

Modelling distributed specifications with simplicial sets

Nasos Evangelou-Oost

PhD project

Advisors: Prof Ian Hayes, Dr Larissa Meinicke

6 June 2022



1 Introduction & preliminaries

2 Simplicial systems

3 Distributed simplicial systems

4 Future work & conclusion

Main ideas in this talk:

- 1 simplicial sets are a nice model for abstract systems as sets of possible observations closed under “stuttering” and “mumbling”
- 2 simplicial subsets of a system form a “refinement algebra”—a lattice where the ordering represents implementability of specifications
- 3 sequential consistency in distributed specifications can be defined with the “sheaf condition”

Main inspirations:

- Contextuality in relational databases (Abramsky 2013)
- Higher-dimensional automata (Glabbeek 2006)
- Sheaf systems (Goguen 1992)
- Refinement algebras (Hoare et al. 2009; Hayes et al. 2016)
- Rely-guarantee logic and concurrent separation logic (Jones 1983; O’Hearn 2007)
- Presheaf models for concurrency (Joyal, Nielsen and Winskel 1996)
- Causality and sequential consistency in distributed systems (Lamport 1979)

Definition (preorder)

A **preorder** \mathcal{P} consists of a set \mathcal{P} and a relation $(\leq) \subseteq \mathcal{P} \times \mathcal{P}$ satisfying $\forall a, b, c \in \mathcal{P}$:

- reflexivity: $a \leq a$
- transitivity: $a \leq b$ and $b \leq c$ implies $a \leq c$

Basic operations on preorders:

- coproduct: $\mathcal{P} + \mathcal{Q} =$ disjoint union of \mathcal{P} and \mathcal{Q}

$$a \leq_{\mathcal{P} + \mathcal{Q}} b \iff \begin{cases} a \leq_{\mathcal{P}} b, & a, b \in \mathcal{P} \\ a \leq_{\mathcal{Q}} b, & a, b \in \mathcal{Q} \end{cases}$$

- product: $\mathcal{P} \times \mathcal{Q} =$ cartesian product of \mathcal{P} and \mathcal{Q}

$$(a, a') \leq_{\mathcal{P} \times \mathcal{Q}} (b, b') \iff a \leq_{\mathcal{P}} b \text{ and } a' \leq_{\mathcal{Q}} b'$$

- join: $\mathcal{P} \star \mathcal{Q} =$ disjoint union of \mathcal{P} and \mathcal{Q}

$$a \leq_{\mathcal{P} \star \mathcal{Q}} b \iff \begin{cases} a \leq_{\mathcal{P}} b, & a, b \in \mathcal{P} \\ a \leq_{\mathcal{Q}} b, & a, b \in \mathcal{Q} \\ a \in \mathcal{P} \text{ and } b \in \mathcal{Q} \end{cases}$$

Definition (category)

A **category** \mathcal{C} consists of

- A family of **objects** $\text{ob } \mathcal{C}$
- For each pair of objects $A, B \in \text{ob } \mathcal{C}$ a family of **morphisms** $\mathcal{C}(A, B)$
- For each object $A \in \text{ob } \mathcal{C}$ an **identity** morphism $1_A \in \mathcal{C}(A, A)$
- An associative operation $\circ : \mathcal{C}(B, C) \rightarrow \mathcal{C}(A, B) \rightarrow \mathcal{C}(A, C)$

Examples

- \mathbf{Set} . Objects are sets and morphisms are functions
- A preorder \mathcal{P} . Objects are elements of \mathcal{P} , and $\mathcal{P}(x, y) = \{*\}$ if $x \leq y$ or otherwise $= \emptyset$
- \mathbf{Preord} . Objects are preorders and morphisms are monotone (nondecreasing) functions
- $\Delta \subset \mathbf{Preord}$. Objects are the totally ordered sets $\{0 < \dots < n\}$ for $n \in \mathbb{N}$ and morphisms are monotone functions
- For a category \mathcal{C} , \mathcal{C}^{op} is a category with the same objects as \mathcal{C} but an arrow $f^{\text{op}} : B \rightarrow A$ for every $f : A \rightarrow B$ in \mathcal{C}

