A Parallel Corpus for Autoformalisation

Anthony Bordg

University of Cambridge

IRIF, November 2022

Machine Learning for Translation

Machine learning is a methodology for leveraging data to improve machine performance on various types of tasks.

One such type of tasks is machine translation between two languages.

Three Stages of Machine Translation

- Already here: machine translation between two natural languages (see Google Translate).
- Coming soon: machine translation of a natural language specification into a programming language, *e.g.* Python (see Codex).
- Someday? "autoformalisation", *i.e.* machine translation of informal mathematical proofs into formal ones with minimal input from humans.
  - Szegedy, *A Promising Path Towards Autoformalisation and General Artificial Intelligence*, CICM 2020
  - Wang *et al*, *Exploration of neural machine translation in autoformalization of mathematics in Mizar*, CPP, 2020
  - Wu *et al*, *Autoformalization for Neural Theorem Proving*, AITP, 2022

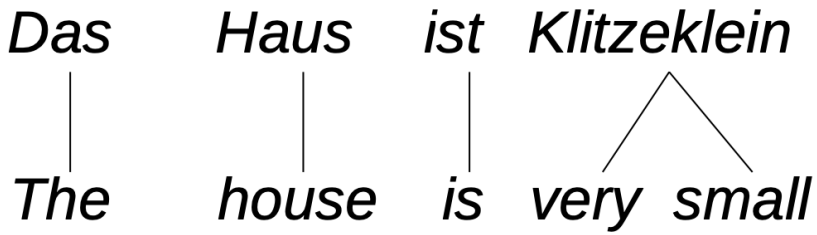
Parallel Corpora

Parallel Corpora are key resources for performing machine translation.
Roughly speaking, a parallel corpus is a triple (S, T, P) where

- S is the source language, *e.g.* German,
- T is the target language, *e.g.* English,
- P is a set of paired "parallel" scripts, *i.e.* it is a pairing of pieces of text written in S and their translations in T.

Alignment in Parallel Corpora

For better machine translation, the pairing in a parallel corpus (S, T, P) can be further refined.
Given a pair $(p_S, p_T) \in P$, we can map tokens in p_S to tokens in p_T to improve our pairing P with tokens alignments within parallel scripts.



The Isabelle Parallel Corpus

The Isabelle Parallel Corpus (IPC) is a parallel corpus for autoformalisation with LaTeX as the source language and the formal system Isabelle/HOL as the target language.
The IPC is thought as a community effort to create a database of aligned pairs of informal artefacts and formal ones, where an artefact is either a definition, a statement or the proof of a given statement.
By "informal" artefacts we mean artefacts as written by mathematicians in LaTeX and by "formal" artefacts we mean artefacts translated by expert users of Isabelle/HOL.

Summary

In this talk, we will present the infrastructure around the IPC, the current content of the corpus and its future extensions.

Outline

We will discuss the following points:

1. the infrastructure around the IPC,
2. the current content of the IPC,
3. the scalability of the IPC,
4. our future plans.

