Formal computational models and non-standard finiteness

Edward Hermann Haeusler

Department of Informatics - PUC-Rio - Brasil
hermann@inf.puc-rio.br

VII workshop de verão Matemática UnB

Foundational motivation

Phi	What is a finite collection of objects ?
Phi/Math	What is an infinite collection of objects ?
Science	What is a computable collection of objects ?

Preamble: The Ontology of Mathematics

Sets (XIX c.) versus **Categories** (XX c.)

$\in$ versus $\rightarrow$

"identity provided as free" vs "identity on arrows"

Categorification

Translating ontologies: "Language of Sets" to "Language of Cats"

Comparing concepts by means of their models

Example I Emptiness in $Sets^{C}$

Example II Dedekind finiteness in $Sets^{M}$ (action of monoids).

Monoid actions (M-Sets)

Let M be a monoid and A be a set.

$\mathcal{A} : A \times M \to A$ is a M-Set, iff, $\mathcal{A}(s, m \star n) = (\mathcal{A}(s, m), n)$.

Let $\mathcal{A}$ and $\mathcal{B}$ be M-Sets.

$\mathcal{F} : \mathcal{A} \to \mathcal{B}$ is a morphism in $Sets^M$, iff, $F : A \to B$
$F(\mathcal{A}(x, m)) = \mathcal{B}(F(x), m)$.

M-Sets and morphisms between M-Sets form a category and is a topos.

A motivating example

Let M be the free monoid generated by $\{m_i / i \in \mathbb{N}\}$

A=$\{a_n / n \in \mathbb{N}\}$, B=$\{b_n / n \in \mathbb{N}\}$, and $C = A \cup B$

Let $\mathcal{C} : M \times C \to C$ be the action of M on C, such that,

$$\mathcal{C}(m_n, x) = \begin{cases} a_k & \text{if } x = a_k \text{ and } k \neq n \\ b_k & \text{if } x = a_k \text{ and } k = n \\ b_k & \text{if } x = b_k \end{cases}$$

Any injective morphism $\mathcal{F} : \mathcal{C} \to \mathcal{C}$ is the identity $Id_{\mathcal{C}}$

Let $G : B \mapsto B$ be an injective function that is not bijective. The action $\mathcal{B}(m_n, b_k) = b_k$ is not Dedekind finite.

Foundational motivation

- It seems that infiniteness happens only in math;
- Infiniteness (finiteness) had to be defined (1800's);
- Dedekind (D), Kuratowski (K), Peano (P) and many other provided finiteness definitions;
- In 1924 Tarksi proved D, K and P are equivalent, using Axiom of Choice;
- This talk: What happens when computational ideas come to the scenario ?

Known definitions for finite sets

Dedekind

A is D-finite, iff, every injective function $f : A \to A$ is onto

Known definitions for finite sets

Kuratowski/Sierpinski/Russell (proof-theoretical version)

A is K-finite, iff, one can prove:

- A is empty, or;
- A is a singleton ($A = \{a\}$, for some a), or;
- $A = A_1 \cup A_2$, and A_1 and A_2 are K-finite.

Known definitions for finite sets

Kuratowski/Sierpinski/Russell (algebraic version I)

A is K-finite, iff,

- $$K(A) = \left\{ S : \begin{array}{l} \emptyset \in S \\ \forall a \in A, \{a\} \in S \\ \forall A_1, A_2 \in 2^A, \text{if } A_1, A_2 \in S \text{ then } A_1 \cup A_2 \in S \end{array} \right\}$$
- $A \in K(A)$

Known definitions for finite sets

Kuratowski/Sierpinski/Russell (algebraic version II)

A is K-finite, iff, the free semi-lattice generated by $\cup$ from $\emptyset$ and the singletons $\{a\}$ ($a \in A$), contains A

Known definitions for finite sets

R. Squire (1997)

A is Squire-finite, iff,

- Let ϕ_p be the formula $\bigvee_{i,j \leq p}(x_i = x_j)$ (there is at most p things)
- $L_p(A) = \{S : S \in 2^A \text{ and } S \models \phi_p\}$
- $A \in L_p(A)$

Known definitions for finite sets

Peano

A is Peano-finite, iff, $A \simeq [p] = \{1, \ldots, p\} \subseteq \mathbb{N}$

Known definitions for finite sets

Remember !!!