Definition (functor)

A **functor** $\mathbf{F} : \mathcal{C} \rightarrow \mathcal{D}$ consists of

- a function

$$\text{ob } \mathcal{C} \rightarrow \text{ob } \mathcal{D}$$

- for each pair of objects $A, B \in \text{ob } \mathcal{C}$, a function

$$\mathcal{C}(A, B) \rightarrow \mathcal{D}(\mathbf{F}A, \mathbf{F}B)$$

- 1 that preserves identities: for all A ,

$$\mathbf{F}1_A = 1_{\mathbf{F}A}$$

- 2 and compositions: for all g, f ,

$$\mathbf{F}(g \circ f) = \mathbf{F}g \circ \mathbf{F}f$$

Examples

- A functor $\mathbf{F} : \mathcal{P} \rightarrow \mathcal{Q}$ between preorders is a monotone function
- **List** : $\text{Set} \rightarrow \text{Set}$ that sends a set X to the set X^* of all lists on X , and other functors from functional programming: **Maybe**, **Tree**, **State**, **Reader**, **Writer**, ...

Natural transformations

Definition (natural transformation)

A **natural transformation** $\alpha : \mathbf{F} \rightarrow \mathbf{G}$ between functors $\mathbf{F} : \mathcal{C} \rightarrow \mathcal{D}$ and $\mathbf{G} : \mathcal{C} \rightarrow \mathcal{D}$ consists of a family of morphisms in \mathcal{D}

$$\{\alpha_c : \mathbf{F}c \rightarrow \mathbf{G}c\}_{c \in \text{ob } \mathcal{C}}$$

such that every square

$$\begin{array}{ccc} \mathbf{F}c & \xrightarrow{\mathbf{F}f} & \mathbf{F}c' \\ \alpha_c \downarrow & & \downarrow \alpha_{c'} \\ \mathbf{G}c & \xrightarrow{\mathbf{G}f} & \mathbf{G}c' \end{array}$$

commutes, i.e. for all $f : c \rightarrow c'$, $\alpha_{c'} \circ \mathbf{F}f = \mathbf{G}f \circ \alpha_c$

Examples

- **concat** : **List** \circ **List** \rightarrow **List** that takes a list of lists and concatenates them. Given the function `toString` : `Int` \rightarrow `String`, naturality says $\text{concat} \circ (\text{map toString}) = (\text{map toString}) \circ \text{concat}$
- Let \mathcal{F} be a category with two objects V and E and two nonidentity morphisms $s, t : E \rightarrow V$. Then a functor $\mathbf{F} : \mathcal{F} \rightarrow \mathcal{Set}$ is a directed graph, and a natural transformation $\alpha : \mathbf{F} \rightarrow \mathbf{G}$ is a graph homomorphism $\mathbf{F} \rightarrow \mathbf{G}$.

Simplicial sets (i)

Definition (simplicial set)

A **simplicial set** is a functor $X : \Delta^{\text{op}} \rightarrow \mathcal{S}et$

Unwinding, a simplicial set X consists of

- ① for each $n \in \mathbb{N}$, a set $X_n := X_n$ of **cells**
- ② for each monotone function $\phi : m \rightarrow n$ between nonempty finite totally ordered sets, a function $\phi^* := X\phi : X_n \rightarrow X_m$, satisfying functor laws (1) and (2):

①

$$(\text{id}_n)^* = \text{id}_{X_n} : X_n \rightarrow X_n$$

②

$$(\phi \circ \theta)^* = \theta^* \circ \phi^*$$

Simplicial sets form a category $\mathcal{s}Set$ with natural transformations as morphisms.