The IPC's Pairing Tool

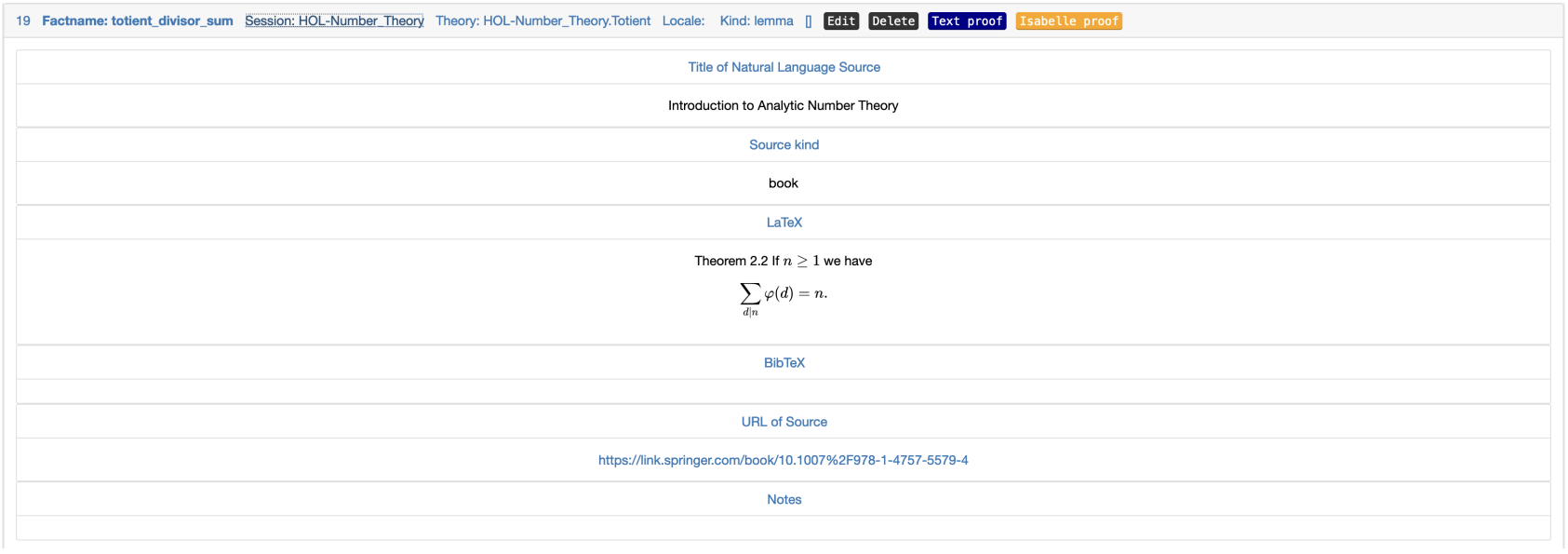
Our pairing tool is built on top of the SErAPIS search engine for Isabelle.[1] Through an interface, one can search (with words like in Google) for a formal artefact in Isabelle/HOL's libraries and its Archive of Formal Proofs. Once the desired artefact is found, one can create a new entry in the database by recording its natural language counterpart using LaTeX and possibly a BibTeX reference and an URL for the source. If an entry has already been created for a formal artefact, this entry can be edited. This way, we're building a parallel corpus made of pairs of natural and Isabelle artefacts.

Check out our article: B., Stathopoulos and Paulson, *A Parallel Corpus for Natural Language Machine Translation to Isabelle*, CICM, 2022.

[1] Stathopoulos, Koutsoukou-Argyraki, Paulson, *SErAPIS: A concept-oriented search engine for the Isabelle libraries based on natural language*, Online proceedings of the Isabelle workshop, 2020

An Entry of the IPC

Our database for the IPC takes the form of a collection of collapsible information cards.


19 **Factname: totient_divisor_sum** Session: HOL-Number_Theory Theory: HOL-Number_Theory.Totient Locale: Kind: lemma [] Edit Delete Text proof Isabelle proof

Title of Natural Language Source

Introduction to Analytic Number Theory

Source kind

book

LaTeX

Theorem 2.2 If $n \geq 1$ we have

$$\sum_{d|n} \varphi(d) = n.$$

BibTeX

URL of Source

https://link.springer.com/book/10.1007%2F978-1-4757-5579-4

Notes

Figure: Visit our website: `https://behemoth.cl.cam.ac.uk/ipc/`.

