

The bagged interpretation

We introduce a more relevant interpretation of contexts.

Bagged interpretation of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$[\![\Gamma]\!] = [\![A_1]\!] \times \dots \times [\![A_n]\!] \in \mathbf{Thin}$$

(Recall that $!([\![\Gamma]\!]) = !([\![A_1]\!] \sqcup \dots \sqcup [\![A_n]\!])$)

Proposition

The underlying groupoid of $[\![\Gamma]\!]$ is equipped with a structure of monoid.

► multiplication: given $\gamma = (\gamma_1, \dots, \gamma_n)$ and $\delta = (\delta_1, \dots, \delta_n)$ in $[\![\Gamma]\!]$,

$$\gamma \vec{\oplus} \delta = (\gamma_1 ++ \delta_1, \dots, \gamma_n ++ \delta_n)$$

where $\gamma_i ++ \delta_i$ is the concatenation of lists in $[\![A_i]\!]$

The bagged interpretation

We introduce a more relevant interpretation of contexts.

Bagged interpretation of a context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$:

$$\llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket \in \mathbf{Thin}$$

(Recall that $\llbracket \Gamma \rrbracket = \llbracket (A_1) \sqcup \dots \sqcup (A_n) \rrbracket$)

Proposition

The underlying groupoid of $\llbracket \Gamma \rrbracket$ is equipped with a structure of monoid.

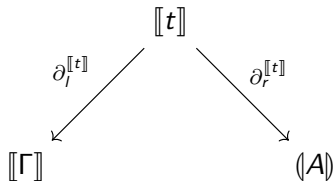
► multiplication: given $\gamma = (\gamma_1, \dots, \gamma_n)$ and $\delta = (\delta_1, \dots, \delta_n)$ in $\llbracket \Gamma \rrbracket$,

$$\gamma \vec{\oplus} \delta = (\gamma_1 ++ \delta_1, \dots, \gamma_n ++ \delta_n)$$

where $\gamma_i ++ \delta_i$ is the concatenation of lists in $\llbracket A_i \rrbracket$

The bagged interpretation

Given $\Gamma = x_1 : A_1, \dots, x_n : A_n$ and a derivation $\Gamma \vdash t : A$, we define a thin span



by induction on the derivation.

The bagged interpretation

Variable case: for $\Gamma = x_1 : A_1, \dots, x_n : A_n$ and $i \in \{ 1, \dots, n \}$,

$$\begin{array}{c}
 \llbracket x_i \rrbracket = \begin{array}{ccc}
 & & \langle A_i \rangle \\
 & \swarrow & \searrow \text{id}_{\langle A_i \rangle} \\
 \langle \square, \dots, \eta_{\langle A_i \rangle}, \dots, \square \rangle & & \langle A_i \rangle \\
 \nwarrow & & \\
 \prod_{j=1}^n \langle A_j \rangle & &
 \end{array}
 \end{array}$$

\rightsquigarrow “for every $a \in \langle A_i \rangle$, there is one computation consuming $(\square, \dots, [a], \dots, \square)$ and producing a ”

The bagged interpretation

λ -abstraction case: given a derivation of $(\Gamma, x : A) \vdash t : B$,

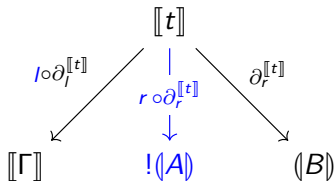
The bagged interpretation

λ -abstraction case: given a derivation of $(\Gamma, x : A) \vdash t : B$,

$$\begin{array}{ccc} & \llbracket t \rrbracket & \\ \partial_l^{\llbracket t \rrbracket} \swarrow & & \searrow \partial_r^{\llbracket t \rrbracket} \\ \llbracket \Gamma \rrbracket \times \llbracket A \rrbracket & & \llbracket B \rrbracket \end{array}$$