Simplicial sets (ii): degenerate and nondegenerate cells

For X a simplicial set, the functions ϕ^* are generated by two collections of functions

- **face operators** $d_i^n : X_n \rightarrow X_{n-1}$ for $n \geq 1$ and $0 \leq i \leq n$
- **degeneracy operators** $s_i^n : X_n \rightarrow X_{n+1}$ for $n \geq 0$ and $0 \leq i \leq n$

$$\begin{array}{ccccc} X_0 & \begin{array}{c} \xleftarrow{d_1} \text{---} \\ \xrightarrow{s_0} \text{---} \\ \xleftarrow{d_0} \text{---} \end{array} & X_1 & \begin{array}{c} \xleftarrow{d_2} \text{---} \\ \xrightarrow{s_1} \text{---} \\ \xleftarrow{d_1} \text{---} \\ \xrightarrow{s_0} \text{---} \\ \xleftarrow{d_0} \text{---} \end{array} & X_2 & \dots \end{array}$$

that satisfy the **simplicial relations**

- $d_i d_j = d_{j-1} d_i, \quad i < j$
- $s_i s_j = s_{j+1} s_i, \quad i \leq j$
- $d_i s_j = \begin{cases} s_{j-1} d_i, & i < j \\ d_i s_j = \text{id}, & i = j, j+1 \\ d_i s_j = s_j d_{i-1}, & i > j+1 \end{cases}$

Definition (degenerate/nondegenerate cell)

For a simplicial set X , cells $s_i(x) \in X_n$ in the image of a degeneracy map s_i are called **degenerate**. Otherwise, they are **nondegenerate**.

Simplicial sets (iii): standard simplices

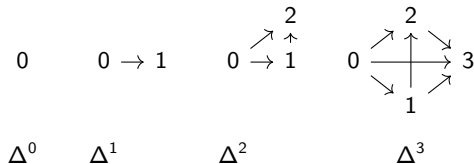
Example

The standard n -simplex Δ^n is the simplicial set whose n -cells are

$$\Delta_k^n := \Delta(k, n) = \text{set of monotone functions } \theta : \{0 < \dots < k\} \rightarrow \{0 < \dots < n\}$$

and if $\phi : m \rightarrow k$ then $\phi^* := \Delta^n \phi$ sends $\theta : k \rightarrow n$ to $\theta \circ \phi : m \rightarrow k$,

The nondegenerate cells in Δ^n correspond to injective functions.



Proposition (Yoneda lemma)

The n -cells of a simplicial set X are in natural bijection with simplicial morphisms from Δ^n to X ,

$$X_n \cong \mathcal{S}et(\Delta^n, X) = \text{set of natural transformations } f : \Delta^n \rightarrow X$$

Simplicial sets (iv): basic operations

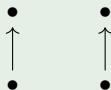
The same basic operations on preorders generalise to simplicial sets:

- coproduct: $(X + Y)_n = X_n \amalg Y_n$
- product: $(X \times Y)_n = X_n \times Y_n$
- join: $(X \star Y)_n = X_n \amalg \left(\amalg_{p+q=n} X_p \times Y_q \right) \amalg Y_n$

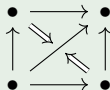
All three operations define binary functors

$$\mathcal{sSet} \times \mathcal{sSet} \rightarrow \mathcal{sSet}$$

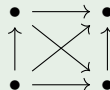
Example



$$\Delta^1 + \Delta^1$$



$$\Delta^1 \times \Delta^1$$



$$\Delta^1 \star \Delta^1$$

Interesting facts:

- $\Delta^p \star \Delta^q \cong \Delta^{p+1+q}$
- nondegenerate cells in $\Delta^p \times \Delta^q$ correspond to (p, q) -shuffles

Definition (nerve of a preorder)

The nerve is a functor $\mathbf{N} : \mathcal{Preord} \rightarrow \mathcal{sSet}$. For \mathcal{P} a preorder, define a simplicial set $\mathbf{N}\mathcal{P}$ whose n -cells are

$$(\mathbf{N}\mathcal{P})_n := \mathcal{Preord}(n, \mathcal{P})$$

(the set of chains of length n in \mathcal{P}).

For $\phi : m \rightarrow n$ we set

$$\phi^* := \mathbf{N}\phi : (\mathbf{N}\mathcal{P})_n \rightarrow (\mathbf{N}\mathcal{P})_m$$

that sends $\theta : n \rightarrow \mathcal{P}$ to $\theta \circ \phi : m \rightarrow \mathcal{P}$.