Pseudo-Alignment Between Textual and Formal Proofs

Sentence	Text
①	$\underbrace{\text{If } n \geq 1}_{I}$ we have $$\overbrace{\sum_{d\mid n} \varphi(d) = n.}^{\alpha}$$
②	Let S denote the set $\{1, 2, \ldots, n\}$.
③	We distribute the integers of S into disjoint sets as follows.
④	For each divisor d of n, let $$A(d) = \{k : (k, n) = d,\ 1 \leq k \leq n\}.$$
⑤	That is, $A(d)$ contains those elements of S which have the gcd d with n.
⑥	The sets $A(d)$ form a disjoint collection whose union is S.
⑦	Therefore if $f(d)$ denotes the number of integers in $A(d)$ we have $$\sum_{d\mid n} f(d) = n. \quad (1)$$
⑧	But $(k, n) = d$ if and only if $(k/d, n/d) = 1$, and $0 < k \leq n$ if and only if $0 < k/d \leq n/d$.
⑨	Therefore, if we let $q = k/d$, there is a one-to-one correspondence between the elements in $A(d)$ and those integers q satisfying $0 < q \leq n/d$, $(q, n/d) = 1$.
⑩	The number of such q is $\varphi(n/d)$.
⑪	Hence $\underbrace{f(d) = \varphi(n/d)}_{\beta}$ and (1) becomes $$\underbrace{\sum_{d\mid n} \varphi(n/d) = n.}_{\gamma}$$
⑫	But this is equivalent to the statement $\sum_{d\mid n} \varphi(d) = n$ because when d runs through all divisors of n so does n/d.
⑬	This completes the proof.

