Overview of some recent Deducteam works on proof systems interoperability

Frédéric Blanqui





Deducteam (in May 2026)

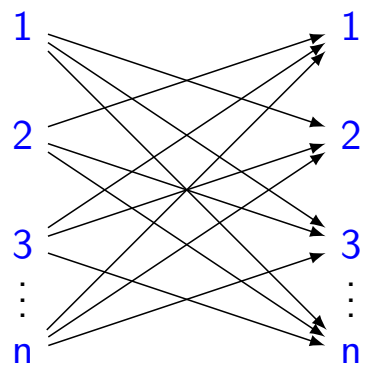


Proof systems interoperability: why?

- avoid work duplication
- independent proof checking
- certify ATP/SMT proofs
- reverse mathematics

Proof systems interoperability: difficulties?

- proof assistants greatly differ in their syntax and underlying logic, and are constantly evolving
- in contrast to programming languages, proof systems have different expressive power
- developing and maintaining translators between all proof assistants is not sustainable

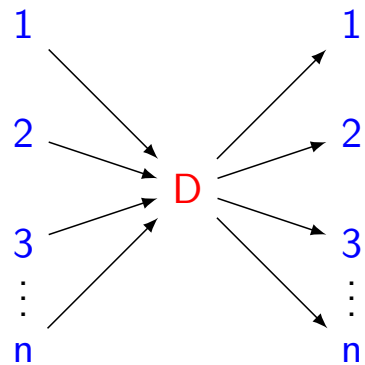


$n(n-1)$ translators

Deducteam strategy: use a logical framework

a **logical framework** is a formal language in which one can express deduction rules and proofs of different systems

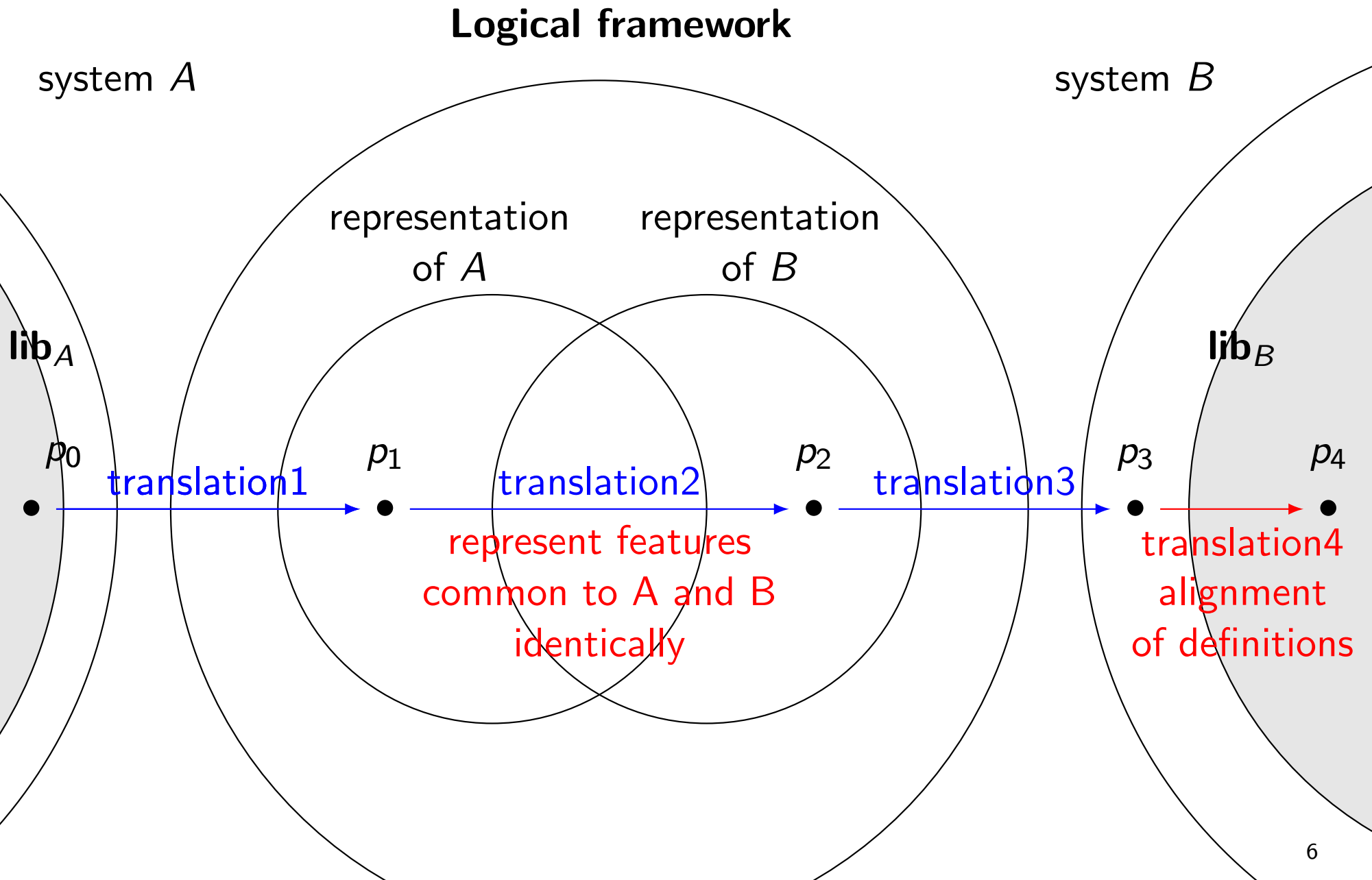
e.g. predicate logic, pure type systems, $\lambda\Pi$-calculus, **Dedukti**



