

Relational Models of Skeletal Specifications

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

LIMD seminar – September 24, 2026

Université Savoie Mont Blanc

Introduction

My object of study:

PROGRAMMING LANGUAGES

↳ build mathematical models

Goal  Generic method

Description of a language

$\mapsto$

Its model

Introduction

My object of study:

PROGRAMMING LANGUAGES

↳ build mathematical models

Goal  Generic method

Description of a language

↳ Skel

⟶

Its model

Design a notion of model

- suitable for Skel
- good mathematical properties

- ▷  **calculus**
- ▷ **Skeletal semantics**
- ▷ **Initial-algebra semantics**
- ▷ **Virtual double theories**



calculus



SKI calculus^[1] a.k.a. combinatory logic

- programming language/computational system
- simple syntax
- computation = reduction relation
- Turing complete i.e. equivalent to λ -calculus

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

Syntax

Syntax = shape of programs $\rightsquigarrow$ what does a program **look like**?

In SKI calculus: programs = **terms**

Build from:

- 3 constants S , K and I
- application “ $- \cdot -$ ”

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

Example

S

K

I

$K \cdot I$

associativity: $m \cdot n \cdot p := (m \cdot n) \cdot p$

$S \cdot I \cdot K = (S \cdot I) \cdot K$

$S \cdot (I \cdot K)$

$I \cdot m$

$I \cdot m \cdot n$

$I \cdot m \cdot (I \cdot n)$

$K \cdot I \cdot S \cdot S$

$S \cdot I \cdot K \cdot S$

$S \cdot K \cdot I \cdot S$

Semantics = computation $\rightsquigarrow$ what does a program **do**?

In SKI calculus:

- a term **transforms** (“reduces”) into another term
- a reduction relation “ $\rightarrow$ ”
- given by deduction rules

$$\frac{n_1 \rightarrow n'_1 \quad \dots \quad n_k \rightarrow n'_k \quad \} \text{ hypotheses}}{m \rightarrow m' \quad \} \text{ conclusion}}$$

Semantics

Deduction rules for “ $\rightarrow$ ”:

$$\frac{}{I \cdot m_1 \rightarrow m_1} I$$

$$\frac{}{K \cdot m_1 \cdot m_2 \rightarrow m_1} K$$

$$\frac{}{S \cdot m_1 \cdot m_2 \cdot m_3 \rightarrow (m_1 \cdot m_3) \cdot (m_2 \cdot m_3)} S$$

$$\frac{m_1 \rightarrow m'_1}{m_1 \cdot m_2 \rightarrow m'_1 \cdot m_2} \text{app}_l$$

$$\frac{m_2 \rightarrow m'_2}{m_1 \cdot m_2 \rightarrow m_1 \cdot m'_2} \text{app}_r$$

Example 

S
 $I \cdot m \cdot n$

K
 $I \cdot m \cdot (I \cdot n)$

I
 $K \cdot I \cdot S \cdot S$

$K \cdot I$
 $S \cdot I \cdot K \cdot S$

$S \cdot I \cdot K$
 $I \cdot m$
 $S \cdot K \cdot I \cdot S$

Turing completeness

Definition (Church-Turing Thesis)

A function is **computable** *if and only if* it can be defined in the λ -calculus (*if and only if* it can be computed by a Turing machine).

Definition

A computational system is **Turing complete** if it can compute all computable functions.

Theorem

SKI calculus is Turing complete.

Proof: Simulate the λ -calculus in the SKI calculus

□

Turing completeness – Proof

Definition (λ -calculus)

$$t, u ::= x \mid t \cdot u \mid \lambda x. t$$

$x \in Var$ a set of variables.

β -reduction “ $\rightarrow_\beta$ ” defined by the rules:

$$\frac{}{(\lambda x. t) \cdot u \rightarrow_\beta t[x/u]} \beta \quad \frac{t \rightarrow_\beta t'}{\lambda x. t \rightarrow_\beta \lambda x. t'} \lambda \quad \frac{t \rightarrow_\beta t'}{t \cdot u \rightarrow_\beta t' \cdot u} \text{app}_l \quad \frac{u \rightarrow_\beta u'}{t \cdot u \rightarrow_\beta t \cdot u'} \text{app}_r$$

where $t[x/u]$ is the term t where each occurrence of x is replaced by u

Goal  simulate β reduction.

