Relational theories and their locally presentable categories of models

Initial-algebra semantics for skeletal specifications

Ivan Di Liberti, Tom Hirschowitz, **Robin Jourde**, Alan Schmitt

Third Virtual Workshop on Double Categories – November 10, 2026

Université Savoie Mont Blanc

Motivation

Initial-algebra semantics

Specify programming languages **implicitly**:

- first define **category of models**,
- desired language implicitly defined as **initial object** therein…

up to isomorphism.

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$

$\mathbf{Lex}$ = $\mathbf{l}$eft $\mathbf{ex}$act = categories with finite limits.

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$

$\mathbf{Lex}$ = $\mathbf{l}$eft $\mathbf{ex}$act = categories with finite limits.

Local presentability. $\mathbf{LPCAT}$ = sub-2-category of $\mathbf{CAT}$:

- of categories with good properties (complete, cocomplete, ...)
- closed under useful constructions.

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$

$\mathbf{Lex}$ = $\mathbf{l}$eft $\mathbf{ex}$act = categories with finite limits.

Local presentability. $\mathbf{LPCAT}$ = sub-2-category of $\mathbf{CAT}$:

- of categories with good properties (complete, cocomplete, ...)
- closed under useful constructions.

initial object 😀

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$

$\mathbf{Lex}$ = $\mathbf{l}$eft $\mathbf{ex}$act = categories with finite limits.

Local presentability. $\mathbf{LPCAT}$ = sub-2-category of $\mathbf{CAT}$:

- of categories with good properties (complete, cocomplete, ...)
- closed under useful constructions.

initial object 😀

Not so convenient for programming languages: need **relations**.

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$
This work:	Skel[2] programs	$\longrightarrow$	$\mathbf{VDbTh}$	$\longrightarrow$	$\mathbf{LPCAT}$
			$\mathbb{T}$	$\longmapsto$	$\mathbf{VDBTH}(\mathbb{T}, \mathbb{Rel})$

- $\mathbf{VDbTh}$ = $\mathbf{V}$irtual $\mathbf{Dou}\mathbf{b}$le $\mathbf{Th}$eories = to be defined.
- Keep local presentability.

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod
2. Bodin M, Gardner P, Jensen T, Schmitt A. 2019. Skeletal Semantics and Their Interpretations

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$
This work:	Skel[2] programs	$\longrightarrow$	$\mathbf{VDbTh}$	$\longrightarrow$	$\mathbf{LPCAT}$
			$\mathbb{T}$	$\longmapsto$	$\mathbf{VDBTH}(\mathbb{T}, \mathbb{Rel})$

- $\mathbf{VDbTh}$ = $\mathbf{V}$irtual $\mathbf{Dou}\mathbf{b}$le $\mathbf{Th}$eories = to be defined.
- Keep local presentability.

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod
2. Bodin M, Gardner P, Jensen T, Schmitt A. 2019. Skeletal Semantics and Their Interpretations

Skeletal semantics

Skeletal semantics

- `Skel` = simple, purely functional, strongly typed language, with **non-determinism**
- designed for **specifying** programming languages[2]
  ↳ a `Skel` program = a specification of a language
  Syntax with algebraic datatypes
  Semantics `Skel` "functions" ⇝ evaluation function, reduction *relation*...
- compiled to OCaml, Rocq...
- used to formalise fragments of JavaScript [3] & Python [4]

2. Bodin M, Gardner P, Jensen T, Schmitt A. 2019. Skeletal Semantics and Their Interpretations
3. Khayam A, Noizet V, Schmitt A. 2021. JSkel: Towards a Formalization of JavaScript's Semantics
4. Andrieux M, Schmitt A. 2024. Skeletal Semantics of a Fragment of Python

Example of a programming language – SK calculus

Example (SK calculus[5] a.k.a. combinatory logic)
Simple Turing complete programming language/computational system

"Programs" = **terms**, built from constants S and K and application "$- \cdot -$"

$$m, n, p ::= S \mid K \mid m \cdot n$$

↳ **Notation** (associativity) $m \cdot n \cdot p := (m \cdot n) \cdot p$

"Computation" = a **reduction relation** "$- \to -$", given by deduction rules

$$\frac{}{K \cdot m \cdot n \to m}\, K \qquad \frac{}{S \cdot m \cdot n \cdot p \to (m \cdot p) \cdot (n \cdot p)}\, S$$

$$\frac{m \to m'}{m \cdot n \to m' \cdot n}\, \mathrm{app}_l \qquad \frac{n \to n'}{m \cdot n \to m \cdot n'}\, \mathrm{app}_r$$

5. Schönfinkel M. 1924. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*. 92(3):305–16

Skeletal semantics by example

Example (Running example: SK calculus a.k.a. combinatory logic)

	SK calculus	Skel
Syntax	$m, n, p ::= S \mid K \mid m \cdot n$	`type term = S | K | App (term, term)`
	K, S, $(S \cdot K) \cdot K$, ...	`K`, `S`, `App(App(S,K),K)`, ...
Semantics	reduction relation "$- \to -$"	`val red : term → term`

Notation

`M·N` := `App(M,N)`, with associativity.