$2n$ translators

- express proofs of different systems in Dedukti
- develop translators between those systems and Dedukti
- check that definitions used in different systems are equivalent

Proof systems interoperability: Deducteam strategy



The $\lambda\Pi$ logical framework

the $\lambda\Pi$-calculus is the simplest type system combining both simple and dependent types [de Bruijn, 1968]

$$\frac{\Gamma \vdash A \ominus \mathsf{Type} \quad \Gamma, x : A \vdash B \ominus \mathsf{Type}}{\Gamma \vdash (\Pi x : A, B) \ominus \mathsf{Type}} \qquad (\Pi x : A, B \text{ is } A \to B \text{ if } x \notin B)$$

$$\frac{\Gamma \vdash t \ominus (\Pi x : A, B) \quad \Gamma \vdash u \ominus A}{\Gamma \vdash tu \ominus B[x/u]}$$

can encode different logics [Harper, Honsell, Plotkin, LICS'87]

The Dedukti logical framework
=
$\lambda\Pi$ modulo some equational theory E

- Dedukti is like the $\lambda\Pi$-calculus but on equivalence classes of types modulo some equational theory E:

$$\frac{\Gamma \vdash t : A \quad A =_E B}{\Gamma \vdash t : B}$$

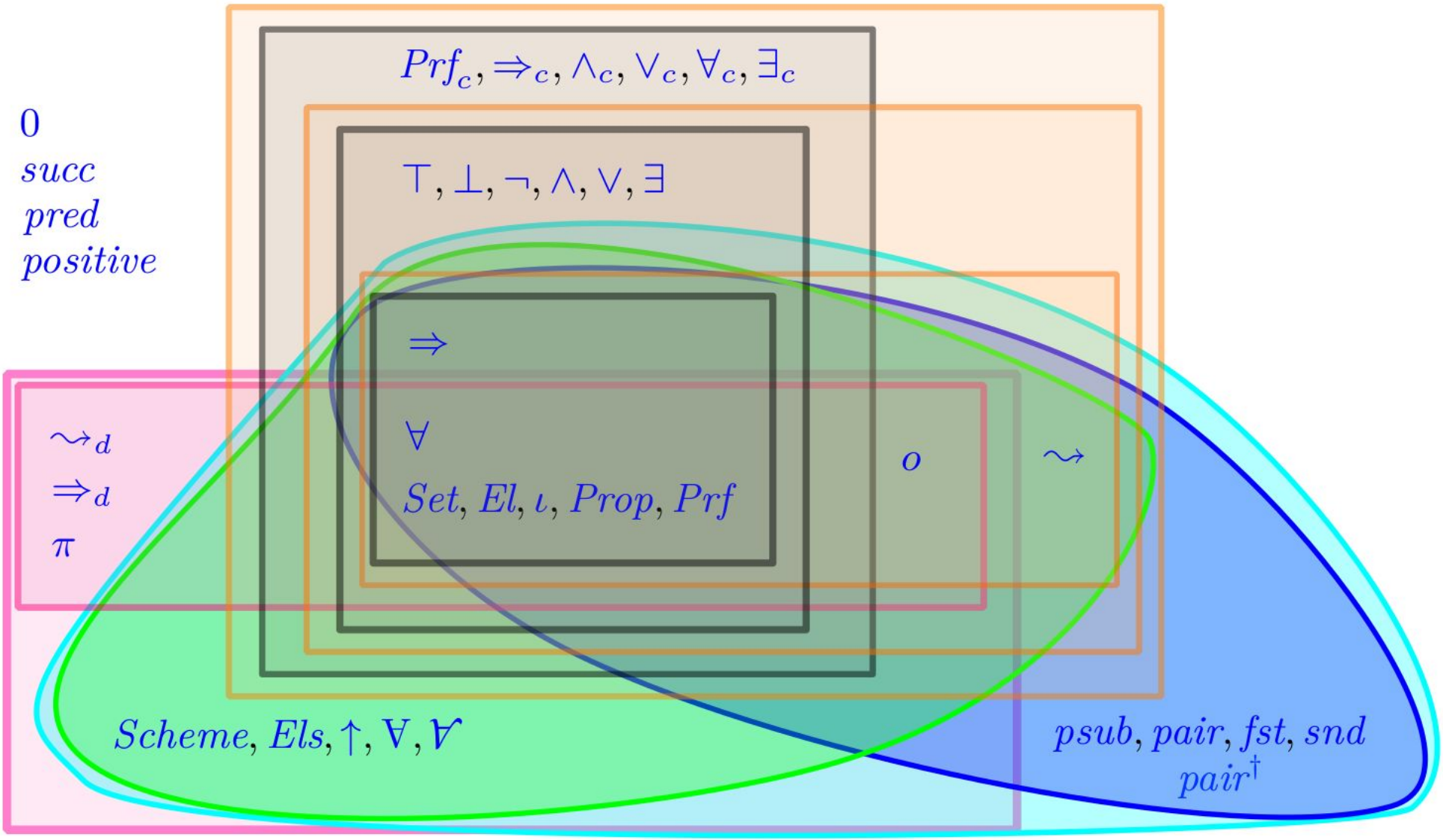
  equations allow more shallow encodings (systems with implicit computations can be encoded in $=_E$)