Properties of the nerve:

- preserves limits, e.g. products: $\mathbf{N}(\mathcal{P} \times \mathcal{Q}) = \mathbf{N}\mathcal{P} \times \mathbf{N}\mathcal{Q}$
- preserves coproducts (but not all colimits): $\mathbf{N}(\mathcal{P} + \mathcal{Q}) = \mathbf{N}\mathcal{P} + \mathbf{N}\mathcal{Q}$
- preserves joins: $\mathbf{N}(\mathcal{P} \star \mathcal{Q}) = \mathbf{N}\mathcal{P} \star \mathbf{N}\mathcal{Q}$
- is fully faithful: the functions $\mathcal{Preord}(\mathcal{P}, \mathcal{Q}) \rightarrow \mathcal{sSet}(\mathbf{N}\mathcal{P}, \mathbf{N}\mathcal{Q})$ are bijective

1 Introduction & preliminaries

2 **Simplicial systems**

3 Distributed simplicial systems

4 Future work & conclusion

Let \mathcal{S} be a set of *states* (or *events*) preordered by “reachability” (or “causality”). Let S be its nerve,

$$S := \mathbf{N}\mathcal{S}$$

For $n \in \mathbb{N}$, the set S_n consists of all *traces* of $n + 1$ reachable states in \mathcal{S} ,

$$S_n = \{(x_0, \dots, x_n) \mid x_0 \leq \dots \leq x_n\}$$

Degeneracy operators are *stutterings*, e.g. for $x := (x_0, \dots, x_n) \in S_n$,

$$s_i(x) = (x_0, \dots, x_i, x_i, \dots, x_n)$$

the first face operator is *prefix* and the last is *postfix*:

$$d_0(x) = (x_0, \dots, x_{n-1}), \quad d_n(x) = (x_1, \dots, x_n)$$

the other face operators are *mumblings*

$$d_i(x) = (x_0, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

System (i)

Definition (system)

A **system** is a simplicial morphism (a natural transformation) $a : A \rightarrow S$.

Systems form a category $\mathcal{S}ys$ where a morphism $f : a \rightarrow b$ is a simplicial morphism $f : A \rightarrow B$ such that $b \circ f = a$, i.e. the triangle below commutes

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ & \searrow a & \downarrow b \\ & & S \end{array}$$

Proposition

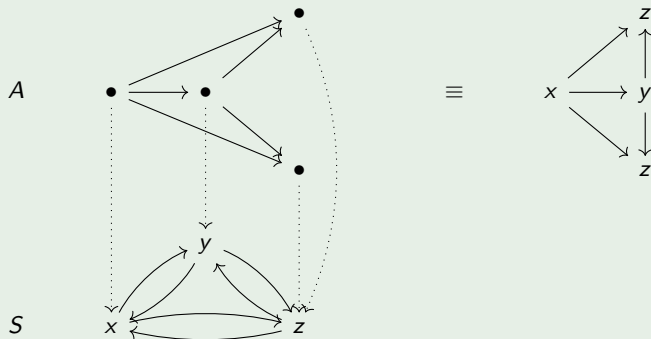
A system $a : A \rightarrow S$ is fully determined by $a_0 : A_0 \rightarrow S_0$.

The category $\mathcal{S}ys$ has many good properties due to it being a **topos**

- it is complete (has all limits) and cocomplete (all colimits)
- it is cartesian closed, i.e. there are exponential objects $a \Rightarrow b$
- every object has a lattice of subobjects
- it has an internal higher-order logic/dependant type theory

Example

An example of a system $a : A \rightarrow S$. On the left, dotted arrows indicate the map $a_0 : A_0 \rightarrow S_0$ (and a_n for $n > 0$ is forced). On the right, the same system with the convention that cells in A are labelled by their image under a and S is implicit.



In these pictures only nondegenerate cells up to dimension 1 (vertices and edges) are shown, but A has nondegenerate cells up to dimension 2, whereas S has nondegenerate cells in all dimensions.

Definition (synchronous product)

For $a, b \in \mathcal{S}ys$ we rename the product $a \times b$ to the **synchronous parallel of a and b** , written $a \parallel b$.