E.g. `S·K·K` = `App(App(S,K),K)`

Skeletal semantics by example

Example (Running example: SK calculus, reduction relation)

$$\frac{}{K \cdot m \cdot n \to m}\, K$$

$$\frac{}{S \cdot m \cdot n \cdot p \to (m \cdot p) \cdot (n \cdot p)}\, S$$

$$\frac{m \to m'}{m \cdot n \to m' \cdot n}\, \mathrm{app}_l$$

$$\frac{n \to n'}{m \cdot n \to m \cdot n'}\, \mathrm{app}_r$$

```
val red (t:term) : term =
  branch
    let K·x1·x2 = t in x1
  or
    let S·x1·x2·x3 = t in
    (x1·x3)·(x2·x3)
  or
    let x1·x2 = t in
    let x1 = red x1 in x1·x2
  or
    let x1·x2 = t in
    let x2 = red x2 in x1·x2
  end
```

} **skeleton**

Roadmap

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$
This work:	Skel[2] programs	$\longrightarrow$	$\mathbf{VDbTh}$	$\longrightarrow$	$\mathbf{LPCAT}$
			$\mathbb{T}$	$\longmapsto$	$\mathbf{VDBTH}(\mathbb{T}, \mathbb{Rel})$

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod
2. Bodin M, Gardner P, Jensen T, Schmitt A. 2019. Skeletal Semantics and Their Interpretations

Virtual double theories

Overview: `Skel` vs virtual double theories

Example (Running example: SK calculus)

`Skel` program	⇝	*Freely generated* ***virtual double theory***
`type term`		**object** t
constructors `S`, `K`, `App`		**tight** morphisms $s, k : I \to t$ and $app : t \otimes t \to t$
`val red : term → term`		**loose** morphism $red : t \nrightarrow t$
skeletons `let ... = ... in ...`		loose **paths** $A_0 \overset{f_1}{\nrightarrow} \cdots \overset{f_n}{\nrightarrow} A_n$

`val red (t:term) : term =`
`  branch`
`  skeleton₁`
`  or ... or`
`  skeleton₄`
`  end`
		$\forall 1 \le i \le 4$, a **cell**
		

Virtual double theories

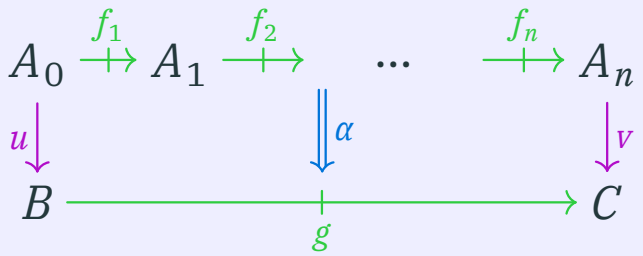
Let's introduce virtual double theories step-by-step:

1. virtual double category
2. locally posetal
3. cartesian
4. equipment

Example (Running example: SK calculus)
...and build up the virtual double theory $\mathbb{T}_{\mathsf{SK}}$

Virtual double categories

Definition (Virtual double category $\mathbb{V}$)
Like a double category, *without loose composition*:



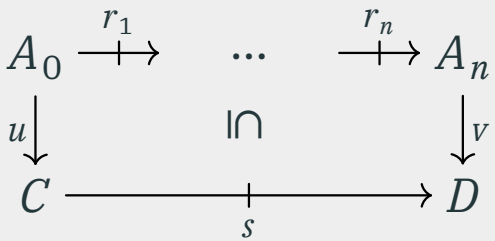
- objects
- tight morphisms ⇝ category $\underline{\mathbb{V}}$, with composition and identities
- loose morphisms ⚠️ no composition, no identity
- n-ary cells ⇝ (n-ary) composition and (unary) identities

Virtual double categories

Example ($\mathbb{Rel}$)

- objects A, B = sets
- tight morphisms $u : A \to B$ = functions
- loose morphisms $r : A \nrightarrow B \in \mathbb{Rel}$ = binary relations $r \subseteq A \times B$
- a n-ary cell $\alpha : r_1; \cdots; r_n \overset{v}{\underset{u}{\Rightarrow}} s$ whenever

$$\forall a, b, (r_1; \cdots; r_n)(a, b) \Rightarrow s(u(a), v(b))$$



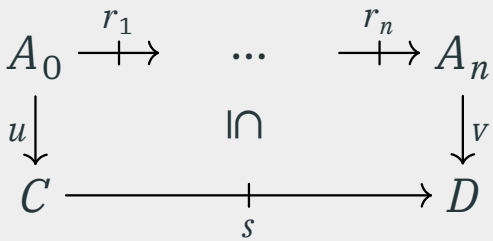
where $(r_1; \cdots; r_n)(a, b) \Leftrightarrow \exists (a_i), a = a_0 \wedge b = a_n \wedge \forall i, r_i(a_{i-1}, a_i)$.