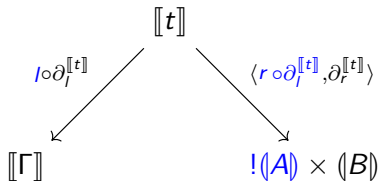
The bagged interpretation

λ -abstraction case: given a derivation of $(\Gamma, x : A) \vdash t : B$,



The bagged interpretation

λ -abstraction case: given a derivation of $(\Gamma, x : A) \vdash t : B$,



The bagged interpretation

λ -abstraction case: given a derivation of $(\Gamma, x : A) \vdash t : B$,

$$\llbracket \lambda x. t \rrbracket = \begin{array}{c} \llbracket t \rrbracket \\ \swarrow \scriptstyle l \circ \partial_l^{\llbracket t \rrbracket} \quad \searrow \scriptstyle \langle r \circ \partial_l^{\llbracket t \rrbracket}, \partial_r^{\llbracket t \rrbracket} \rangle \\ \llbracket \Gamma \rrbracket \qquad \qquad \qquad !\langle A \rangle \times \langle B \rangle \end{array}$$

\rightsquigarrow “if t consumes $(\gamma, [a_i]_i)$ to produce b , then $\lambda x. t$ consumes γ to produce $[a_i]_i \multimap b$ ”

The bagged interpretation

Application case: given a derivation of $\Gamma \vdash t : A \rightarrow B$ and of $\Gamma \vdash u : A$,

$$\begin{array}{c}
\llbracket t \ u \rrbracket \\
= \\
\begin{array}{c}
\begin{array}{c}
\llbracket t \rrbracket \times !\llbracket u \rrbracket \\
\swarrow \partial_l^{\llbracket t \rrbracket} \times !\partial_l^{\llbracket u \rrbracket} \\
\llbracket \Gamma \rrbracket \times !\llbracket \Gamma \rrbracket \\
\swarrow \dots \\
\llbracket \Gamma \rrbracket
\end{array}
\qquad
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\llbracket t \rrbracket \times !\llbracket u \rrbracket \\
\swarrow \partial_r^{\llbracket t \rrbracket} \times !\partial_r^{\llbracket u \rrbracket} \\
(!\langle A \rangle \times \langle B \rangle) \times !\langle A \rangle
\end{array}
\qquad
\begin{array}{c}
\text{linev}_{!\langle A \rangle, \llbracket B \rrbracket} \\
\swarrow \partial_l^{\text{linev}_{!\langle A \rangle, \llbracket B \rrbracket}} \\
\langle B \rangle
\end{array}
\end{array}
\qquad
\begin{array}{c}
\text{linev}_{!\langle A \rangle, \llbracket B \rrbracket} \\
\searrow \partial_r^{\text{linev}_{!\langle A \rangle, \llbracket B \rrbracket}} \\
\langle B \rangle
\end{array}
\end{array}
\end{array}
\end{array}$$

\rightsquigarrow “if t consumes γ to produce $[a_i]_i \multimap b$ and u consumes δ to produce $[a_i]_i$, then $t \mid u$ consumes $\gamma \oplus \delta$ to produce b ”

The bagged interpretation

Given $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$, write see_Γ for the Seely morphism

$$\text{see}_\Gamma : \quad !(\langle A_1 \rangle \sqcup \dots \sqcup \langle A_n \rangle) \quad \rightarrow \quad ![A_1] \times \dots \times ![A_n] \quad \in \mathbf{Gpd}$$

Theorem (Compatibility)

Given a derivation $\Gamma \vdash t : A$, we have a canonical isomorphism of spans

$$\begin{array}{ccc}
 & \langle t \rangle & \\
 \partial_l^{\langle t \rangle} \swarrow & & \searrow \partial_r^{\langle t \rangle} \\
 !(\langle \Gamma \rangle) & & \langle A \rangle \\
 \text{see}_\Gamma \swarrow & & \\
 [\Gamma] & &
 \end{array}
 \quad \cong \quad
 \begin{array}{ccc}
 & [t] & \\
 \partial_l^{[t]} \swarrow & & \searrow \partial_r^{[t]} \\
 [\Gamma] & & \langle A \rangle
 \end{array}$$

in **Thin**.