- Every “finite” set is Dedekind, Peano, Kuratowksi, Squire and etc, finite
- The above notions are not equivalence without the Axiom of Choice

Usual assumptions on (classical) computational models

prog **The size of any program is finite**

data **The amount of resource (memory/input) used is (stepwise) finite**

time **Meaningful computations should stop (elapsed time is finite)**

Set-theoretical description of Turing Machines

Syntax1 $\langle Q, \Sigma, \{q_o\}, \{q_f\}, \delta \rangle$, and, $\delta \subseteq 2^{\Sigma \times Q \times \{\leftarrow, \rightarrow\} \times \Sigma \times Q}$

Syntax2 If Q and Σ are finite then δ is finite too.

Seman1 $\hat{\delta} : \Sigma^\star \times Q \longrightarrow Q$

Seman2 $\Sigma^\star$ and $\hat{\delta}$ come from Σ and δ by induction/recursion.

Turing Machines expressed in a local language

- Fix a finiteness definition $fin(X)$
- Use $fin(X)$,Syntax1,Syntax2_{fin}, Seman1_{fin} and Seman2_{fin} to define $TMComputable(f)$

$\hat{\delta}$ without *NNO*

- $T : Pos \longrightarrow \Sigma$
- $f_\delta : \mathcal{P}(Q \times Pos \times T_{fin}) \longrightarrow \mathcal{P}(Q \times Pos_{fin} \times Q)$, such that $T_{fin} = \{h / fin(\{p \in Pos : h(p) \neq \not{b}\})\}$
- $X = \mathcal{P}(Q \times Pos \times T_{fin})$, $St(X) = \{S \subseteq X : f_\delta(S) \subseteq S\}$
- $i : St(X) \hookrightarrow \mathcal{P}(X)$ has left adjoint $\mathcal{O} : \mathcal{P}(X) \to St(X)$, $\mathcal{O}(Z) = \bigcap\{S \in St(X) : Z \subseteq S\}$
- $\mathcal{O} \circ i$ is a closure operator, then $\hat{\delta} = \mathcal{O} \circ i$
- $f = \{\langle x^{\Sigma^\star}, y^{\Sigma^\star}\rangle / < q, z^{Pos}, y^T > \in \hat{\delta} \circ < q_0^Q, o^{Pos}, x^T >$ and $final(q)\}$

Finiteness in Computation

Depending on $fin(X)$, one may have:

- Infinitely long deterministic programs (externally).
- Infinitely branching non-deterministic programs. (externally)
- Finite computations with subsequences of infinite computations (externally)
- Internally finite !!!

Finiteness in Computation

Depending on $fin(X)$, one may have:

- Infinitely long deterministic programs (externally).
- Infinitely branching non-deterministic programs. (externally)
- Finite computations with subsequences of infinite computations (externally)
- Internally finite !!!

Examples

In $Sets^M$, G a particular free group. $fin(A) = Dedekind(A)$ in $Sets^G$

- $Dedekind(A) = D(A) = \forall f \in A^A(Mono(f) \Rightarrow Iso(f))$
- $Sets^G \models \exists W(D(W) \wedge \neg D(\mathcal{P}(W))$
- A T.M. with states in W can be a non-standard computational model with a D-infinitely long program, or a D-infinitely branching non-deterministic behavior.

Examples

In $Sets^{0\to 1}$ with $fin(A) = Kuratowski(A)$

$Kuratowski(A) = K(A) =$

$$\forall z \in \Omega^{\Omega^A}([0 \to A] \in z \wedge \forall a \in A(\{a\} \in z) \wedge \forall y \in \Omega^A \forall y' \in \Omega^A((y \in z \wedge y' \in z) \Rightarrow (y \cup y') \in z) \Rightarrow [id_A] \in z)$$

- $Sets^{0\to 1} \models \exists W(K(W) \wedge \exists V(V \subseteq W \wedge \neg K(V)))$
- A T.M. with states in W can be a non-standard computational model able to K-finitely compute on K-infinitely long transitions.

Examples

Externally

- A non-standard computational model with an infinitely long program, or an infinitely branching non-deterministic behavior.
- A non-standard computational model able to finitely compute on infinitely long transitions.

Further investigation

- How this relates to physically based non-standard computation ?
- Does anyone needs any physical assumption to provide non-standard (hypercomputational) models of computing ? General Relativity Topos and Malament-Hogarth relativistic computer can be obtained on a purely foundational way ?
- How far can we go without Natural Numbers Object ?? (Benabou on Topos with NNO, Tenenbaum's Theorem; P.Freyd Topos Numerology !!)
- Relationship to the Effective Topos ?

Thank You!