- a Dedukti theory is a set of typed constants together with a set of typed equations E (2nd-order generalized algebraic theory)

- there exist Dedukti theories for many logics from (fragments of) first-order logic, arithmetic, Zermelo set theory, to higher-order logic, pure type systems,... [Dowek *et al*, years 2000]

The modular Dedukti theory U and its sub-theories

[B., Dowek, Grienenberger, Hondet, Thiré, 2021]



Example: Dedukti theory for first-order logic (FOL)

constants:

I : Type — type of FO terms

$f : I \to \ldots \to I \to I$ — for each function symbol f of arity n

$Prop$: Type — type of FO formulas

$P : I \to \ldots \to I \to Prop$ — for each predicate symbol P of arity n

$\top : Prop$, $\neg : Prop \to Prop$, $\forall : (I \to Prop) \to Prop$, ...

$Prf : Prop \to$ Type — [Curry-de Bruijn-Howard]

equations:

$$Prf\,(A \Rightarrow B) = Prf\ A \to Prf\ B \qquad \text{[Brouwer-Heyting-Kolmogorov]}$$
$$Prf\,(\forall A) = \Pi x : I, Prf\,(A\,x)$$
$$Prf\,(A \wedge B) = \Pi C : Prop, (Prf\ A \to Prf\ B \to Prf\ C) \to Prf\ C$$
$$Prf\,\bot = \Pi C : Prop, Prf\ C$$
$$Prf\,(\neg A) = Prf\,A \to Prf\,\bot$$

...

theorem: π proof of A in FOL $\Leftrightarrow |\pi| \circ Prf\,|A|$ in Dedukti

Dedidability of type-checking in a Dedukti theory ?