Outline

From relations to spans

Interpreting programs

A rigid intersection type system

Intersection type systems

[de2018execution]: interpretations of programs in **Rel** can be presented syntactically through an **intersection type system**.

$$\Gamma \vdash t : A \quad \rightsquigarrow \quad \Theta \vdash t : \alpha \triangleleft A \quad \text{and} \quad \Theta \vdash_{\text{m}} t : [\alpha_1, \dots, \alpha_n] \triangleleft A$$

[olimpieri2021intersection]: interpretations of programs in **Esp** can also be presented syntactically through an **intersection type system**

Can we have a similar presentation for **Thin**?

Intersection type systems

Idea of intersection type system: give several types to a pure λ -term:

$$\Gamma \vdash t : \tau_1 \cap \cdots \cap \tau_n$$

In this setting, we can type $\lambda x.xx$:

$$\vdash \lambda x.xx : ((A \rightarrow A) \cap A) \rightarrow A$$

The system is said

- ▶ **non-commutative** when $\sigma \cap \tau \neq \tau \cap \sigma$
- ▶ **non-idempotent** when $\sigma \cap \sigma \neq \sigma$

Intersection type systems

In the context of **Rel** and **Thin**, it is better to change perspective.

Broke:

- ▶ simply-typed λ -calculus is pure λ -calculus with types
- ▶ intersection types = “typing a term with several types”

Woke:

- ▶ pure λ -calculus is simply-typed λ -calculus with a reflexive type/object
- ▶ intersection types = “assigning different **values** to a term”

[de2018execution]: the relational model be described syntactically by an intersection type system, with multisets as bags

For **Thin**: we use the same system with **lists** as bags

ITS for **Thin**

Simple types considered:

$$A, B, \dots ::= \mathbf{Bool} \mid A \rightarrow B$$

Refinement ~~types~~ **values** and intersection ~~types~~ **values**:

$$\begin{aligned}\alpha, \beta, \dots &::= \mathbf{tt} \mid \mathbf{ff} \mid \kappa \multimap \alpha \\ \kappa, \lambda, \dots &::= [\alpha_1, \dots, \alpha_n] \quad (n \in \mathbb{N})\end{aligned}$$

where $[\alpha_1, \dots, \alpha_n]$ is a **list** of elements.

Refinement judgements $\alpha \triangleleft A$ and $\kappa \triangleleft_{\mathbf{m}} A$ and their rules:

$$\begin{array}{c} \hline \mathbf{ff} \triangleleft \mathbf{Bool} \end{array} \quad \begin{array}{c} \hline \mathbf{tt} \triangleleft \mathbf{Bool} \end{array} \quad \frac{\kappa \triangleleft_{\mathbf{m}} A \quad \beta \triangleleft B}{\kappa \multimap \alpha \triangleleft A \rightarrow B} \quad \frac{\forall i \in \{1, \dots, n\} \quad \alpha_i \triangleleft A}{[\alpha_1, \dots, \alpha_n] \triangleleft_{\mathbf{m}} A}$$

Resource contexts: sequences Θ of bindings of the form

$$\Theta, \Sigma, \dots ::= (x_i : [a_{i,1}, \dots, a_{i,n_i}] \triangleleft A_i)_{1 \leq i \leq n} \quad (n \in \mathbb{N})$$

Addition of resource contexts: given

$$\Theta = (x_i : \kappa_i \triangleleft A_i)_{1 \leq i \leq n} \quad \Sigma = (x_i : \lambda_i \triangleleft A_i)_{1 \leq i \leq n}$$

we put

$$\Theta \vec{\oplus} \Sigma = (x_i : (\kappa_i ++ \lambda_i) \triangleleft A_i)_{1 \leq i \leq n}$$

where $\kappa_i ++ \lambda_i$ stands for the **concatenation of lists**.