Turing completeness – Proof

Intermediate language $\Lambda^{\text{SKI}} = \lambda\text{-calculus} + \text{constants } S, K \text{ and } I$

Define a transformation $\llbracket - \rrbracket$ on Λ^{SKI}

$$\llbracket c \rrbracket := c \quad \text{if } c \in \text{Var} \cup \{S, K, I\}$$

$$\llbracket t \cdot u \rrbracket := \llbracket t \rrbracket \cdot \llbracket u \rrbracket$$

$$\llbracket \lambda x. t \rrbracket := K \cdot \llbracket t \rrbracket \quad \text{if } x \notin t$$

$$\llbracket \lambda x. x \rrbracket := I$$

$$\llbracket \lambda x. \lambda y. t \rrbracket := \llbracket \lambda x. \llbracket \lambda y. t \rrbracket \rrbracket$$

$$\llbracket \lambda x. t \cdot u \rrbracket := S \cdot \llbracket \lambda x. t \rrbracket \cdot \llbracket \lambda x. u \rrbracket$$

Proposition

If t is a closed λ -term, $\llbracket t \rrbracket$ has no λ and no variable: $\llbracket t \rrbracket$ is a term of SKI calculus.

Example

$\lambda x. x \cdot x$

$\lambda x. f \cdot x$

$\lambda x. \lambda y. x$

$\lambda x. \lambda y. y \cdot x$

Turing completeness – Proof

Lemma

For all t, u , if $t \rightarrow_{\beta} u$ then $\llbracket t \rrbracket \rightarrow^{\star} \llbracket u \rrbracket$ and if $t \nrightarrow_{\beta}$ then $\llbracket t \rrbracket \nrightarrow$.

Example

$(\lambda x.x \cdot x) \cdot a$ $(\lambda x.f \cdot x) \cdot a$ $(\lambda x.\lambda y.x) \cdot a \cdot b$ $(\lambda x.\lambda y.y \cdot x) \cdot a \cdot b$

$\Rightarrow$ Every computation in the λ -calculus can be performed in the SKI calculus!

Remark

I is superfluous: it can be simulated with $S \cdot K \cdot K$ 

$\hookrightarrow$ rest of this talk: SK calculus

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” $\rightsquigarrow$ 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

Skeletal semantics by example

Example (Running example: SK calculus)

SK calculus

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

$K, S, (S \cdot K) \cdot K, \dots$

reduction relation “ $\rightarrow$ ”

Skel

`type term = S | K | App (term, term)`

`K, S, App(App(S, K), K), ...`

`val red : term → term`

Notation

$M \cdot N$ for `App(M, N)` + associativity: e.g. $S \cdot K \cdot K = \text{App}(\text{App}(S, K), K)$

Skeletal semantics by example

Example (Running example: SK calculus)

$$\frac{}{K \cdot m_1 \cdot m_2 \rightarrow m_1} K$$

```
val red (t:term) : term =  
  branch  
    let K·x1·x2 = t in x1  } skeleton
```

$$\frac{}{S \cdot m_1 \cdot m_2 \cdot m_3 \rightarrow (m_1 \cdot m_3) \cdot (m_2 \cdot m_3)} S$$

```
or  
  let S·x1·x2·x3 = t in  
    (x1·x3) · (x2·x3)
```

$$\frac{m_1 \rightarrow m'_1}{m_1 \cdot m_2 \rightarrow m'_1 \cdot m_2} \text{app}_l$$

```
or  
  let x1·x2 = t in  
  let x1 = red x1 in x1·x2
```

$$\frac{m_2 \rightarrow m'_2}{m_1 \cdot m_2 \rightarrow m_1 \cdot m'_2} \text{app}_r$$

```
or  
  let x1·x2 = t in  
  let x2 = red x2 in x1·x2  
end
```

Initial-algebra semantics

Initial-algebra semantics

Goal  Get a **mathematical** representation/definition of a language or system
↳ e.g. prove and apply abstract theorems, study properties, etc.

Specify programming languages **implicitly**:

- first define a whole **category of models**,
- desired language implicitly defined as **initial object**^{*} therein

^{*}(up to isomorphism)

(Category theory – 1/4 Categories)

Definition (Category C)

- a collection $\text{ob } C$ of **objects**
- a collection $\text{mor } C$ of **morphisms** such that
 - each morphism $f \in \text{mor } C$ has a **source** A and a **target** B ($A, B \in \text{ob } C$)

$$f : A \rightarrow B$$

- morphisms **compose**: if $f : A \rightarrow B$ and $g : B \rightarrow C$,

$$g \circ f : A \rightarrow C$$

- there are **identities**: for any $A \in \text{ob } C$,

$$\text{id}_A : A \rightarrow A \quad \text{id}_A \circ f = f \quad f \circ \text{id}_A = f$$

Example

Sets and functions $\rightsquigarrow$ a category **Set**, groups and group homomorphisms $\rightsquigarrow$ **Grp**, any poset, ...