This is the system defined by

$$(A \parallel B)_n = \{(x, y) \mid x \in A_n, y \in B_n, a_n(x) = b_n(y)\}$$

where the arrow $a \parallel b$ whose n -cells are

$$(a \parallel b)_n(x, y) = a_n(x), \quad x \in A_n, y \in B_n$$

In other words, it is the composite arrow in the pullback diagram

$$\begin{array}{ccc} A \parallel B & \dashrightarrow & B \\ \downarrow & \searrow a \parallel b & \downarrow b \\ A & \xrightarrow{a} & S \end{array}$$

The synchronous parallel is a **symmetric monoidal operator**, i.e. it is

- commutative up to isomorphism
- associative
- has as unit the system $\text{id} : S \xrightarrow{\cong} S$

Definition (nondeterministic sum)

For $a, b \in \mathcal{Sys}$ we rename the coproduct $a + b$ to the *nondeterministic sum of A and B* , written $a \sqcup b$.

This is the system $a \sqcup b : A + B \rightarrow S$ whose n -cells are

$$(A + B)_n = A_n \amalg B_n$$

where the arrow $a \sqcup b$ is defined

$$(a \sqcup b)_n(x, 1) = a_n x, \quad x \in A_n$$

$$(a \sqcup b)_n(y, 2) = b_n y, \quad y \in B_n$$

In other words, it is the composite arrow (where ∇ is the codiagonal)

$$A + B \xrightarrow{a+b} S + S \xrightarrow{\nabla} S$$

$\text{-----} \text{-----} \text{-----}$
 $a \sqcup b$

The nondeterministic sum is also symmetric monoidal operator, i.e. it is

- commutative up to isomorphism
- associative
- has as unit the system $! : \emptyset \rightarrow S$

Concatenation product

Assume \mathcal{S} is a **total order** (so any state is reachable from any other). Then there is a monotone function

$$\kappa : \mathcal{S} \star \mathcal{S} \rightarrow \mathcal{S}$$

restricting to the identity on each factor.

Definition (concatenation product)

For $A, B \in \mathcal{Sys}$ we define the **concatenation product of A and B** , written $a \triangleright b$, as the system whose n -cells are

$$(A \triangleright B)_n = \{(x, y) \mid x \in A_i, y \in B_j, i + 1 + j = n\}$$

and the arrow $a \triangleright b$ is defined

$$(a \triangleright b)_n(x, y) = ((a_i x)_{i=0}, \dots, (a_i x)_{i=j}, (b_j y)_{j=0}, \dots, (b_j y)_{j=i})$$

In other words, it is the composite

$$A \star B \xrightarrow{a \star b} S \star S \xrightarrow{\text{N}\kappa} S$$

$a \triangleright b$

The concatenation product with synchronous parallel forms a *duoidal* structure

- concatenation is monoidal with unit $! : \emptyset \rightarrow S$
- there is an *interchange* natural transformation

$$(a \parallel b) \triangleright (c \parallel d) \hookrightarrow (a \triangleright c) \parallel (b \triangleright d)$$

Definition (simplicial subset)

For $X, Y \in \mathcal{S}et$, we say X is a *simplicial subset of Y* iff

1

$$\forall n \in \mathbb{N}. X_n \subseteq Y_n$$

2

$$\forall \phi : m \rightarrow n \in \Delta. (X\phi)_n = (Y\phi)_n|_{X_n}$$

For $A, B \subseteq X$, we can define the *intersection of A and B* by

$$(A \cap B)_n := A_n \cap B_n$$

and the *union of A and B* by

$$(A \cup B)_n := A_n \cup B_n$$

Proposition

The set $\text{sub } X$ of simplicial subsets of X is a *complete heyting algebra*, i.e. a complete bounded lattice with an operator \Rightarrow that is right adjoint to the lattice meet, i.e. there is a Galois connection

$$a \cap b \subseteq c \iff a \subseteq b \Rightarrow c$$

We call the set of simplicial subsets of S *specifications*. The ordering on $\text{sub } S$ is interpreted as *implementability*:

$$a \subseteq b \iff a \text{ implements } b$$

This ordering on specifications induces a preordering on systems, where

$$a \leq b \iff \text{im } a \subseteq \text{im } b$$

Operations on systems induce operations on specifications:

- $\text{im}(a \sqcup b) = \text{im } a \cup \text{im } b$
- $\text{im}(a \parallel b) = \text{im } a \cap \text{im } b$

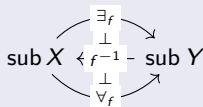
so that disjoint union and synchronous parallel correspond to the lattice join and meet.