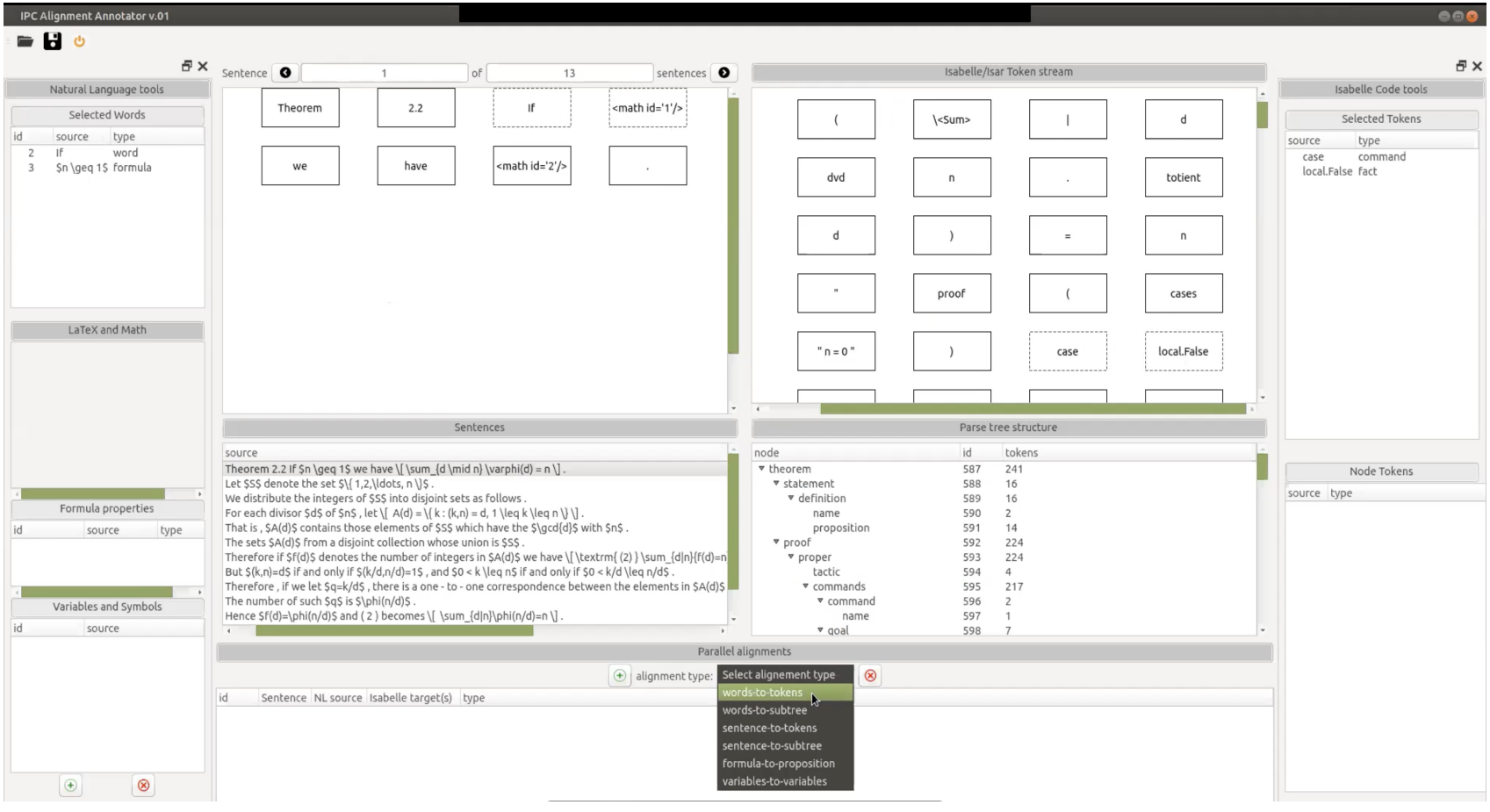
Theory: HOL-Number_Theory.Totient

Sentence		Isabelle command
α	1	`lemma totient_divisor_sum: "(∑d | d dvd n. totient d) = n"`
		`proof (cases "n = 0")`
I	2	`  case False`
	3	`  hence "n > 0" by simp`
④	4	`  define A where "A = (λd. {k∈{0<..n}. gcd k n = d})"`
β	5	`  have *: "card (A d) = totient (n div d)" if d: "d dvd n" for d`
		`    using ‹n > 0› and d unfolding A_def by (rule card_gcd_eq_totient)`
	6	`  have "n = card {1..n}" by simp`
② ⑥	7	`  also have "{1..n} = (⋃d∈{d. d dvd n}. A d)" by safe (auto simp: A_def)`
(1) in ⑦	8	`  also have "card ... = (∑d | d dvd n. card (A d))"`
		`    using ‹n > 0› by (intro card_UN_disjoint) (auto simp: A_def)`
γ	9	`  also have "... = (∑d | d dvd n. totient (n div d))" by (intro sum.cong refl *) auto`
⑫	10	`  also have "... = (∑d | d dvd n. totient d)" using ‹n > 0›`
		`    by (intro sum.reindex_bij_witness[of _ "(div) n" "(div) n"]) (auto elim: dvdE)`
	11	`  finally show ?thesis ..`
⑬	12	`qed auto`

Figure: An example of a many-to-many mapping.

Building Tools for Alignment

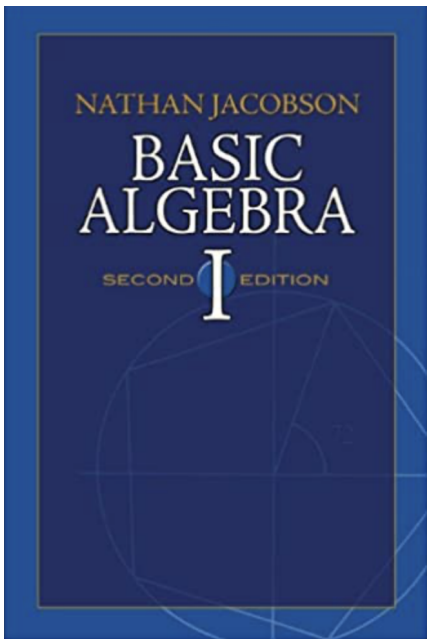
Tools have been built to fine-tune the alignment, *e.g.* LaTeX to Isabelle/HOL tokens, with parsers and automated extraction of variables and symbols.



Tracking Dependencies

A suite of graph analysis tools has been developed for modelling the relationships between formal artefacts. Given a formal definition or a formal statement, one will be able to track the definitions used in them. Given a formal proof, one will be able to track the facts called within this proof. Tracking dependencies allows to understand which artefacts could be paired within a parallel corpus to better the alignment between informal and formal artefacts.