Virtual double categories

Example ($\mathbb{Rel}$)

- objects A, B = sets
- tight morphisms $u : A \to B$ = functions
- loose morphisms $r : A \nrightarrow B \in \mathbb{Rel}$ = binary relations $r \subseteq A \times B$
- a n-ary cell $\alpha : r_1; \cdots; r_n \overset{v}{\underset{u}{\Rightarrow}} s$ whenever

$$\forall a, b, (r_1; \cdots; r_n)(a, b) \Rightarrow s(u(a), v(b))$$



where $(r_1; \cdots; r_n)(a, b) \Leftrightarrow \exists (a_i), a = a_0 \wedge b = a_n \wedge \forall i, r_i(a_{i-1}, a_i)$.

Definition (Locally posetal virtual double category)

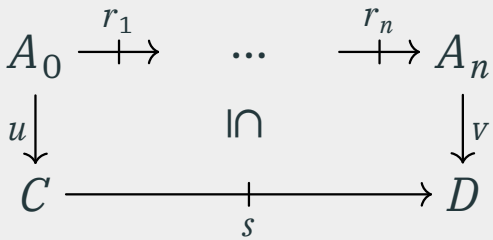
Virtual double category with at most one cell by perimeter

Virtual double categories

Example ($\mathbb{Rel}$)

- objects A, B = sets
- tight morphisms $u : A \to B$ = functions
- loose morphisms $r : A \nrightarrow B \in \mathbb{Rel}$ = binary relations $r \subseteq A \times B$
- a n-ary cell $\alpha : r_1; \cdots; r_n \overset{v}{\underset{u}{\Rightarrow}} s$ whenever

$$\forall a, b, (r_1; \cdots; r_n)(a, b) \Rightarrow s(u(a), v(b))$$



where $(r_1; \cdots; r_n)(a, b) \Leftrightarrow \exists (a_i), a = a_0 \wedge b = a_n \wedge \forall i, r_i(a_{i-1}, a_i)$.

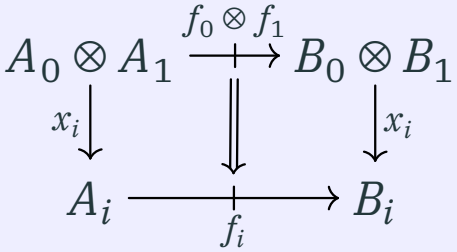
Definition (Locally posetal virtual double category)

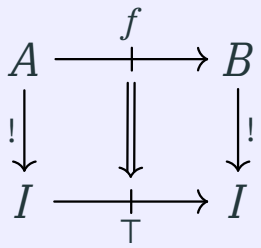
Virtual double category with at most one cell by perimeter $\rightsquigarrow$ all in the following.

Cartesian virtual double categories

Definition (Cartesian virtual double category)
Virtual double category $\mathbb{V}$ with

- tight finite products $A \otimes B$ and I,
- "acting on loose morphisms": e.g.





+ axioms...

where x_i is the projection.

Formally, $\Delta : \mathbb{V} \to \mathbb{V} \times \mathbb{V}$ and $! : \mathbb{V} \to \mathbb{1}$ have right adjoints $\otimes$ and I in $\mathbf{VDbCat}$.

Example ($\mathbb{Rel}$)
Cartesian product of sets $A \times B$ and 1

Cartesian virtual double categories

Example (SK calculus – building the virtual double theory $\mathbb{T}_{\mathsf{SK}}$)

	Freely add	Obtained from the *cartesian* structure
Objects	t	products I, $t \otimes t = t^2$, t^3, ...
Tight morphisms	$s, k : I \to t$ and $app : t \otimes t \to t$	projections $x_i : t^2 \to t$, ...
Loose morphisms	$red : t \nrightarrow t$	action on loose morphisms $red \otimes red : t^2 \nrightarrow t^2$, ...

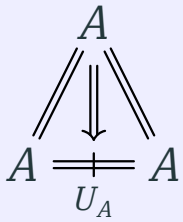
(to be continued in 2 slides...)