Concatenation product is defined on specifications by taking the image, i.e.

$$a \circ b := \text{im}(a \circ b)$$

Proposition

For a simplicial morphism $f : X \rightarrow Y$, there is an adjoint triple of monotone functions $\exists_f \dashv f^{-1} \dashv \forall_f$



1 Introduction & preliminaries

2 Simplicial systems

3 Distributed simplicial systems

4 Future work & conclusion

Let V be a set of variables, and let \mathcal{Var} be a topology on V (a set of subsets of V closed under unions and finite intersections).

For each $v \in V$ let \mathcal{T}_v be a preordered set of values that the variable v may take (e.g. maybe $\mathcal{T}_v = \text{string}$ or bool), the order representing reachability.

For each $U \in \mathcal{Var}$, define

$$\mathbf{T}U := \prod_{v \in U} \mathcal{T}_v$$

For each inclusion $V \subseteq U$ define an action of restriction $\mathbf{T}U \rightarrow \mathbf{T}V$ as restriction of tuples. This defines a functor

$$\mathbf{T} : \mathcal{Var}^{\text{op}} \rightarrow \mathcal{Preord}$$

Precomposing this with the nerve $\mathbf{N} : \mathcal{Preord} \rightarrow \mathcal{sSet}$, we obtain the **state** presheaf

$$S := \mathbf{N} \circ \mathbf{T} : \mathcal{Var}^{\text{op}} \rightarrow \mathcal{sSet}$$

Define a **distributed system** to be an arrow $a : A \rightarrow S$. These form a category \mathcal{DSys} where a morphism from $a : A \rightarrow S$ to $b : B \rightarrow S$ is a commutative triangle

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ a \downarrow & \swarrow b & \\ S & & \end{array}$$

As with simplicial sets, limits and colimits of distributed systems are computed pointwise. In particular,

$$(a \times b)U = aU \times bU, \quad (a + b)U = aU + bU$$

for $U \in \mathcal{Var}$.

Let $a, b \in \mathcal{DSys}$. Nondeterministic sum and synchronous product are defined as with \mathcal{Sys} , as the coproduct

$$a \sqcup b := a + b$$

and cartesian product

$$a \parallel b := a \times b$$

in \mathcal{DSys} . Since the definitions of concatenation product on systems is functorial, a concatenation on \mathcal{DSys} is defined pointwise by

$$(a \triangleright b)U := aU \triangleright bU$$

for $U \in \mathcal{Var}$.

Sequential consistency (i)

Lemma (Eilenberg-Zilber)

For each cell x of a simplicial set X there is a unique nondegenerate cell u_x such that there exists a surjection $\phi \in \Delta$ with $x = \phi^(u_x)$.*

To detect **sequential inconsistency** in specifications, we discard the “absolute timing” information that is encoded in the stutterings of traces and retain only the “relative timing” encoded in nondegenerate traces.

Define a functor

$$\mathbf{nd} : \mathcal{S}\mathcal{S}et \rightarrow \mathcal{S}et$$

$$X \mapsto \text{set of nondegenerate cells of } X$$

$$f \mapsto (x \mapsto u_x)$$

and let \mathbf{nd}_* be postcomposition by \mathbf{nd} , i.e. $\mathbf{nd}_* a := \mathbf{nd} \circ a$.

If $a \in \text{sub } S$ is a distributed specification, $\mathbf{nd}_* a : \mathcal{V}\mathcal{a}\mathcal{r}^{\text{op}} \rightarrow \mathcal{S}et$ is a presheaf (a functor into $\mathcal{S}et$).