use equations from left to right only, *i.e.* rewriting

ensure confluence, subject-reduction, and termination

possibly adding new rewrite rules (Knuth-Bendix completion)

Available checkers for oriented Dedukti theories

- https://github.com/Deducteam/Dedukti (in OCaml)
  historical checker [Saillard *et al*, 2015]
- https://github.com/01mf02/kontroli-rs (in Rust)
  parallel checker [Färber, 2022]
- https://github.com/Deducteam/lambdapi (in OCaml)
  extension of Dedukti with its own syntax, implicit arguments, coercions, unification rules, tactics, local confluence checker, external provers call, LSP server, VSCode/Emacs plugins, ...

  checkout the version 4.0.0 just released!

  See https://lambdapi.readthedocs.io/

  the next release will include type classes using ELPI [Gontard, 2026]

FOL theory in Lambdapi syntax

```
// sorts
constant symbol Set : TYPE;
constant symbol ι : Set;
injective symbol τ : Set → TYPE;
symbol el a : τ a; // we assume that sets are non empty

// propositions
constant symbol Prop : TYPE;
injective symbol π : Prop → TYPE; // Curry-de Bruijn-Howard

constant symbol ∧ : Prop → Prop → Prop;
notation ∧ infix right 7;
constant symbol ∧ᵢ [p q] : π p → π q → π (p ∧ q);
symbol ∧ₑ1 [p q] : π (p ∧ q) → π p;
symbol ∧ₑ2 [p q] : π (p ∧ q) → π q;

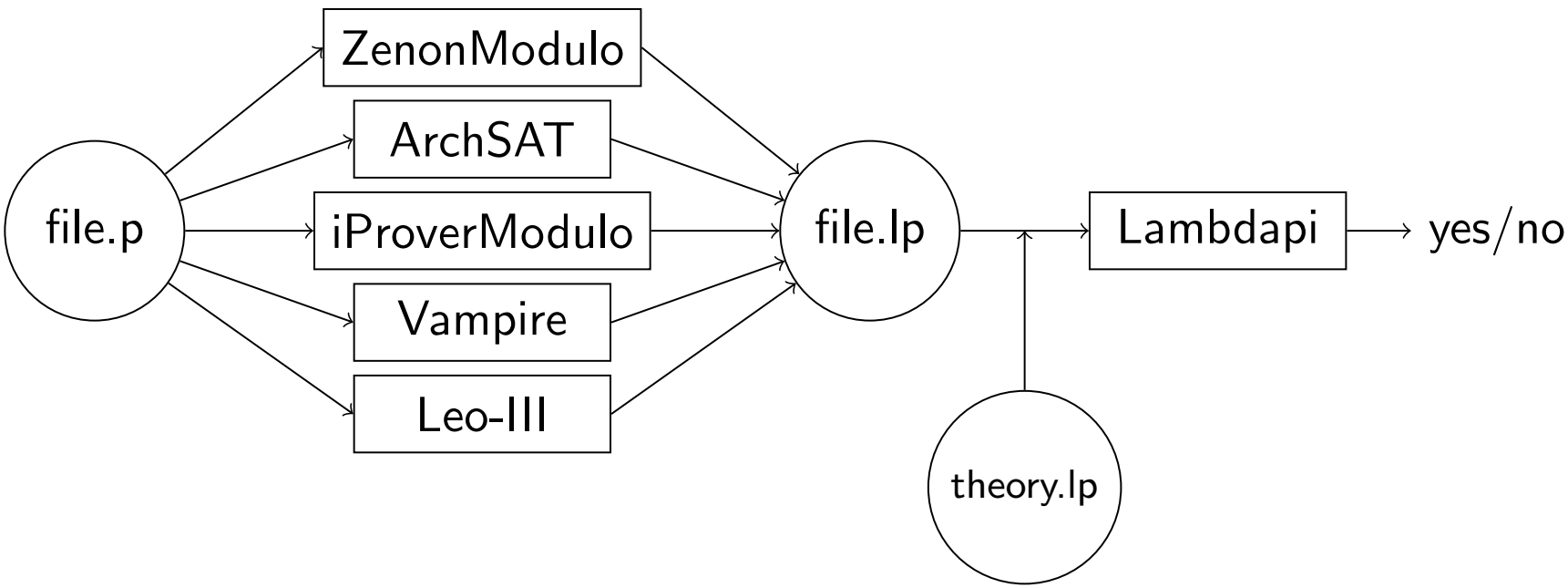
constant symbol ⇒ : Prop → Prop → Prop;
notation ⇒ infix right 5;
rule π ($p ⇒ $q) ↪ π $p → π $q; // BHK interpretation

constant symbol ∀ [a] : (τ a → Prop) → Prop;
notation ∀ quantifier;
rule π (`∀ x, $f.[x]) ↪ Π x, π $f.[x]; // BHK interpretation

...
```