(Category theory – 2/4 Functors)

Definition (Functor)

C and D are categories, a **functor** $F : C \rightarrow D$ is

- a function on **objects** $F : \text{ob } C \rightarrow \text{ob } D$
- that extends to **morphisms**

$$F(f : A \rightarrow B) = F(f) : FA \rightarrow FB$$

- that **preserves identities and composition**

$$F(\text{id}_A) = \text{id}_{FA} \quad F(g \circ f) = Fg \circ Ff$$

Example

“Forgetful” functor $U : \mathbf{Grp} \rightarrow \mathbf{Set}, \dots$

Proposition

Categories and functors $\rightsquigarrow$ a category **Cat**

(Category theory – 3/4 Some limits and colimits)

Definition (Initial object)

Object X is **initial** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : X \rightarrow Y \in \text{mor } \mathcal{C} \rightsquigarrow$ “smallest” object

Definition (Terminal object)

Object X is **terminal** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : Y \rightarrow X \in \text{mor } \mathcal{C} \rightsquigarrow$ “biggest” object

Example

In a poset,

(Category theory – 3/4 Some limits and colimits)

Definition (Initial object)

Object X is **initial** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : X \rightarrow Y \in \text{mor } \mathcal{C} \rightsquigarrow$ “smallest” object

Definition (Terminal object)

Object X is **terminal** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : Y \rightarrow X \in \text{mor } \mathcal{C} \rightsquigarrow$ “biggest” object

Example

In a poset, $\perp$ is initial and $\top$ is terminal.

(Category theory – 3/4 Some limits and colimits)

Definition (Initial object)

Object X is **initial** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : X \rightarrow Y \in \text{mor } \mathcal{C} \rightsquigarrow$ “smallest” object

Definition (Terminal object)

Object X is **terminal** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : Y \rightarrow X \in \text{mor } \mathcal{C} \rightsquigarrow$ “biggest” object

Example

In a poset, $\perp$ is initial and $\top$ is terminal.

In **Set**,

(Category theory – 3/4 Some limits and colimits)

Definition (Initial object)

Object X is **initial** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : X \rightarrow Y \in \text{mor } \mathcal{C} \quad \rightsquigarrow$ “smallest” object

Definition (Terminal object)

Object X is **terminal** in $\mathcal{C}$ iff $\forall Y \in \text{ob } \mathcal{C}, \exists! f : Y \rightarrow X \in \text{mor } \mathcal{C} \quad \rightsquigarrow$ “biggest” object

Example

In a poset, $\perp$ is initial and $\top$ is terminal.

In **Set**, $\emptyset$ is initial and $\{*\}$ is terminal.

(Category theory – 4/4 Some limits and colimits)

Definition (Product)

A **product** of $X, Y \in \text{ob } C$ is an object $X \times Y \in \text{ob } C$ such that

- there are **projection morphisms** $\pi_1 : X \times Y \rightarrow X$ and $\pi_2 : X \times Y \rightarrow Y$
- for any morphisms $f : A \rightarrow X, g : A \rightarrow Y$, there is

$$\langle f, g \rangle : A \rightarrow X \times Y$$

$$\hookrightarrow \Delta_X = \langle \text{id}_X, \text{id}_X \rangle : X \rightarrow X \times X$$

- for any morphisms $f : A \rightarrow X, h : B \rightarrow Y$, there is

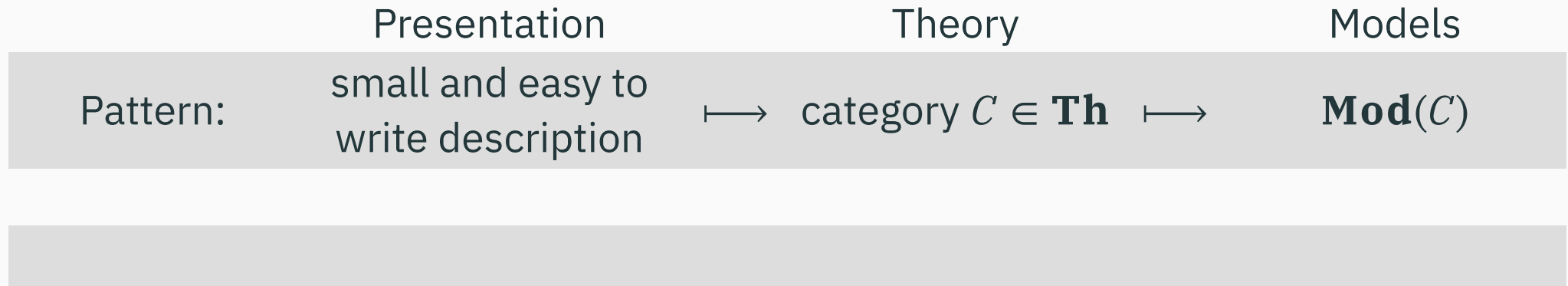
$$f \times h : A \times B \rightarrow X \times Y$$

Example

In **Set**, cartesian product.

$$\langle f, g \rangle(a) = (f(a), g(a)) \quad \Delta_X(x) = (x, x) \quad (f \times h)(a, b) = (f(a), h(b))$$