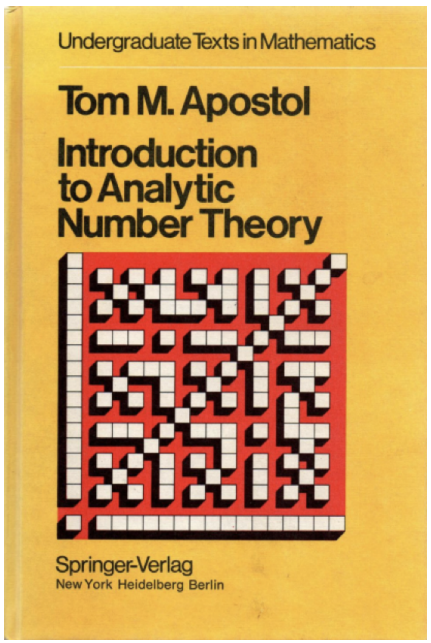
IPC's Table of Contents



Some parts of the first three chapters of Jacobson's *Basic Algebra* textbook have been formalised in Isabelle, covering monoids, groups and rings.[2] This formal development includes 175 theorems. We have paired the formal definitions and statements with their informal counterparts in the IPC.

[2] Clemens Ballarin, *Exploring the Structure of an Algebra Text with Locales*, J Autom Reasoning, 2020

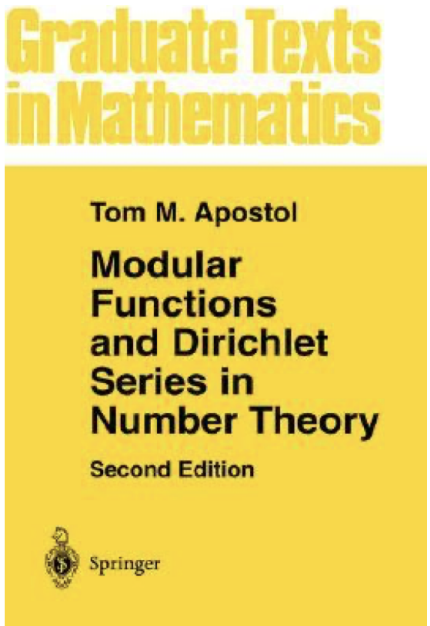
IPC's Table of Contents (continued)



Apostol's
Introduction to Analytic Number Theory
has been largely formalised in Isabelle/HOL ($\sim$ 80%, mainly by Manuel Eberl).[3]
It represents $\sim$ 30,000 lines of code. We have paired the formal definitions and statements with their informal counterparts in the IPC.

[3] Manuel Eberl, *Nine Chapters of Analytic Number Theory in Isabelle/HOL*, ITP 2019

A Future Extension of the IPC



With Manuel Eberl, Wenda Li and Larry Paulson, we're formalising Apostol's follow-up textbook. The definitions and statements in all eight chapters have been formalised. Proofs in chapters 1,2,3 and 7 have been formalised and also a few bits and bobs in chapters 4 and 6 ($\sim$ 55,000 lines of code as of November 2022). This project features elliptic functions, Eisenstein series, Ramanujan's τ function, modular forms ... Since its inception in 2021, the project has been aimed at extending the IPC.

Future Extension and Usage of the IPC

Extension: add to the IPC all the pen-and-paper proofs from Apostol's undergraduate textbook.

Autoformalisation of advanced number theory may be a far away goal, but more approachable goals for which the IPC could be useful include:

- developing natural language search for Isabelle's libraries
- "informalisation", *i.e.* generating natural language descriptions of Isabelle artefacts.

The IPC, a Scalable Database?

To perform autoformalisation on many parts of mathematics at various levels of abstraction, one needs a target language sufficiently expressive for all kinds of mathematical objects.
Isabelle/HOL is based on Church's simple type theory, but dependently-typed systems (Agda, Coq, Lean ...) are more expressive than simply-typed systems.
Is it possible to formalise more abstract mathematics into Isabelle/HOL, so that we could be able to collect a wide array of data points in the IPC?

Can Schemes be formalised in Isabelle/HOL?

Schemes are abstract spaces introduced in 1960 by Alexander Grothendieck and studied in algebraic geometry.