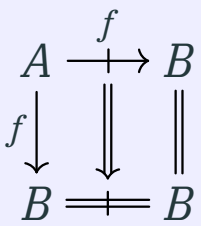
Virtual equipments

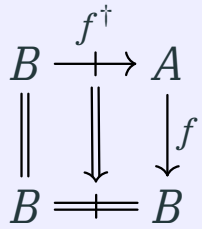
Definition (Virtual equipment)

Virtual double category with

- for any A, **loose unit** $U_A : A \nrightarrow A$,
- for any $f : A \to B$, **companion** $f : A \nrightarrow B$ and **conjoint** $f^\dagger : B \nrightarrow A$.







+ axioms...

Example ($\mathbb{R}\mathrm{el}$)

- loose unit = diagonal relation $U_A(a, a') \Leftrightarrow a = a'$
- companion of a function f = graph, conjoint = converse of the graph

$$f(a, b) \quad \Leftrightarrow \quad b = f(a) \quad \Leftrightarrow \quad f^\dagger(b, a)$$

Virtual equipments

Example (SK calculus – building the virtual double theory $\mathbb{T}_{\mathsf{SK}}$, continued)

	Freely add	Obtained from the *equipment* structure
Objects	t	$\varnothing$
Tight morphisms	$s, k : I \to t$ and $app : t \otimes t \to t$	$\varnothing$
Loose morphisms	$red : t \nrightarrow t$	loose units $U_t : t \nrightarrow t$, ... companions $s, k : I \nrightarrow t$, $app : t \otimes t \nrightarrow t$, $app \circ (k \otimes \mathsf{id}) : t \nrightarrow t$, ... conjoints $s^\dagger, k^\dagger : t \nrightarrow I$, $app^\dagger : t \nrightarrow t \otimes t$, $(app \circ (k \otimes \mathsf{id}))^\dagger : t \nrightarrow t$, ...

From both structures, get: $red \otimes t = red \otimes U_t : t \otimes t \nrightarrow t \otimes t$, ...

↳ Finally freely add some cells...

Building the virtual double theory – adding cells

Recall

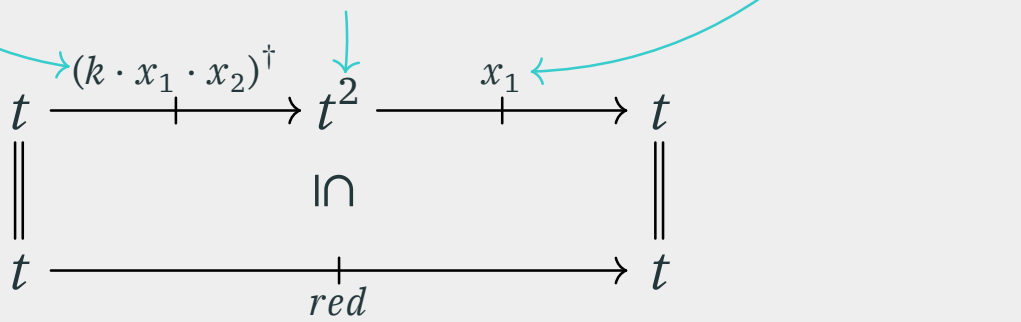
$$\frac{}{K \cdot m \cdot n \to m}\, K$$

```
let K·x1·x2 = e in x1
```

Example (SK calculus – virtual double theory $\mathbb{T}_{\mathsf{SK}}$, continued)

Freely add

conjoint of $k \cdot x_1 \cdot x_2$ product companion of x_1



where $x_1 : t^2 \to t$ and $k \cdot x_1 \cdot x_2 : t^2 \xrightarrow{k \otimes t^2} t^3 \xrightarrow{app \otimes t} t^2 \xrightarrow{app} t.$