Initial-algebra semantics: usual pattern



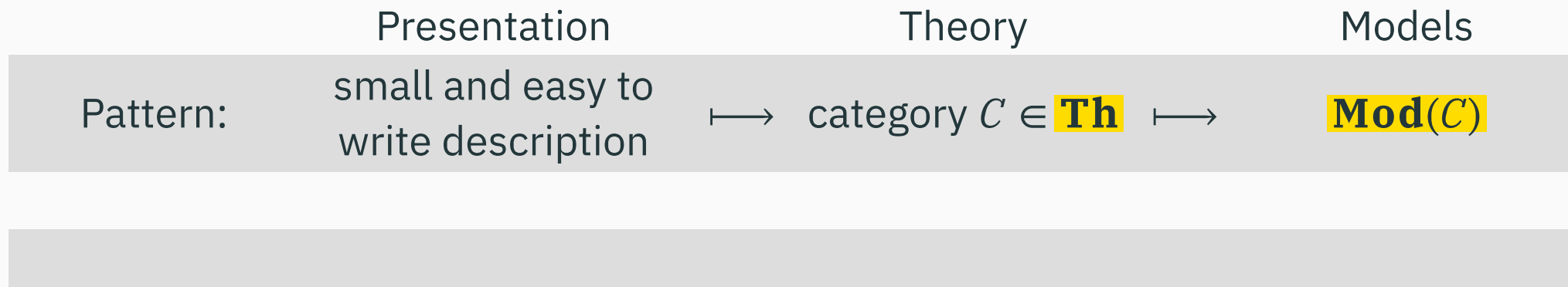
Initial-algebra semantics: usual pattern



Locally presentable categories

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

Initial-algebra semantics: usual pattern



Locally presentable categories

- categories with good properties (complete, cocomplete, ...)
 - closed under useful constructions.
- initial object 😊

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Pattern:	small and easy to write description	$\mapsto$	category $C \in \mathbf{Th}$	$\mapsto$	$\mathbf{Mod}(C)$
Ehresmann ^[5] :	finite limit sketch	$\mapsto$	$C \in \mathbf{Lex}$	$\mapsto$	$\mathbf{LEX}(C, \mathbf{Set})$

Locally presentable categories

- categories with good properties (complete, cocomplete, ...)
 - closed under useful constructions.
- initial object 😊

Lex = **l**eft **e**xact = categories with finite limits (products, terminal object).

Let's do an example!

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

Example: theory of groups

Definition ((Finite limit) sketch of groups)

- a sort X
- operations $\odot : X \times X \rightarrow X, i : X \rightarrow X, e : 1 \rightarrow X$
- equations:

$$(x \odot y) \odot z = x \odot (y \odot z)$$

$$x \odot i(x) = e$$

$$e \odot x = x$$

$$\begin{array}{ccc} X \times X \times X & \xrightarrow{\odot \times id} & X \times X \\ id \times \odot \downarrow & & \downarrow \odot \\ X \times X & \xrightarrow{\odot} & X \end{array}$$

$$\begin{array}{ccccc} X & \xrightarrow{\Delta} & X \times X & \xrightarrow{id \times i} & X \times X \\ ! \downarrow & & & & \downarrow \odot \\ 1 & \xrightarrow{e} & & & X \end{array}$$

$$\begin{array}{ccccc} X & \xrightarrow{\cong} & 1 \times X & \xrightarrow{e \times id} & X \times X \\ & \searrow & & & \downarrow \odot \\ & & & & X \end{array}$$

+ $i(x) \odot x = e$ and $x \odot e = x$ 

Example: theory of groups

Definition (Theory of groups)

Category (with finite limits) C_{grp} generated from this sketch:

- $X, 1, X \times X, X \times X \times X, \dots \in \text{ob } C_{\text{grp}}$
- $\odot : X \times X \rightarrow X, i : X \rightarrow X, e : 1 \rightarrow X \in \text{mor } C_{\text{grp}}$
 $\Delta : X \rightarrow X \times X, ! : X \rightarrow 1, \text{id} \times e : X \times 1 \rightarrow X \times X, \dots \in \text{mor } C_{\text{grp}}$
 $i \circ e, \odot \circ \Delta, \dots \in \text{mor } C_{\text{grp}}$
- equations are satisfied

Example: theory of groups

Definition (Models of C_{grp})

$$\mathbf{Mod}(C_{\text{grp}}) := \mathbf{LEX}(C_{\text{grp}}, \mathbf{Set})$$

Finite limit preserving *functors* from C_{grp} to **Set**. A model $M : C_{\text{grp}} \rightarrow \mathbf{Set}$

- maps X to a **set** $G = M(X)$,
- maps $\odot : X \times X \rightarrow X$ to a **function** $+: G \times G \rightarrow G$,
- maps $e : 1 \rightarrow X$ to a **function** $\{*\} \rightarrow G$ i.e. a element $0 \in G$,
- maps $i : X \rightarrow X$ to a **function** $- : G \rightarrow G$,
- preserves the equations: e.g.

$$\begin{array}{ccc} G & \xrightarrow{\Delta} & G \times G \xrightarrow{id \times -} G \times G \\ \downarrow ! & & \downarrow + \\ \{*\} & \xrightarrow{"0"} & G \end{array} \quad \Leftrightarrow \quad \begin{array}{l} + \circ (id \times -) \circ \Delta = 0 \\ \forall g \in G, g + -g = 0 \end{array}$$

...