Intersection type judgements $\Theta \vdash t : \alpha \triangleleft A$ and their rules:

$$\text{(IT-Var)} \quad \frac{\alpha \triangleleft A_i}{(x_1 : [] \triangleleft A_1, \dots, x_i : [\alpha] \triangleleft A_i, \dots, x_n : [] \triangleleft A_n) \vdash x_i : \alpha \triangleleft A_i}$$

$$\text{(IT-App)} \quad \frac{\Theta \vdash t : \kappa \multimap \beta \triangleleft A \rightarrow B \quad \Theta' \vdash_{\text{m}} u : \kappa}{\Theta \vec{\oplus} \Theta' \vdash t \ u : \beta \triangleleft B}$$

$$\text{(IT-Lam)} \quad \frac{(\Theta, x : \kappa \triangleleft A) \vdash t : \beta \triangleleft B}{\Theta \vdash \lambda x. t : \kappa \multimap \beta \triangleleft A \rightarrow B}$$

$$\text{(IT-Int)} \quad \frac{\Theta_i \vdash t : \alpha_i \triangleleft A \quad \text{for } i \in \{1, \dots, n\}}{\Theta_1 \vec{\oplus} \dots \vec{\oplus} \Theta_n \vdash_{\text{m}} t : [\alpha_1, \dots, \alpha_n] \triangleleft A}$$

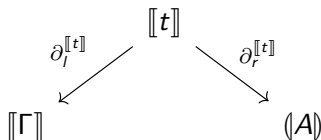
Since we are considering **Bool** and **if**'s:

$$\text{(IT-If-True)} \quad \frac{\Theta \vdash t : \mathbf{tt} \triangleleft \mathbf{Bool} \quad \Theta_{\text{then}} \vdash u : \alpha \triangleleft A}{\Theta \oplus \vec{\Theta}_{\text{then}} \vdash_{\text{m}} \text{if } t \text{ then } u \text{ else } v : \alpha \triangleleft A}$$

$$\text{(IT-If-Else)} \quad \frac{\Theta \vdash t : \mathbf{tt} \triangleleft \mathbf{Bool} \quad \Theta_{\text{else}} \vdash v : \alpha \triangleleft A}{\Theta \oplus \vec{\Theta}_{\text{else}} \vdash_{\text{m}} \text{if } t \text{ then } u \text{ else } v : \alpha \triangleleft A}$$

Theorem

Given a derivation $\Gamma \vdash t : A$ and its bagged interpretation



we have a bijection

$$\text{Ob}(\llbracket t \rrbracket) \cong \{ p \mid p \text{ derivation of } \Theta \vdash t : \alpha : A \text{ for } \Theta \triangleleft \Gamma \}.$$

- Contrarily to **Rel** and **Esp** [olimpieri2021intersection], no quotient is required here for the bijection.

A known broken system

That system is the direct system we obtain when dropping commutativity in the ITS of **Rel**.

It is known for **not** satisfying subject reduction!

A known broken system

In the context $f: \mathbf{Bool} \rightarrow \mathbf{Bool} \rightarrow \mathbf{Bool}, x: \mathbf{Bool}$ consider

$$t_1 = (\lambda y. \lambda z. f \ z \ y) \ x \ x \qquad t_2 = f \ x \ x$$

A known broken system

In the context $f: \mathbf{Bool} \rightarrow \mathbf{Bool} \rightarrow \mathbf{Bool}, x: \mathbf{Bool}$ consider

$$t_1 = (\lambda y. \lambda z. f \ z \ y) \ x \ x \quad t_2 = f \ x \ x$$

Given the resource contexts

$$\Theta_{\mathbf{ff}, \mathbf{tt}} = (f: [[\mathbf{ff}]] \multimap [\mathbf{tt}] \multimap \mathbf{tt}] \triangleleft \mathbf{Bool} \rightarrow \mathbf{Bool}, x: [\mathbf{ff}; \mathbf{tt}] \triangleleft \mathbf{Bool})$$

$$\Theta_{\mathbf{tt}, \mathbf{ff}} = (f: [[\mathbf{ff}]] \multimap [\mathbf{tt}] \multimap \mathbf{tt}) \triangleleft \mathbf{Bool} \rightarrow \mathbf{Bool}, x: [\mathbf{tt}; \mathbf{ff}] \triangleleft \mathbf{Bool})$$