PUC RIO

The Effective Topos [Hyland, 1982]

A Topos holding:

(1) Bool Its internal logic proves $\phi \vee \neg\phi$;

(2) Comp Every morphism is “computable” ;

(3) NNO It has a Natural Numbers Object.

proves that the halting problem is “computable”.

The Effective Topos [Hyland, 1982]

A Topos holding:

(1) Bool Its internal logic proves $\phi \vee \neg\phi$;

(2) Comp Every morphism is "computable" ;

(3) NNO It has a Natural Numbers Object.

proves that the halting problem is "computable".

Alternatives

(1+3) $\Rightarrow$ *Sets* is such topos. FOL + Arith

(1+2) $\Rightarrow$ *FinSets* is such topos. FiniteFOL

(2+3) $\Rightarrow$ Several toposes are known: Effective Topos, Recursive Topos, etc. ILogic. Even Martin-Löf type theory is here (cannot be a topos !!).

Other approaches on hypercomputation

- Analogic computation;
- Smale-Blum holomorphic computation;
- Infinite Turing machines;
- Physically based computation;

Categorification

Definition

Categorification is the process of finding category-theoretic analogs of set-theoretic concepts by replacing sets with categories, functions with functors, and equations between functions by natural isomorphisms between functors, which in turn should satisfy certain equations of their own, called “coherence laws”.

[Baez98,Crane96]

Categorification

A Category is composed by:

- A collection Obs of objects.
- For each pair of objects $A, B \in Obs$ there is a collection $Hom(A, B)$ of morphisms (arrows) from A to B.
- A composition operation $\circ : Hom(A, B) \times Hom(B, C) \to Hom(A, C)$
- For each object A, $Id_A \in Hom(A, A)$.
- $f \circ Id_A = Id_B \circ f = f$.
- $f \circ (g \circ h) = (f \circ g) \circ h$.

Category Theory defines its concepts and constructions by means of Universal Properties providing them unique up to isomorphims.

Categorification

Example

FinSets is the categorification of the (Algebra of) Natural Numbers

$$x \times (y + z) = x \times y + x \times z$$

$$x \times 0 = 0$$

$$\eta_{x,y,z} : x \times (y + z) \rightleftharpoons x \times y + x \times z : \eta^{-1}_{x,y,z}$$

$$\epsilon_x : x \times 0 \rightleftharpoons 0 : \epsilon_x^{-1}$$

Decategorification

Now, given a category C, we may “decategorify” it by forgetting about the morphisms and pretending that isomorphic objects are equal. We are left with a set (or class) whose elements are isomorphism classes of objects of C. This process is dangerous, because it destroys useful information. It amounts to forgetting which road we took from x to y, and just remembering that we got there. Sometimes this is actually useful, but most of the time people do it unconsciously, out of mathematical naivete. We write equations, when we really should specify isomorphisms. “Categorification” is the attempt to undo this mistake. Like any attempt to restore lost information, it not a completely systematic process. return

In general there is more than one way to “categorificate” a set-theoretic concept

Disjoint Union

$$\begin{array}{rcl} A+B & = & (\{a\}\times A)\cup(\{b\}\times B) \\ & \sim & (\{0\}\times A)\cup(\{1\}\times B) \\ & \vdots & \\ & \sim & (\{z\}\times A')\cup(\{w\}\times B') \\ \textit{whenever} & & B\sim B' \textit{ and } B\sim B' \end{array}$$

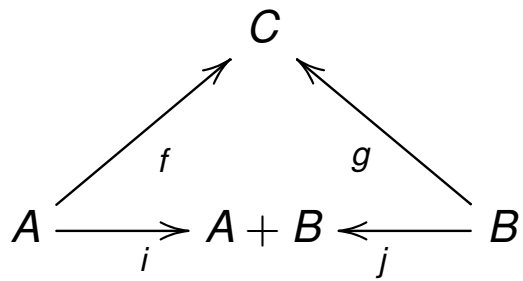
return

Categorical Co-product

$$A \xrightarrow[i]{} A + B \xleftarrow[j]{} B$$

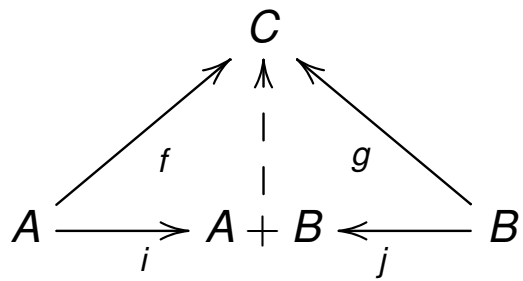
return

Categorical Co-product



return

Categorical Co-product



return