Example: theory of groups

Definition (Models of C_{grp})

$$\mathbf{Mod}(C_{\text{grp}}) := \mathbf{LEX}(C_{\text{grp}}, \mathbf{Set}) = \mathbf{Grp} \text{ 😎}$$

Finite limit preserving *functors* from C_{grp} to **Set**. A model $M : C_{\text{grp}} \rightarrow \mathbf{Set}$

- maps X to a **set** $G = M(X)$,
- maps $\odot : X \times X \rightarrow X$ to a **function** $+: G \times G \rightarrow G$,
- maps $e : 1 \rightarrow X$ to a **function** $\{*\} \rightarrow G$ i.e. a element $0 \in G$,
- maps $i : X \rightarrow X$ to a **function** $- : G \rightarrow G$,
- preserves the equations: e.g.

$$\begin{array}{ccc} G & \xrightarrow{\Delta} & G \times G \xrightarrow{\text{id} \times -} G \times G \\ \downarrow ! & & \downarrow + \\ \{*\} & \xrightarrow{\text{"0"}} & G \end{array} \quad \Leftrightarrow \quad \begin{array}{l} + \circ (\text{id} \times -) \circ \Delta = 0 \\ \forall g \in G, g + -g = 0 \end{array}$$

...

Example: theory of groups

Remark (Model morphism)

Model morphism = function between the sets that preserves the operations

↳ *group homomorphism*

Remark (Initial group)

Trivial group $G = \{0\}$

Not a very interesting programming language...

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Pattern:	small and easy to write description	$\mapsto$	category $C \in \mathbf{Th}$	$\mapsto$	$\mathbf{Mod}(C)$
Ehresmann ^[5] :	finite limit sketch	$\mapsto$	$C \in \mathbf{Lex}$	$\mapsto$	$\mathbf{LEX}(C, \mathbf{Set})$

Locally presentable categories

$\hookrightarrow$ initial object 😊

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Pattern:	small and easy to write description	$\mapsto$	category $C \in \mathbf{Th}$	$\mapsto$	$\mathbf{Mod}(C)$
Ehresmann ^[5] :	finite limit sketch	$\mapsto$	$C \in \mathbf{Lex}$	$\mapsto$	$\mathbf{LEX}(C, \mathbf{Set})$

Locally presentable categories

$\hookrightarrow$ initial object 😊

Not so convenient for programming languages: need **relations** (e.g. “ $- \rightarrow -$ ”)

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

Initial-algebra semantics: usual pattern

	Presentation		Theory		Models
Pattern:	small and easy to write description	$\mapsto$	category $C \in \mathbf{Th}$	$\mapsto$	$\mathbf{Mod}(C)$
Ehresmann ^[5] :	finite limit sketch	$\mapsto$	$C \in \mathbf{Lex}$	$\mapsto$	$\mathbf{LEX}(C, \mathbf{Set})$
Our work:	Skel program	$\mapsto$	$\mathbb{C} \in \mathbf{VDbTh}$	$\mapsto$	$\mathbf{VDBTH}(\mathbb{C}, \mathbf{Rel})$

Locally presentable categories

↳ initial object 😊

Not so convenient for programming languages: need **relations** (e.g. “ $- \rightarrow -$ ”)

↳ **VDbTh** = **V**irtual **D**ouble **T**heories = to be defined.

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

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

$s, k: I \rightarrow t$ and $app: t \otimes t \rightarrow t$

val $red: term \rightarrow term$

loose morphism $red: t \rightrightarrows t$

skeletons **let** ... = ... **in** ...

loose **paths** $A_0 \xrightarrow{p_1} A_1 \dots A_{n-1} \xrightarrow{p_n} A_n$

val $red(t:term): term =$

branch

skeleton1

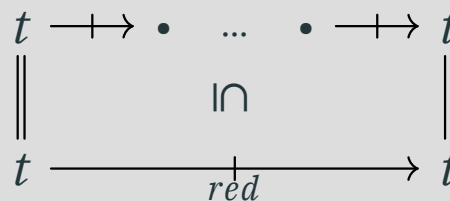
or ... **or**

skeleton4

end

$\forall 1 \leq i \leq 4$, a **cell**

$skeleton_i$



Virtual double theories

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

1. virtual double category
2. cartesian
3. equipment

Example (Running example: SK calculus)
...and build up the virtual double theory

Virtual double categories

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

Like a category, with *two kind of morphisms*:

$$\begin{array}{ccccccc} A_0 & \xrightarrow{p_1} & A_1 & \xrightarrow{p_2} & A_2 & \cdots & A_{n-1} & \xrightarrow{p_n} & A_n \\ f \downarrow & & & & \text{in} & & & & \downarrow g \\ B & \xrightarrow{\quad} & & & q & \xrightarrow{\quad} & & & C \end{array}$$