we have

$$\Theta_{\mathbf{ff}, \mathbf{tt}} \not\vdash t_1: \mathbf{tt} \triangleleft \mathbf{Bool}$$

$$\Theta_{\mathbf{ff}, \mathbf{tt}} \vdash t_2: \mathbf{tt} \triangleleft \mathbf{Bool}$$

$$\Theta_{\mathbf{tt}, \mathbf{ff}} \vdash t_1: \mathbf{tt} \triangleleft \mathbf{Bool}$$

$$\Theta_{\mathbf{tt}, \mathbf{ff}} \not\vdash t_2: \mathbf{tt} \triangleleft \mathbf{Bool}$$

while $t_1 \rightarrow^* t_2 \rightsquigarrow$ **subject reduction not satisfied**

A known broken system

But in **Thin**, the reduction $t_1 \rightarrow^* t_2$ is interpreted as a reindexing

$$\llbracket t_1 \rightarrow^* t_2 \rrbracket : \llbracket t_1 \rrbracket \rightarrow \llbracket t_2 \rrbracket$$

so that subject reduction is weakly recovered:

$$\Theta_{\mathbf{tt}, \mathbf{ff}} \vdash t_1 : \mathbf{tt} \triangleleft \mathbf{Bool} \quad \cong \quad \Theta_{\mathbf{ff}, \mathbf{tt}} \vdash t_2 : \mathbf{tt} \triangleleft \mathbf{Bool}$$

(The above is informal!)

ITS for morphisms

Let's remember that thin spans are spans of groupoids, **with morphisms**.

Can we describe the ones in the interpretation of λ -terms?

$$\begin{array}{ccc} & \llbracket t \rrbracket & \\ \partial_l^{\llbracket t \rrbracket} \swarrow & & \searrow \partial_r^{\llbracket t \rrbracket} \\ \llbracket \Gamma \rrbracket & & \llbracket A \rrbracket \end{array} \in \mathbf{Gpd}$$

ITS for morphisms

Refinement ~~types~~ **value morphisms** and intersection ~~types~~ **value morphisms**:

$$\alpha, \beta, \dots ::= \mathbf{tt} \mid \mathbf{ff} \mid \kappa \multimap \alpha$$

$$\kappa, \lambda, \dots ::= [\alpha_1, \dots, \alpha_n] \quad (n \in \mathbb{N})$$

$$\phi, \psi, \dots ::= \mathbf{id}_{\mathbf{tt}} \mid \mathbf{id}_{\mathbf{ff}} \mid \theta \multimap \phi$$

$$\theta, \zeta, \dots ::= (\pi, [\phi_1, \dots, \phi_n]) \quad (n \in \mathbb{N}, \pi \in \mathcal{S}_n)$$

ITS for morphisms

Resource morphism contexts:

$$\Xi, \Xi', \dots \quad ::= \quad (x_i : \theta_i :: \kappa_i \Rightarrow \kappa'_i \triangleleft A_i)_{1 \leq i \leq n}$$

ITS for morphisms

Intersection type morphism judgements

$$\Xi \vdash t : \phi :: \alpha \Rightarrow \alpha' \triangleleft A \quad \text{and} \quad \Xi \vdash_{\text{m}} t : \theta :: \kappa \Rightarrow \kappa' \triangleleft A$$

and their rules:

$$\text{(ITM-Var)} \quad \frac{\phi :: \alpha \Rightarrow \alpha' \triangleleft A_i}{(\dots, x_i : (\text{id}_{\{1\}}, [\phi]) :: [\alpha] \Rightarrow [\alpha'] \triangleleft A_i, \dots) \vdash x_i : \phi :: \alpha \Rightarrow \alpha' \triangleleft A_i}$$

$$\text{(ITM-App)} \quad \frac{\Xi \vdash t : (\theta \multimap \phi) :: (\kappa \multimap \beta) \Rightarrow (\kappa' \multimap \beta') \triangleleft A \rightarrow B \quad \Xi' \vdash_{\text{m}} u : \theta :: \kappa \Rightarrow \kappa'}{\Xi \oplus \Xi' \vdash t u : \phi :: \beta \Rightarrow \beta' \triangleleft B}$$