Example

Assume $V = \{x, y, z\}$ and $\text{Var} = P\{x, y, z\}$. Let $a \in \text{sub } S$ be a specification such that

$$t_{xy} = \begin{pmatrix} x_0 & x_1 & x_1 \\ y_0 & y_0 & y_1 \end{pmatrix} \in (\text{nd}_* a)\{x, y\}$$

$$t_{yz} = \begin{pmatrix} y_0 & y_1 & y_1 \\ z_0 & z_0 & z_1 \end{pmatrix} \in (\text{nd}_* a)\{y, z\}$$

$$t_{zx} = \begin{pmatrix} z_0 & z_1 & z_1 \\ x_0 & x_0 & x_1 \end{pmatrix} \in (\text{nd}_* a)\{z, x\}$$

This set of traces is *compatible* in that restricting any pair to their common intersection agrees:

$$t_{xy}|_{\{y\}} = (y_0, y_1) = t_{yz}|_{\{y\}}, \quad t_{yz}|_{\{z\}} = (z_0, z_1) = t_{zx}|_{\{z\}}, \quad t_{zx}|_{\{x\}} = (x_0, x_1) = t_{xy}|_{\{x\}}$$

But the traces t_{xy} , t_{yz} , $t_{z,x}$ cannot be *sequentialised*—there cannot exist a $t \in (\text{nd}_* a)V$ with

$$t|_{\{x,y\}} = t_{xy}, \quad t|_{\{y,z\}} = t_{yz}, \quad t|_{\{z,x\}} = t_{zx}$$

Sequential consistency (iii): definition

Definition (sheaf)

Let $\mathbf{F} : \mathcal{T}^{\text{op}} \rightarrow \text{Set}$ be a presheaf on the lattice of open sets \mathcal{T} of a topological space. Then \mathbf{F} is a **sheaf** iff

- for every family of open sets $\{U_i\}_i$
- and every **compatible** family $\{t_i \mid t_i \in U_i\}$, meaning $\forall i, j. t_i|_{U_i \cap U_j} = t_j|_{U_j \cap U_i}$

there **exists** a **unique** $t \in F(\cup_i U_i)$ such that $\forall i. t|_{U_i} = t_i$.

Definition (sequential consistency)

A specification $a \in \text{sub } S$ is **sequentially consistent** iff $\text{nd}_* a$ is a sheaf.

So the specification in the example of the last slide is not sequentially consistent since we found a compatible family of “local” traces $\{t_{xy}, t_{yz}, t_{zx}\}$ that do not glue to a “global” trace t over $\{x, y, z\}$.

Proposition

Sequentially consistent specifications form a Heyting algebra.

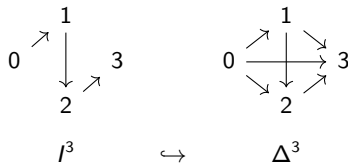
Caution: this Heyting algebra of sequentially consistent specifications is a subset but **not** a sublattice of $\text{sub } S$, since the union of two sequentially consistent specifications is not usually sequentially consistent!

Coherence (i): spines

Definition (spine of a simplex)

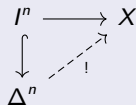
The ***n*-spine** I^n is the simplicial subset of the standard simplex Δ^n consisting of all its adjacent edges.

Note that $I^0 = \Delta^0$ and $I^1 = \Delta^1$, but $I^n \subset \Delta^n$ for $n \geq 1$.



Definition (unique spine extension)

A simplicial set X satisfies ***unique spine extension*** iff every map from the n -spine to X extends uniquely to a map $! : \Delta^n \rightarrow X$.



Definition (coherence)

A system $a : A \rightarrow S$ is **coherent** iff A satisfies unique spine extension.

This is like a “temporal gluing condition” vs. the “spatial gluing condition” of the sheaf condition for sequential consistency.

Example

The system $a : A \rightarrow S$ as displayed

$$x \xrightarrow{xz} y \xrightarrow{yz} z$$

is not coherent as the map $I^2 \rightarrow A$ that sends the first edge to (x, z) and the second to (y, z) has no extension to Δ^2 , since there is no 2-cell (x, y, z) in A .