- objects
- **tight (vertical) morphisms** $\rightsquigarrow$ category (composition and identities)
- **loose (horizontal) morphisms** ⚠ no composition
- n -ary cells

Virtual double categories

Example (Rel)

- objects $A, B =$ sets
- tight morphisms $f : A \rightarrow B =$ functions
- loose morphisms $r : A \rightrightarrows B =$ binary relations $r \subseteq A \times B$
- a n -ary cell $r_1; \dots; r_n \xrightarrow[f]{g} r'$ whenever

$$\forall a, b, (r_1; \dots; r_n)(a, b) \Rightarrow r'(f(a), g(b))$$

$$\begin{array}{ccccc} A_0 & \xrightarrow{r_1} & A_1 & \dots & \xrightarrow{r_n} & A_n \\ f \downarrow & & & \text{Id} & & \downarrow g \\ C & \xrightarrow{\quad} & & \xrightarrow{r'} & & D \end{array}$$

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

Cartesian virtual double categories

Definition (Cartesian virtual double category)

Virtual double category with

- tight finite products $A \otimes B$ and I ,
- “acting on loose morphisms”.

Example ($\mathbf{Rel}$)

Cartesian product of sets $A \times B$ and $\{*\}$

Cartesian virtual double categories

Example (SK calculus – building the virtual double theory)

- objects
 - **freely add** t
 - **cartesian** $\Rightarrow$ products $I, t \otimes t = t^2, t^3, \dots$
- tight morphisms
 - **freely add** $s, k: I \rightarrow t$ and $app: t \otimes t \rightarrow t$
 - **cartesian** $\Rightarrow$ projections $x_i: t^2 \rightarrow t, \dots$
- loose morphisms
 - **freely add** $red: t \rightharpoonup t$
 - **cartesian** $\Rightarrow$ action on loose morphisms $red \otimes t: t^2 \rightharpoonup t^2$

(to be continued in 2 slides...)

Virtual equipments

Definition (Virtual equipment)

Virtual double category with **loose unit** $U_A : A \rightrightarrows A$ for any A and **companion** $f : A \rightrightarrows B$ and **conjoint** $f^\dagger : B \rightrightarrows A$, for any $f : A \rightarrow B$.

$$\begin{array}{c} A \\ \swarrow \quad \searrow \\ A \quad \text{In} \quad A \\ \text{---} U_A \text{---} \end{array} \quad \begin{array}{ccc} A & \xrightarrow{f} & B \\ f \downarrow & \text{In} & \parallel \\ B & \text{---} & B \end{array} \quad \begin{array}{ccc} B & \xrightarrow{f^\dagger} & A \\ \parallel & \text{In} & \downarrow f \\ B & \text{---} & B \end{array} \quad \text{and axioms...}$$

Example (Rel)

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

Virtual equipments

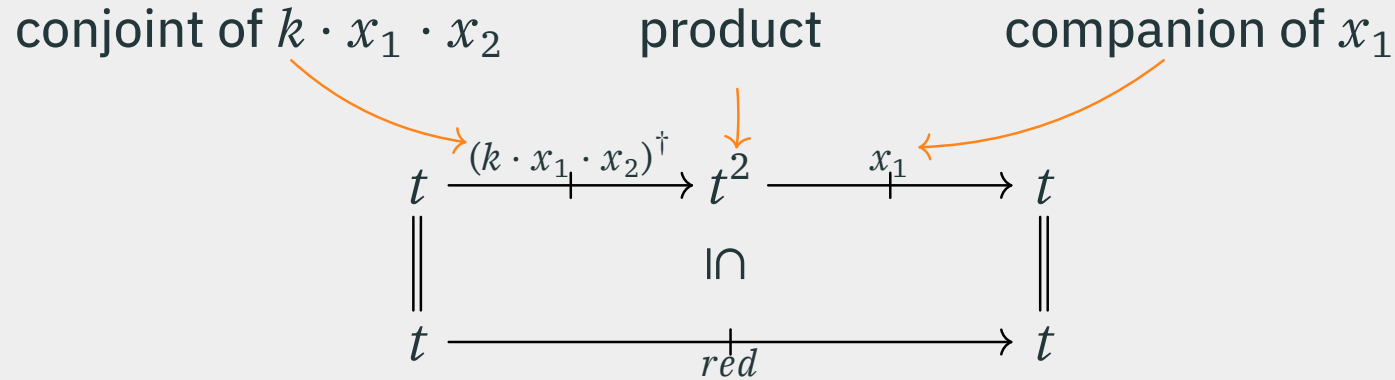
Recall

$$\frac{}{K \cdot m_1 \cdot m_2 \rightarrow m_1} K$$

let $K \cdot x_1 \cdot x_2 = e$ in x_1

Example (SK calculus – virtual double theory, continued)