Building the virtual double theory – adding cells

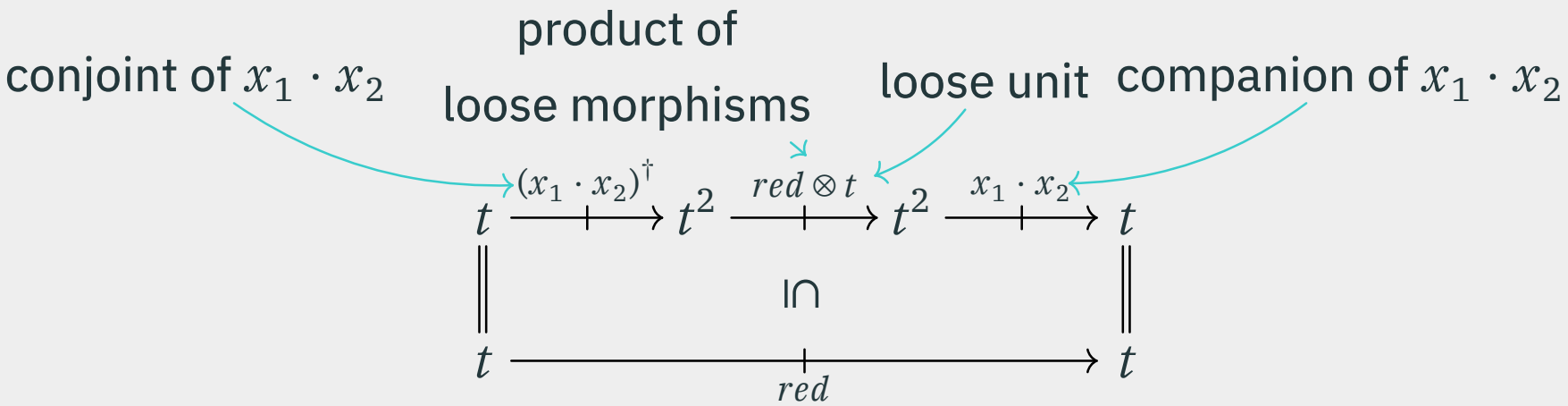
Recall

$$\frac{m \to m'}{m \cdot n \to m' \cdot n}\,\mathrm{app}_l$$

```
let x1·x2 = e in
let x1 = red x1 in x1·x2
```

Example (SK calculus – virtual double theory $\mathbb{T}_{\mathsf{SK}}$, continued)

Freely add

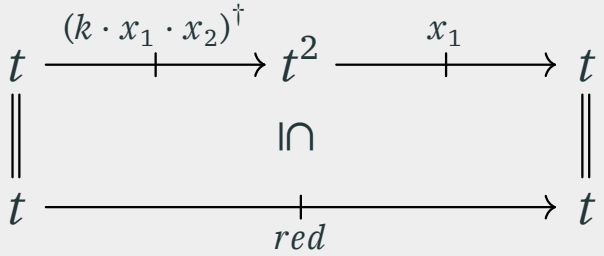


where

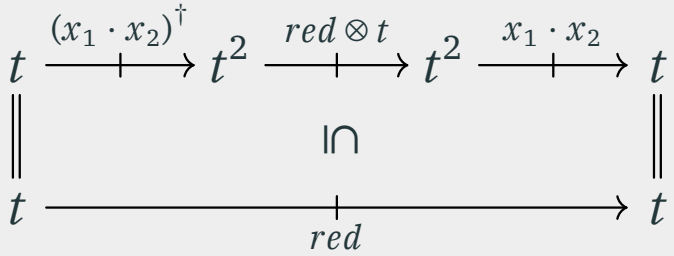
$$x_1 \cdot x_2 : t^2 \xrightarrow{app} t.$$

Building the virtual double theory – adding cells

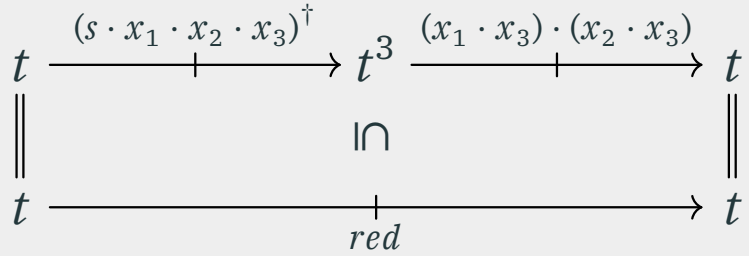
Example (SK calculus – virtual double theory $\mathbb{T}_{\mathsf{SK}}$: summary)



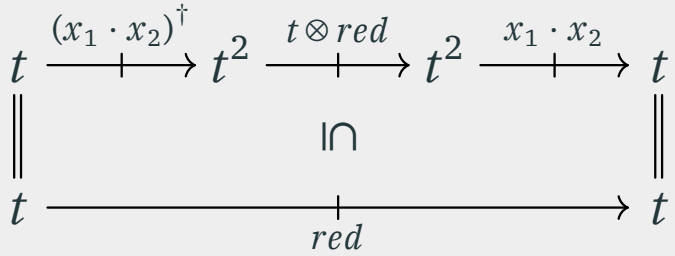

```
let K·x1·x2 = e in x1
```




```
let x1·x2 = e in
let x1 = red x1 in x1·x2
```




```
let S·x1·x2·x3 = e in
(x1·x3)·(x2·x3)
```




```
let x1·x2 = e in
let x2 = red x2 in x1·x2
```