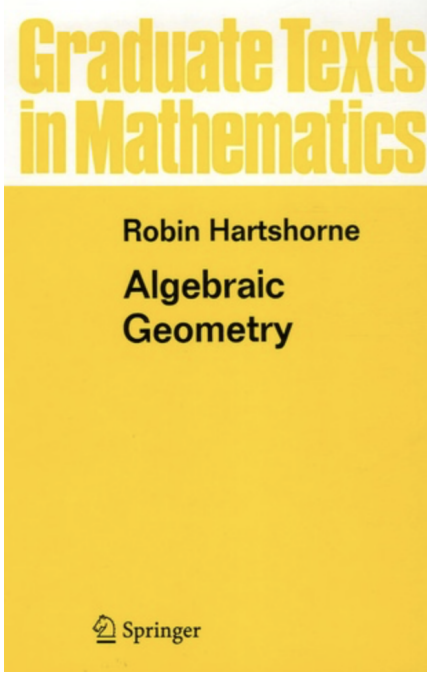
Schemes have been recently formalised in Lean.

Following this formalisation, the mathematician Kevin Buzzard challenged all the other interactive theorem provers to formalise schemes.

Following Hartshorne's textbook *Algebraic Geometry*, Isabelle/HOL met the challenge in 2021.

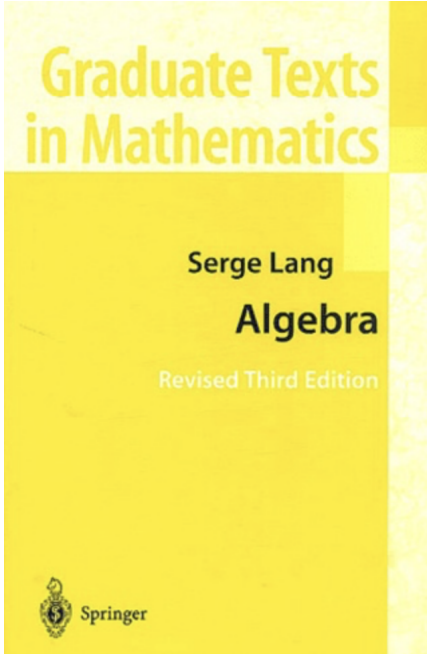
The details are in B., Paulson and Li, *Simple Type Theory is not too Simple: Grothendieck's Schemes Without Dependent Types*, Experimental Mathematics, 2022.

IPC's Table of Contents II



Schemes are now in the IPC.
For machine learning, the definition alone wouldn't be useful. We had to pair all the formal prerequisites (affine schemes, sheaves and presheaves of rings . . .) with their informal counterparts, covering some material in the chapter II of Hartshorne's textbook.

IPC's Table of Contents II (continued)



We took
also the opportunity for pairing the prerequisites in algebra (prime and maximal ideals . . .)
not covered by the formal development from Jacobson's *Basic Algebra* textbook, bridging the gap in the IPC with this last development.

Another Case Study: Strict Omega Categories

In recent years, higher-dimensional structures, such as omega-groupoids and omega-categories, have been formalised in some dependently-typed systems like Agda and Coq (both extended by a strict equality).

These structures seem to make use of dependent types in a decisive way. In particular, the aforementioned achievements are based on type-theoretic reformulations of these structures.

Can strict omega-categories be formalised without dependent types?

Yes, we have formalised strict omega-categories in Isabelle/HOL following Leinster's book *Higher Operads, Higher Categories*.

We report on this formalisation in our preprint: B. and Doña Mateo, *Encoding Dependently-Typed Constructions into Simple Type Theory*, 2022.