- **equipment** $\Rightarrow$ conjoiners and companions
- **freely add**



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

Virtual equipments

Recall

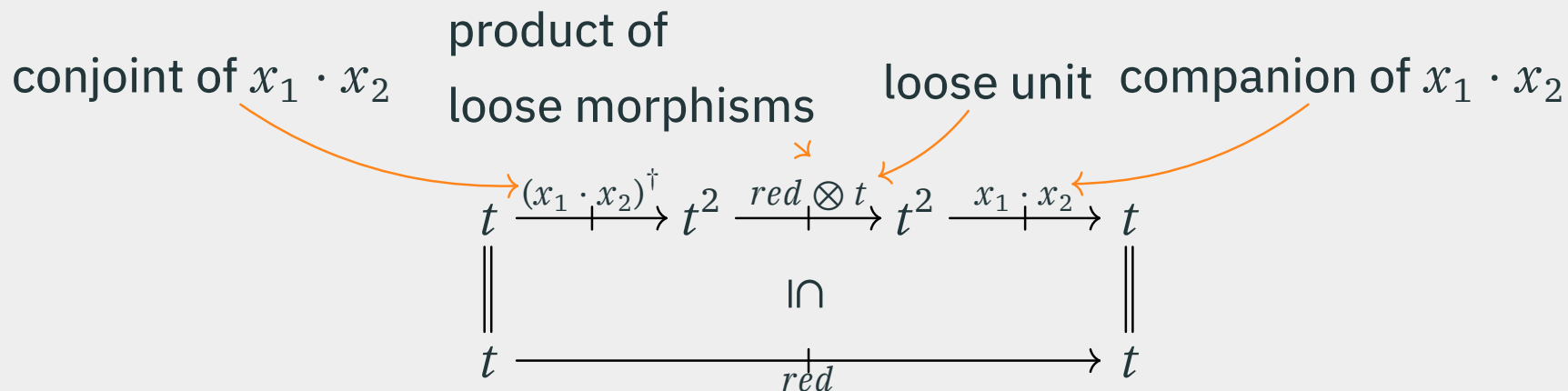
$$\frac{m_1 \rightarrow m'_1}{m_1 \cdot m_2 \rightarrow m'_1 \cdot m_2}$$

```
let x1 · x2 = e in
```

```
let x1 = red x1 in x1 · x2
```

Example (SK calculus – virtual double theory, continued)

- freely add



where

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

Virtual equipments

Example (SK calculus – virtual double theory: summary)

$$\begin{array}{ccccc} t & \xrightarrow{(k \cdot x_1 \cdot x_2)^\dagger} & t^2 & \xrightarrow{x_1} & t \\ \parallel & & \text{In} & & \parallel \\ t & \xrightarrow{\text{red}} & & & t \end{array}$$

let $K \cdot x_1 \cdot x_2 = e$ in x_1

$$\begin{array}{ccccc} t & \xrightarrow{(x_1 \cdot x_2)^\dagger} & t^2 & \xrightarrow{\text{red} \otimes t} & t^2 & \xrightarrow{x_1 \cdot x_2} & t \\ \parallel & & & \text{In} & & & \parallel \\ t & \xrightarrow{\text{red}} & & & & & t \end{array}$$

let $x_1 \cdot x_2 = e$ in

let $x_1 = \text{red } x_1$ in $x_1 \cdot x_2$

$$\begin{array}{ccccc} t & \xrightarrow{(s \cdot x_1 \cdot x_2 \cdot x_3)^\dagger} & t^3 & \xrightarrow{(x_1 \cdot x_3) \cdot (x_2 \cdot x_3)} & t \\ \parallel & & \text{In} & & \parallel \\ t & \xrightarrow{\text{red}} & & & t \end{array}$$

let $S \cdot x_1 \cdot x_2 \cdot x_3 = e$ in

$(x_1 \cdot x_3) \cdot (x_2 \cdot x_3)$

$$\begin{array}{ccccc} t & \xrightarrow{(x_1 \cdot x_2)^\dagger} & t^2 & \xrightarrow{t \otimes \text{red}} & t^2 & \xrightarrow{x_1 \cdot x_2} & t \\ \parallel & & & \text{In} & & & \parallel \\ t & \xrightarrow{\text{red}} & & & & & t \end{array}$$

let $x_1 \cdot x_2 = e$ in

let $x_2 = \text{red } x_2$ in $x_1 \cdot x_2$

Notice: $\text{let } \dots = \dots \text{ in } \dots \rightsquigarrow$ sequencing of loose morphisms

Virtual double theories

Definition (Virtual double theory)

Cartesian virtual equipment.

$\rightsquigarrow$ with structure-preserving functors: a category **VDbTh**

Theorem

VDbTh is locally presentable.

Definition (Model)

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

Example

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

- Model M ?