Notice: `let ... = ... in ...` ⇝ sequencing of loose morphisms

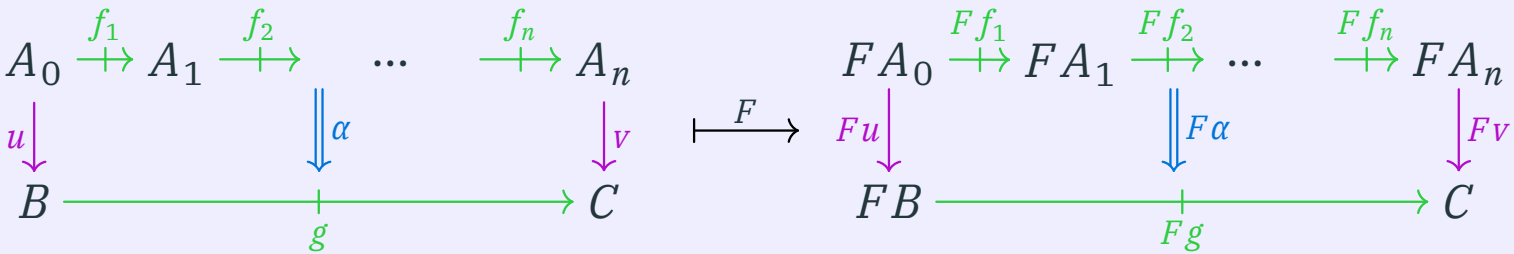
Virtual double theories

Definition (Virtual double theory)
(Locally posetal) cartesian virtual equipment.

Virtual double theories

Definition (Virtual double theory)
(Locally posetal) cartesian virtual equipment.

Definition (Virtual double functor $F : \mathbb{V} \to \mathbb{W}$)



s.t. (tight) functor $\underline{F} : \underline{\mathbb{V}} \to \underline{\mathbb{W}}$, F preserves cell composition and identities.

Definition (Morphism of virtual double theories)
Cartesian (preserves products $\otimes$ and I) and **normal** (preserves loose units U_A) virtual double functor.

Virtual double theories

Definition

$\mathbf{VDbTh}$ category of **small** virtual double theories and their morphisms:

- (locally posetal) cartesian virtual equipments and
- cartesian normal virtual double functors.

$\mathbf{VDBTH}$ for the large ones.

Theorem

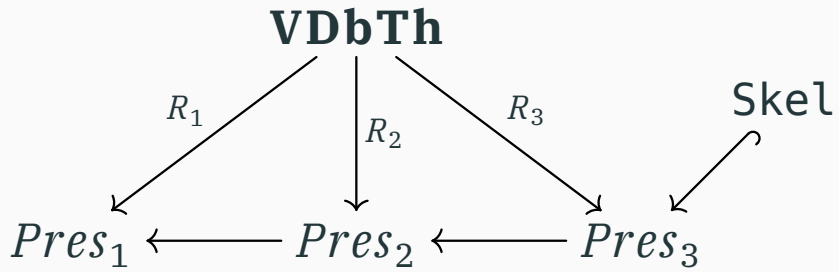
$\mathbf{VDbTh}$ is locally presentable.

Layered presentations

Freely build the virtual double theory from:

1. objects
2. tight and loose morphisms whose source and target are products
3. cells whose perimeter is made of companions, conjoints, products, ...

↳ a layered notion of presentation where n-objects have boudaries among freely generated $(n-1)$-objects



VDbTh locally presentable ⇒ **right adjoints** R_i

Roadmap

	Presentation		Theory		Models
Ehresmann[1]:	finite limit sketches	$\longrightarrow$	$\mathbf{Lex}$	$\longrightarrow$	$\mathbf{LPCAT}$
			T	$\longmapsto$	$\mathbf{LEX}(T, \mathbf{Set})$
This work:	Skel[2] programs	$\longrightarrow$	$\mathbf{VDbTh}$	$\longrightarrow$	$\mathbf{LPCAT}$
			$\mathbb{T}$	$\longmapsto$	$\mathbf{VDBTH}(\mathbb{T}, \mathbb{Rel})$

1. Ehresmann C. 1965. *Catégories et Structures*. Dunod
2. Bodin M, Gardner P, Jensen T, Schmitt A. 2019. Skeletal Semantics and Their Interpretations

Models of virtual double theories

Models of virtual double theories

Definition (Virtual double natural transformation $\varphi : F \Rightarrow G$)

- (tight) natural transformation $\varphi : \underline{F} \Rightarrow \underline{G}$
- "natural" mapping

$$A \overset{p}{\nrightarrow} B \quad \mapsto \quad \begin{array}{ccc} FA & \overset{Fp}{\nrightarrow} & FB \\ {\scriptstyle \varphi_A}\downarrow & \Downarrow \varphi_p & \downarrow{\scriptstyle \varphi_B} \\ GA & \underset{Gp}{\nrightarrow} & GB \end{array}$$

Remark

With transformations, $\mathbf{VDBTH}$ is actually a 2-category.