$$\text{(ITM-Lam)} \quad \frac{(\Xi, x : \theta :: \kappa \Rightarrow \kappa' \triangleleft A) \vdash t : \phi :: \beta \Rightarrow \beta' \triangleleft B}{\Xi \vdash \lambda x. t : (\theta \multimap \phi) :: (\kappa \multimap \beta) \Rightarrow (\kappa' \multimap \beta') \triangleleft A \rightarrow B}$$

$$\text{(ITM-Int)} \quad \frac{n \in \mathbb{N} \quad \sigma \in \mathcal{S}_n \quad \forall i \in \{1, \dots, n\}, \Xi_i \vdash t : \phi_i :: \alpha_i \Rightarrow \alpha'_i \triangleleft A}{(\sigma \otimes (\Xi_i)_{i \in [n]}) \vdash_{\text{m}} t : (\sigma, [\phi_1, \dots, \phi_n]) :: [\alpha_i]_{1 \leq i \leq n} \Rightarrow [\alpha'_{\sigma^{-1}(i)}]_{1 \leq i \leq n} \triangleleft A}$$

ITS for morphisms

Given a derivation $\Gamma \vdash t : A$, we get a groupoid $\llbracket t \rrbracket^{\mathbf{IT}}$:

- ▶ objects: derivations $\Theta \vdash t : \alpha \triangleleft A$
- ▶ morphisms: derivations $\Xi \vdash t : \phi :: \alpha \Rightarrow \alpha' \triangleleft A$
- ▶ composition of derivations as composition

Theorem (C., F.)

*The interpretation in **Thin** is described by the presented ITS, i.e.,*

$$\llbracket t \rrbracket \cong \llbracket t \rrbracket^{\mathbf{IT}}.$$

Conclusion

Thin spans

- ▶ a **quantitative semantic model** based on spans of groupoids
- ▶ provide a **proof-relevant extension** of the relational model **Rel**
- ▶ **syntactic description** given by an intersection type system

Related works

Other quantitative models:

- ▶ **Generalized species of structures**
fiore2008cartesian
- ▶ **Template games**
mellies2019template

Syntactic descriptions through intersection type systems:

- ▶ **olimpieri2021intersection**
- ▶ **tsukada2017generalised**

The end

Any questions?

Seely equivalence

Recall: a common approach for exhibiting a categorical model of **LL** is to find a Seely isomorphism

$$\text{see}_{A,B}: !A \otimes !B \rightarrow !(A \& B).$$

Seely equivalence

In **Thin**,

$$\mathcal{A} \otimes \mathcal{B} \triangleq (A \times B, \dots) \quad \text{and} \quad \mathcal{A} \& \mathcal{B} \triangleq (A \sqcup B, \dots).$$

We have the 2-categorical analogue of a Seely isomorphism, already in **Gpd**:

Proposition

Given $A, B \in \mathbf{Gpd}$, there is an adjoint equivalence of groupoids

$$\begin{array}{ccc} & \xrightarrow{\text{see}_{A,B}} & \\ \text{List}^*(A) \times \text{List}^*(B) & \perp & \text{List}^*(A \sqcup B) \\ & \xleftarrow{\overline{\text{see}}_{A,B}} & \end{array}$$

Idea: given $a = (a_i)_{i \in I}$ and $b = (b_j)_{j \in J}$, one can merge a and b as $c = (c_k)_{k \in K}$ with $K \cong I \sqcup J$.