Definition (Model)

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

Example

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

- Model M ?
 - set $X = M(t)$ equipped with
 - endorelation $r = M(e)$

Definition (Model)

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

Example

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

- Model M ?
 - set $X = M(t)$ equipped with
 - endorelation $r = M(e)$
- model morphism $\varphi : (X, r) \Rightarrow (X', r')$
 - function $f : X \rightarrow X'$, such that
 - $\forall x, y \in X, r(x, y) \Rightarrow r'(f(x), f(y))$

Theorem

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

Models of SK(I) calculus

Skel program	$\rightsquigarrow$	$\mathbb{C}_{\text{SKI}}$	$\xrightarrow{M}$	$\mathbb{R}\text{el}$
type term		t		a set T
S , K		$s, k : I \rightarrow t$		elements S , K $\in T$
App		$app : t \otimes t \rightarrow t$		function $\cdot : T \times T \rightarrow T$
val red : term $\rightarrow$ term		$red : t \twoheadrightarrow t$		relation $\twoheadrightarrow \subseteq T \times T$
let K .x1.x2 = e in x1		$ \begin{array}{ccccc} t & \xrightarrow{(k \cdot x_1 \cdot x_2)^\dagger} & t^2 & \xrightarrow{x_1} & t \\ \parallel & & \text{in} & & \parallel \\ t & \xrightarrow{\text{red}} & t & & t \end{array} $	$\forall u, v \in T, \mathbf{K} \cdot u \cdot v \twoheadrightarrow u$ 	
let x1.x2 = e in let x1 = red x1 in x1.x2		$ \begin{array}{ccccccc} t & \xrightarrow{(x_1 \cdot x_2)^\dagger} & t^2 & \xrightarrow{\text{red} \otimes t} & t^2 & \xrightarrow{x_1 \cdot x_2} & t \\ \parallel & & & & & & \parallel \\ t & \xrightarrow{\text{red}} & t & & t & & t \end{array} $	$\forall u, v, u' \in T, u \twoheadrightarrow u' \Rightarrow$ $u \cdot v \twoheadrightarrow u' \cdot v$ 	
...		...		...

Models of SK(I) calculus

Initial model

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

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

↳ terms of SK calculus

- $\Rightarrow$ is the relation *inductively defined* from the 4 implications:

$$\Rightarrow \cong \rightarrow$$

Initial model $\cong$ SK calculus as defined in the first part! 🎉

Conclusion

Conclusion

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

	Presentation		Theory		Models
Pattern:	small description	$\mapsto$	category $C \in \mathbf{Th}$	$\mapsto$	$\mathbf{Mod}(C)$
Ehresmann:	finite limit sketch	$\mapsto$	$C \in \mathbf{Lex}$	$\mapsto$	$\mathbf{LEX}(C, \mathbf{Set})$
Our work:	Skel program	$\mapsto$	$\mathbb{C} \in \mathbf{VDbTh}$	$\mapsto$	$\mathbf{VDBTH}(\mathbb{C}, \mathbf{Rel})$

Locally presentable **category of theories** and **categories of models**!

Conclusion

Skel

types and constructors a.k.a. terms

tuples

skeletons

`val f = branch s1 or ... or sn end`

terms $\subseteq$ skeletons

pattern matching

derivation

Virtual double theory

tight category

cartesian

loose paths

cells $s_1 \subseteq f, \dots, s_n \subseteq f$

companion

conjoint

cell composition

Presentation

small description

Ehresmann: finite limit sketch

Our work: Skel program

Theory

category $C \in \mathbf{Th}$

$C \in \mathbf{Lex}$

$\mathbb{C} \in \mathbf{VDbTh}$

Models

$\mathbf{Mod}(C)$

$\mathbf{LEX}(C, \mathbf{Set})$

$\mathbf{VDBTH}(\mathbb{C}, \mathbf{Rel})$

Locally presentable **category of theories** and **categories of models**!

Appendix

Virtual? double categories

- merely need virtual for $\mathbf{Skel}$
- actually necessary for local presentability in $\mathbf{Rel}$

Example (Theory of non-empty sets)

$$\begin{array}{ccc}
 I & \xlongequal{\quad} & I \\
 \parallel & \text{in} & \parallel \\
 I & \xrightarrow{\quad} t \xrightarrow{\quad} & I
 \end{array}
 \quad
 \begin{array}{c}
 \top \\
 \Downarrow \\
 \exists x, \top
 \end{array}$$

Models = non-empty sets $\rightsquigarrow$ not locally presentable (not complete)

$$\emptyset \twoheadrightarrow 1 \begin{array}{c} \xrightarrow{0} \\ \xleftarrow{1} \end{array} 2$$

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

Bibliography

- [1] M. Schönfinkel, “Über die Bausteine der mathematischen Logik,” *Mathematische Annalen*, vol. 92, no. 3, pp. 305–316, Sep. 1924, doi: 10.1007/BF01448013.
- [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] C. Ehresmann, *Catégories et Structures*. Dunod, 1965.

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