Strict Omega-Categories

Definition (Strict Omega-Categories)

A strict ω-category is a globular set X equipped with

- a function $\circ_n : X_m \times_{X_n} X_m \to X_m$ for all natural numbers m and n with $n < m$; we write $\circ_n(y, x)$ as $y \circ_n x$ and call it a composite of x and y
- a function $i : X_n \to X_{n+1}$ for each natural number n; we write $i(x)$ as 1_x and call it the identity of x,

satisfying the following axioms

1. (sources and targets of composites) if $n < m$ and $(y, x) \in X_m \times_{X_n} X_m$ then

$$\left.\begin{aligned} s(y \circ_n x) &= s(x) \\ t(y \circ_n x) &= t(y) \end{aligned}\right\} \text{ if } n = m - 1$$

and

Strict Omega-Categories (continued)

Definition (continued)

$$\left.\begin{aligned} s(y \circ_n x) &= s(y) \circ_n s(x) \\ t(y \circ_n x) &= t(y) \circ_n t(x) \end{aligned}\right\} \text{if } n \leq m-2$$

2. (sources and targets of identities) if $x \in X_n$ then

$$s(1_x) = x = t(1_x)$$

for all natural number n

3. (associativity) if $n < m$ and $x, y, z \in X_m$ with $(z, y), (y, x) \in X_m \times_{X_n} X_m$ then

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

Strict Omega-Categories (continued)

Definition (continued)

4. (identities) if $n < m$ and $x \in X_m$ then

$$i^{m-n}(t^{m-n}(x)) \circ_n x = x = x \circ_n i^{m-n}(s^{m-n}(x)),$$

where $i^{m-n} \colon X_n \to X_m$ denotes the $(m-n)$-th iterate of i

5. (nullary interchange) if $q < p$ and $(y, x) \in X_p \times_{X_q} X_p$ then $1_y \circ_q 1_x = 1_{y \circ_q x}$

Strict Omega-Categories (end)

Definition (end)

6. (binary interchange) if $q < p < m$ and $x, x', y, y' \in X_m$ with

$$(y', y), (x', x) \in X_m \times_{X_p} X_m$$

and

$$(y', x'), (y, x) \in X_m \times_{X_q} X_m$$

then

$$(y' \circ_p y) \circ_q (x' \circ_p x) = (y' \circ_q x') \circ_p (y \circ_q x)$$

Strict Omega-Categories Formalised

```
locale strict_omega_category = globular_set +
  fixes comp :: "nat ⇒ nat ⇒ 'a ⇒ 'a ⇒ 'a"
    and i' :: "nat ⇒ nat ⇒ 'a ⇒ 'a"
  assumes i'_fun: "n ≤ m ⟹ i' m n ∈ X n → X m"
    and i'_n_n: "i' n n = id"
    and comp_fun: "is_composable_pair m n x' x ⟹ comp m n x' x ∈ X m"
    and s_comp_Suc: "is_composable_pair (Suc m) m x' x ⟹ s m (comp (Suc m) m x' x) = s m x"
    and t_comp_Suc: "is_composable_pair (Suc m) m x' x ⟹ t m (comp (Suc m) m x' x) = t m x'"
    and s_comp: "⟦is_composable_pair (Suc m) n x' x; n < m⟧ ⟹
          s m (comp (Suc m) n x' x) = comp m n (s m x') (s m x)"
    and t_comp: "⟦is_composable_pair (Suc m) n x' x; n < m⟧ ⟹
          t m (comp (Suc m) n x' x) = comp m n (t m x') (t m x)"
    and s_i: "x ∈ X n ⟹ s n (i' (Suc n) n x) = x"
    and t_i: "x ∈ X n ⟹ t n (i' (Suc n) n x) = x"
    and comp_assoc: "⟦is_composable_pair m n x' x; is_composable_pair m n x'' x'⟧ ⟹
           comp m n (comp m n x'' x') x = comp m n x'' (comp m n x' x)"
    and i_comp: "⟦n < m; x ∈ X m⟧ ⟹ comp m n (i' m n (t' m n x)) x = x"
    and comp_i: "⟦n < m; x ∈ X m⟧ ⟹ comp m n x (i' m n (s' m n x)) = x"
    and bin_interchange: "⟦q < p; p < m;
          is_composable_pair m p y' y; is_composable_pair m p x' x;
          is_composable_pair m q y' x'; is_composable_pair m q y x⟧ ⟹
          comp m q (comp m p y' y) (comp m p x' x) = comp m p (comp m q y' x') (comp m q y x)"
    and null_interchange: "⟦q < p; is_composable_pair p q x' x⟧ ⟹
          comp (Suc p) q (i' (Suc p) p x') (i' (Suc p) p x) = i' (Suc p) p (comp p q x' x)"
```

Did we face some limitations of Isabelle's type system?