Definition (Model)

For $\mathbb{T} \in \mathbf{VDbTh}$, $\mathbf{Mod}(\mathbb{T}) := \mathbf{VDBTH}(\mathbb{T}, \mathbb{Rel})$.

Theorem

For $\mathbb{T} \in \mathbf{VDbTh}$, $\mathbf{Mod}(\mathbb{T})$ is locally presentable.

Models of virtual double theories

Example

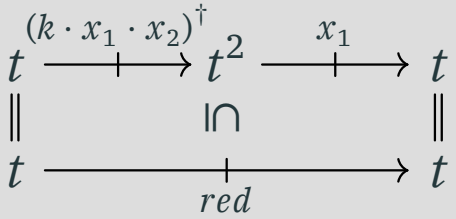
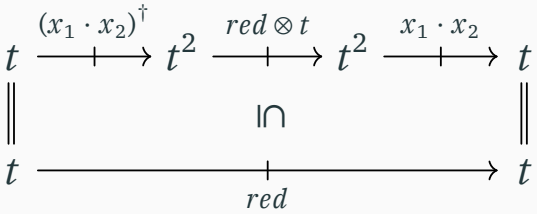
Virtual double theory $\mathbb{E}$ = generated by $e : t \nrightarrow t$

- Model $M : \mathbb{E} \to \mathbb{R}\mathrm{el}$
  ↳ set $X = M(t)$ equipped with an endorelation $r = M(e)$
- Model morphism $\varphi : (X, r) \Rightarrow (X', r')$
  ↳ function $f : X \to X' \quad (= \varphi_t : M(t) \to M'(t))$ s.t.

$$\forall x, y \in X, r(x, y) \Rightarrow r'(f(x), f(y))$$

$$\begin{array}{ccc} X & \overset{r}{\nrightarrow} & X \\ {\scriptstyle f}\downarrow & \cap & \downarrow{\scriptstyle f} \\ X' & \underset{r'}{\nrightarrow} & X' \end{array} \quad = \quad \begin{array}{ccc} Mt & \overset{Me}{\nrightarrow} & Mt \\ {\scriptstyle \varphi_t}\downarrow & \cap & \downarrow{\scriptstyle \varphi_t} \\ Mt' & \underset{Me'}{\nrightarrow} & Mt' \end{array}$$

Example Models of SK calculus: $\mathbf{Mod}(\mathbb{T}_{\mathsf{SK}})$

Skel program	$\rightsquigarrow$	$\mathbb{T}_{\mathsf{SK}}$	$\xrightarrow{M}$	$\mathbb{R}\mathrm{el}$
`type term`		t		a set T
`S`, `K`		$s, k : I \to t$		elements $\mathbf{S}, \mathbf{K} \in T$
`App`		$app : t \otimes t \to t$		function $\bullet : T \times T \to T$
`val red : term → term`		$red : t \nrightarrow t$		relation $\rightarrowtriangle \subseteq T \times T$
`let K·x1·x2 = e in x1`				$\forall u, v \in T, \mathbf{K} \bullet u \bullet v \rightarrowtriangle u$
`let x1·x2 = e in` `let x1 = red x1 in x1·x2`				$\forall u, v, u' \in T, u \rightarrowtriangle u' \Rightarrow$ $u \bullet v \rightarrowtriangle u' \bullet v$
…		…		…

Example Models of SK calculus: $\mathbf{Mod}(\mathbb{T}_{SK})$

Initial model

- T is the set *inductively defined* from $\mathbf{S}$, $\mathbf{K}$ and $\bullet$:

$$T \cong \{\, m, n ::= S \mid K \mid m \cdot n \,\}$$

  ↳ terms of SK calculus

- $\twoheadrightarrow$ is the relation *inductively defined* from the 4 inclusions:

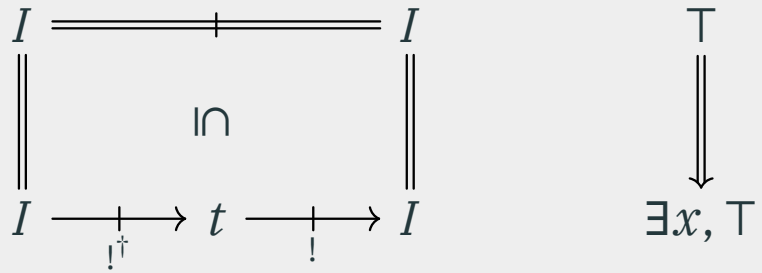
$$\twoheadrightarrow \cong \rightarrow$$