Proposition

A simplicial set satisfies unique spine extension iff it is in the image of the nerve.

Corollary

Because the nerve is fully faithful, maps between coherent systems are in natural bijection to monotone functions between their underlying preorders.

So a system $a : A \rightarrow S$ is coherent iff it is isomorphic to a system $\mathbf{N}a : \mathbf{N}\mathcal{A} \rightarrow \mathbf{N}\mathcal{S}$ where $a : \mathcal{A} \rightarrow \mathcal{S}$ is a monotone function between preorders.

Caution: the image $\text{im}(\mathbf{N}a)$ of a coherent system $\mathbf{N}a : \mathbf{N}\mathcal{A} \rightarrow \mathbf{N}\mathcal{S}$ is not usually coherent (since the image of a functor need not be a subcategory)!

This means we still need simplicial sets to study specifications of coherent systems.

These definitions and facts about coherence generalise to distributed systems in a pointwise way.





1 Introduction & preliminaries






2 Simplicial systems

3 Distributed simplicial systems

4 Future work & conclusion

- Hoare logic, rely-guarantee logic, concurrent separation logic
- ' ω -simplicial sets' for reasoning about fairness
- Formalisation in a proof assistant

-  Abramsky, Samson (2013). 'Relational Databases and Bell's Theorem'. In: *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*. Ed. by Val Tannen et al. Vol. 8000. Lecture Notes in Computer Science. Springer, pp. 13–35. DOI: 10.1007/978-3-642-41660-6_2. URL: https://doi.org/10.1007/978-3-642-41660-6_2.
-  Glabbeek, Rob J. van (2006). 'On the expressiveness of higher dimensional automata'. In: *Theor. Comput. Sci.* 356.3, pp. 265–290. DOI: 10.1016/j.tcs.2006.02.012. URL: <https://doi.org/10.1016/j.tcs.2006.02.012>.
-  Goguen, Joseph A. (1992). 'Sheaf Semantics for Concurrent Interacting Objects'. In: *Math. Struct. Comput. Sci.* 2.2, pp. 159–191. DOI: 10.1017/S0960129500001420. URL: <https://doi.org/10.1017/S0960129500001420>.
-  Hayes, Ian J. et al. (2016). 'An Algebra of Synchronous Atomic Steps'. In: *FM 2016: Formal Methods - 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings*. Ed. by John S. Fitzgerald et al. Vol. 9995. Lecture Notes in Computer Science, pp. 352–369. DOI: 10.1007/978-3-319-48989-6_22. URL: https://doi.org/10.1007/978-3-319-48989-6_22.

-  Hoare, C. A. R. et al. (2009). 'Concurrent Kleene Algebra'. In: **CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings**. Ed. by Mario Bravetti and Gianluigi Zavattaro. Vol. 5710. Lecture Notes in Computer Science. Springer, pp. 399–414. DOI: 10.1007/978-3-642-04081-8_27. URL: https://doi.org/10.1007/978-3-642-04081-8%5C_27.
-  Jones, Cliff B. (1983). 'Tentative Steps Toward a Development Method for Interfering Programs'. In: **ACM Trans. Program. Lang. Syst.** 5.4, pp. 596–619. DOI: 10.1145/69575.69577. URL: <https://doi.org/10.1145/69575.69577>.
-  Joyal, André, Mogens Nielsen and Glynn Winskel (1996). 'Bisimulation from Open Maps'. In: **Inf. Comput.** 127.2, pp. 164–185. DOI: 10.1006/inco.1996.0057. URL: <https://doi.org/10.1006/inco.1996.0057>.
-  Lamport, Leslie (1979). 'How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs'. In: **IEEE Trans. Computers** 28.9, pp. 690–691. DOI: 10.1109/TC.1979.1675439. URL: <https://doi.org/10.1109/TC.1979.1675439>.
-  O'Hearn, Peter W. (2007). 'Resources, concurrency, and local reasoning'. In: **Theor. Comput. Sci.** 375.1-3, pp. 271–307. DOI: 10.1016/j.tcs.2006.12.035. URL: <https://doi.org/10.1016/j.tcs.2006.12.035>.