Breaking Out of Limitations

Let's backpedal and discuss globular sets, a higher-dimensional analogue of a directed graph and a prerequisite for strict omega-categories.

Definition (globular sets)

A globular set X is a diagram

$$\cdots \underset{t}{\overset{s}{\rightrightarrows}} X_n \underset{t}{\overset{s}{\rightrightarrows}} X_{n-1} \underset{t}{\overset{s}{\rightrightarrows}} \cdots \underset{t}{\overset{s}{\rightrightarrows}} X_0$$

of sets and maps satisfying the so-called globular identities

$$s(s(x)) = s(t(x))$$
$$t(t(x)) = t(s(x))$$

for all element $x \in X_m$ and all natural number $m \geq 2$.

Breaking Out (continued)

```
locale globular_set =
  fixes X :: "nat ⇒ 'a set" and s :: "nat ⇒ 'a ⇒ 'a" and t :: "nat ⇒ 'a ⇒ 'a"
  assumes s_fun: "s n ∈ X (Suc n) → X n"
    and   t_fun: "t n ∈ X (Suc n) → X n"
    and  s_comp: "x ∈ X (Suc (Suc n)) ⟹ s n (t (Suc n) x) = s n (s (Suc n) x)"
    and  t_comp: "x ∈ X (Suc (Suc n)) ⟹ t n (s (Suc n) x) = t n (t (Suc n) x)"
```

We can't have in Isabelle/HOL an indexed family of type variables (here, a family indexed by the natural numbers), hence we used a single type variable `'a` to type all the carriers of the sets X_n.

Is it a problem?

Breaking Out (continued)

As a sanity check, we had to provide a standard example of strict omega-category, so we decided to prove that pasting diagrams form a strict omega-category. For the underlying globular set pd of pasting diagrams, there are two equivalent definitions and only one of them can be easily formalised given the above constraint.
First definition: the underlying globular set pd of pasting diagrams is defined inductively as follows,

- $\mathrm{pd}_0 = 1$
- $\mathrm{pd}_{n+1} = (\mathrm{pd}_n)^*$

where $(\_)^*$ is the so-called free monoid functor on Set. This means that pd_{n+1} is the set of finite lists of elements in pd_n.

Assuming that the unique element of the singleton 1 has type `'a`, pd_0 should have type `'a set`, while pd_1 (*resp.* pd_2 ...) should have type `('a list) set` (*resp.* `(('a list) list) set` ...).

Breaking Out (end)

Second definition:

define pd_n as the set of trees of height less than n.

Take 'a := tree, then all the types of the carriers for the pd_n can be embedded in the single type tree set.

The Road Less Travelled

Even though Isabelle/HOL is based on simple type theory, locales can be used as a substitute for dependent types.
Isabelle/HOL could be well suited as a target language for autoformalisation.
Isabelle/HOL has relatively large libraries, a simple yet expressive logic, a vernacular for structured statements/proofs and powerful automation.
Can Isabelle/HOL's simple type theory ease the work of an autoformalisation system that would be able to translate at least partially an informal proof into a formal one and then leverage Isabelle's powerful automation?

Parallel Corpora as a Service

Stop thinking in terms of aligned pairs, think in terms of aligned tuples $(a_0 \in L_0, \ldots, a_{n-1} \in L_{n-1})$, where $n \geq 5$ and $L_0 = \LaTeX, L_1 = \mathit{Agda}, L_2 = \mathit{Coq}, L_3 = \mathit{Isabelle}, L_4 = \mathit{Lean} \ldots$

Our methodology and a suite of similar tools could be used for various other competitive target languages (Agda, Coq, Lean ...).

Take-Home Message

- We promote an integrative approach to autoformalisation.
- Autoformalisation is not restricted to mathematics and we should cover the entire scope of formal verification (algorithms, protocols . . .).

Thank You