Initial model ≅ SK calculus as defined in the first part! 🥳

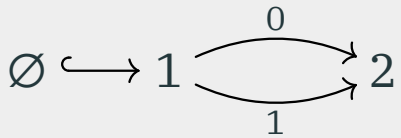
Virtual? double categories

- merely need virtual for $\mathtt{Skel}$
- actually necessary for local presentability with models in $\mathbb{Rel}$

Example (Theory of non-empty sets)



Models = non-empty sets ⇝ not locally presentable 😔 (not complete)



(Remark: in $\mathbb{Span}$, pointed sets: locally presentable![6])

6. Carlson K, Patterson E. 2026. Presheaves on Lax Double Functors; or, Instances of Models of Double Theories

Conclusion

Conclusion

`Skel`	*Virtual double theory*
types and constructors a.k.a. terms	tight category
tuples	cartesian
skeletons	loose paths
`val f = branch s₁ or ... or sₙ end`	cells $s_1 \subseteq f, ..., s_n \subseteq f$
terms $\subseteq$ skeletons	companion
pattern matching	conjoint
derivation	cell composition

Conclusion

`Skel`	*Virtual double theory*
types and constructors a.k.a. terms	tight category
tuples	cartesian
skeletons	loose paths
`val f = branch s₁ or ... or sₙ end`	cells $s_1 \subseteq f, ..., s_n \subseteq f$
terms $\subseteq$ skeletons	companion
pattern matching	conjoint
derivation	cell composition

	Presentation		Theory		Models
Ehresmann:	finite limit sketch	$\longmapsto$	$C \in \mathbf{Lex}$	$\longmapsto$	$\mathbf{LEX}(C, \mathbf{Set})$
This work:	`Skel` program	$\longmapsto$	$\mathbb{C} \in \mathbf{VDbTh}$	$\longmapsto$	$\mathbf{VDBTH}(\mathbb{C}, \mathbb{Rel})$

Locally presentable category of theories and categories of models!

Conclusion

`Skel`	*Virtual double theory*
types and constructors a.k.a. terms	tight category
tuples	cartesian
skeletons	loose paths
`val f = branch s₁ or ... or sₙ end`	cells $s_1 \subseteq f, \ldots, s_n \subseteq f$
terms $\subseteq$ skeletons	companion
pattern matching	conjoint
derivation	cell composition

	Presentation		Theory		Models
Ehresmann:	finite limit sketch	$\longmapsto$	$C \in \mathbf{Lex}$	$\longmapsto$	$\mathbf{LEX}(C, \mathbf{Set})$
This work:	`Skel` program	$\longmapsto$	$\mathbb{C} \in \mathbf{VDbTh}$	$\longmapsto$	$\mathbf{VDBTH}(\mathbb{C}, \mathbb{Rel})$

Locally presentable category of theories and categories of models!

Thanks! Questions?

Powered by Typst

Appendix

Future work

- unspecified terms and functions
- abstract interpretation
- what other PL techniques & theorems can we account for in our framework?
- syntax-semantics duality
- change the monad (from $\mathbb{Rel} = \mathbb{Kl}(\mathcal{P})$ to $\mathbb{Kl}(T)$)
- higher order

Bibliography

[1] C. Ehresmann, *Catégories et Structures*. Dunod, 1965.

[2] M. Bodin, P. Gardner, T. Jensen, and A. Schmitt, “Skeletal Semantics and Their Interpretations,” no. POPL, 2019, doi: 10.1145/3290357.

[3] A. Khayam, V. Noizet, and A. Schmitt, “JSkel: Towards a Formalization of JavaScript's Semantics,” in *JFLA 2021 - Journées Francophones Des Langages Applicatifs*, Virtuelles, France, Apr. 2021, pp. 1–22. Accessed: Jan. 26, 2026. [Online]. Available: https://inria.hal.science/hal-03509431

[4] M. Andrieux and A. Schmitt, “Skeletal Semantics of a Fragment of Python,” in *35es Journées Francophones Des Langages Applicatifs (JFLA 2024)*, Saint-Jacut-de-la-Mer, France, Jan. 2024. Accessed: Jan. 26, 2026. [Online]. Available: https://inria.hal.science/hal-04406392

[5] M. Schönfinkel, “Über die Bausteine der mathematischen Logik,” *Mathematische Annalen*, vol. 92, no. 3, pp. 305–316, Sep. 1924, doi: 10.1007/BF01448013.

Bibliography

[6] K. Carlson and E. Patterson, “Presheaves on Lax Double Functors; or, Instances of Models of Double Theories,” no. arXiv:2510.08861. arXiv, Mar. 2026. doi: 10.48550/arXiv.2510.08861.