The Seely 2-cell

Recall: the Seely isomorphism

$$\text{see}_{A,B}: !A \otimes !B \rightarrow !(A \& B)$$

is supposed to verify the equality

$$\begin{array}{ccc}
 !A \otimes !B & \xrightarrow{\text{see}_{A,B}} & !(A \& B) \\
 \downarrow \delta_A \otimes \delta_B & & \downarrow \delta_{A \& B} \\
 & = & !!(A \& B) \\
 & & \downarrow \langle !I, !r \rangle \\
 !!A \otimes !!B & \xrightarrow{\text{see}_{!A, !B}} & !(!A \& !B)
 \end{array}$$

The Seely 2-cell

The Seely equality appears here as a non-trivial 2-cell in **Gpd**:

$$\begin{array}{ccc}
 !!A \times !!B & \xlongequal{\quad} & !!A \times !!B \\
 \downarrow \text{see}_{!A,!B} & & \downarrow \mu_A \times \mu_B \\
 !(!A \sqcup !B) & & !A \times !B \\
 \downarrow \text{!}[\text{!}(\bar{I}),\text{!}(\bar{r})] & \xRightarrow{\text{See}_{A,B}} & \downarrow \text{see}_{A,B} \\
 !!(A \sqcup B) & & \\
 \downarrow \mu_{A \sqcup B} & & \downarrow \\
 !(A \sqcup B) & \xlongequal{\quad} & !(A \sqcup B)
 \end{array}$$

Cartesian structure

Definition

A bicategory \mathcal{C} is **cartesian** when, for every objects Y, Z , there exist

an object $Y \& Z \in \mathcal{C}$ and morphisms $l: Y \& Z \rightarrow Y$ and $r: Y \& Z \rightarrow Z$

such that, for every X , there is an adjoint equivalence of categories

$$\begin{array}{ccc} & (l \odot (-), r \odot (-)) & \\ & \curvearrowright & \\ \mathcal{C}(X, Y \& Z) & \perp & \mathcal{C}(X, Y) \times \mathcal{C}(X, Z) \\ & \curvearrowleft & \\ & \langle -, - \rangle & \end{array}$$

(+ there exists a terminal object expressed as an adjoint equivalence too).

Cartesian structure

Theorem

The bicategory $\mathbf{Thin}_!$ is cartesian.

Cartesian structure

Theorem

The bicategory $\mathbf{Thin}_!$ is cartesian.

Given two thin groupoids \mathcal{A} and \mathcal{B} , we take $\mathcal{A} \& \mathcal{B} \triangleq (A \sqcup B, \dots)$ and

$$l = \begin{array}{c} A \\ \swarrow \bar{l} \quad \searrow \text{id}_A \\ A \sqcup B \quad A \\ \swarrow \eta_{A \sqcup B} \\ \mathbf{List}^*(A \sqcup B) \end{array}$$

and

$$r = \begin{array}{c} B \\ \swarrow \bar{r} \quad \searrow \text{id}_B \\ A \sqcup B \quad B \\ \swarrow \eta_{A \sqcup B} \\ \mathbf{List}^*(A \sqcup B) \end{array}$$

for $l: \mathcal{A} \& \mathcal{B} \rightarrow \mathcal{A}$ and $r: \mathcal{A} \& \mathcal{B} \rightarrow \mathcal{B}$ in $\mathbf{Thin}_!$.

Closure

A cartesian bicategory \mathcal{C} is **closed** when, for every object Y, Z , there exist
an object $Y \Rightarrow Z \in \mathcal{C}$ and a morphism $\text{ev}_{Y,Z}: (Y \Rightarrow Z) \& Y \rightarrow Z$
such that, for every $X \in \mathcal{C}$, there is an adjoint equivalence

$$\begin{array}{ccc} & \text{ev}_{Y,Z} \odot (-\& Y) & \\ & \curvearrowright & \\ \mathcal{C}(X, Y \Rightarrow Z) & \perp & \mathcal{C}(X \& Y, Z) \\ & \curvearrowleft & \\ & (-)^\dagger & \end{array}$$

Theorem

The cartesian bicategory $\mathbf{Thin}_!$ is closed.

Closure

A cartesian bicategory \mathcal{C} is **closed** when, for every object Y, Z , there exist
an object $Y \Rightarrow Z \in \mathcal{C}$ and a morphism $\text{ev}_{Y,Z}: (Y \Rightarrow Z) \& Y \rightarrow Z$
such that, for every $X \in \mathcal{C}$, there is an adjoint equivalence

$$\begin{array}{ccc} & \text{ev}_{Y,Z} \odot (-\& Y) & \\ & \curvearrowright & \\ \mathcal{C}(X, Y \Rightarrow Z) & \perp & \mathcal{C}(X \& Y, Z) \\ & \curvearrowleft & \\ & (-)^\dagger & \end{array}$$

Theorem

*The cartesian bicategory **Thin**_! is closed.*

The closed structure for **Thin**_!

Given thin groupoids \mathcal{B}, \mathcal{C} , we take $\mathcal{B} \Rightarrow \mathcal{C} \triangleq (!B \times C, \dots)$ and

$$\text{ev}_{\mathcal{B}, \mathcal{C}}: (\mathcal{B} \Rightarrow \mathcal{C}) \& \mathcal{B} \rightarrow \mathcal{C} =$$

$$\begin{array}{c} !B \times C \\ \swarrow \langle l, r, l \rangle \\ !B \times C \times !B \\ \swarrow \eta_{!B \times C \times !B} \\ !(B \times C) \times !B \\ \swarrow \text{see}_{!B \times C, B} \\ !((B \times C) \sqcup B) \end{array} \quad \searrow r \quad C$$

(writing directly ! for **List**^{*}).

Probabilistic extension

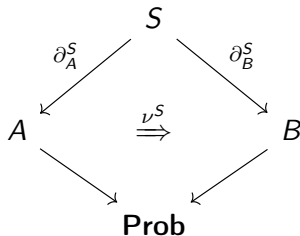
Write **Prob** for the category

- ▶ with one object \bullet
- ▶ with $[0, 1] \subset \mathbb{R}$ as the set of arrows
- ▶ with multiplication as composition

$$\mathbf{Prob} = \begin{array}{c} [0,1] \\ \downarrow \\ \bullet \end{array}$$

Probabilistic extension

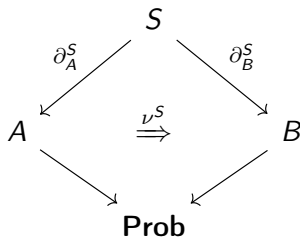
A **span enriched in probabilities** is a span $S: A \leftrightarrow B$ together with a natural transformation



in **Cat**.

Probabilistic extension

A **span enriched in probabilities** is a span $S: A \leftrightarrow B$ together with a natural transformation



in **Cat**.

Concretely, it is the data of $\nu^S(s) \in [0, 1]$ for every $s \in S$ respecting the symmetries in S .

Probabilistic extension

Composing two spans enriched in probabilities S and T :

$$\nu^{T \odot S} =$$

The diagram illustrates the composition of two spans S and T into a single span $T \odot S$. The nodes are arranged in a diamond shape: $T \odot S$ at the top, S and T in the middle, A , B , and C below them, and Prob at the bottom. Dashed arrows point from $T \odot S$ to S and T , with a \vee symbol between them. Solid arrows labeled ∂_A^S and ∂_B^S point from S to A and B respectively. Solid arrows labeled ∂_A^T and ∂_B^T point from T to B and C respectively. A solid arrow labeled ν^S points from A to B , and a solid arrow labeled ν^T points from B to C . At the bottom, two Prob nodes are connected by a double arrow. All paths from the top to the bottom Prob node are shown to be equal with '=' symbols.

Probabilistic extension

By computing the pasting diagram we have

$$\nu^{T \odot S}((s, t)) = \nu^S(s) \cdot \nu^T(t)$$

so that the composition of spans adequately multiplies the probabilities of the witnesses.

Other effectful extensions

More generally, we can hope for

Theorem (In preparation...)

Given an SMCC \mathcal{C} , there is a cartesian closed bicategory $\mathbf{Thin}_{!,\mathcal{C}}$ of spans enriched in \mathcal{C} .

We recover in our setting the weighting of gen. species of structures by SMCC:

► **tsukada2018species**

The end

Any questions?

Whiteboard