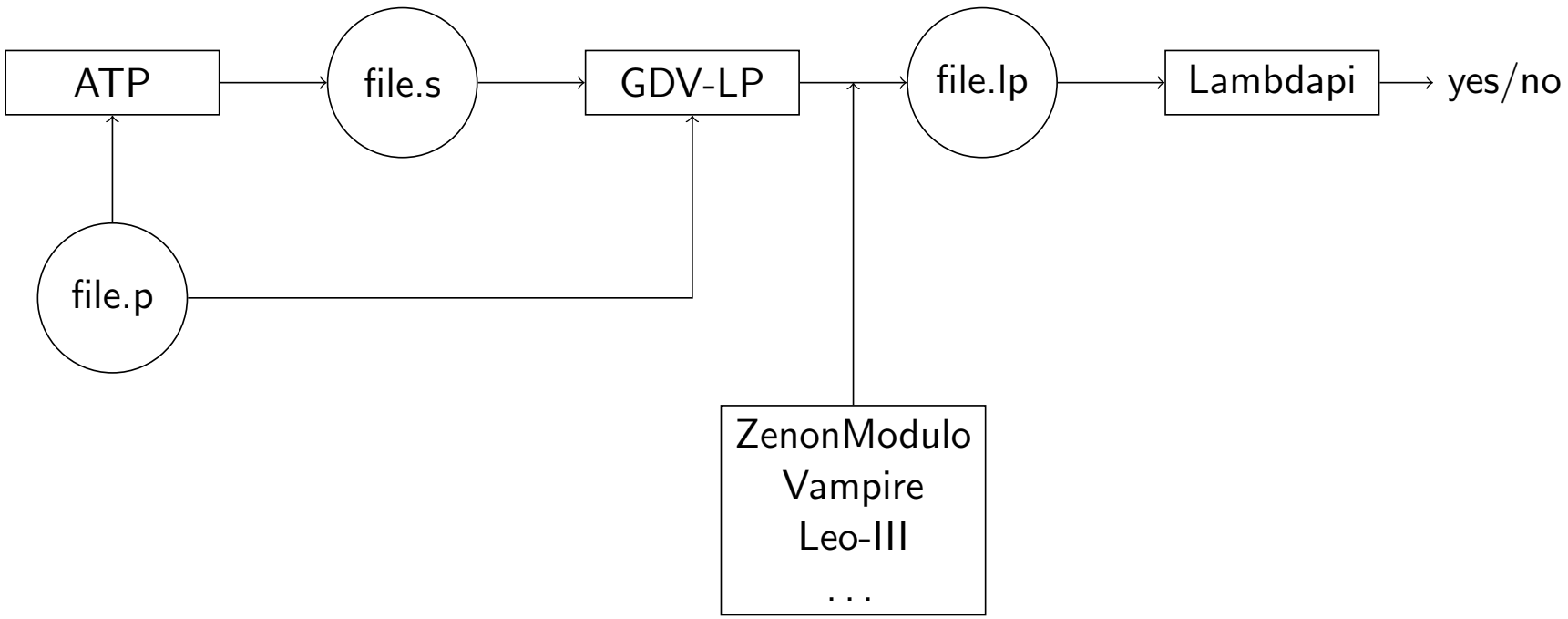
ATPs outputing dk/lp proofs

- ZenonModulo [Delahaye *et al*, 2013]
- ArchSAT [Bury, 2019]
- iProverModulo [Burel, 2019]
- Vampire [Petković *et al*, 2025] NEW!
- Leo-III [Taprogge, 2025] NEW!
- AtelierB PP [Dunne, 2026] NEW!



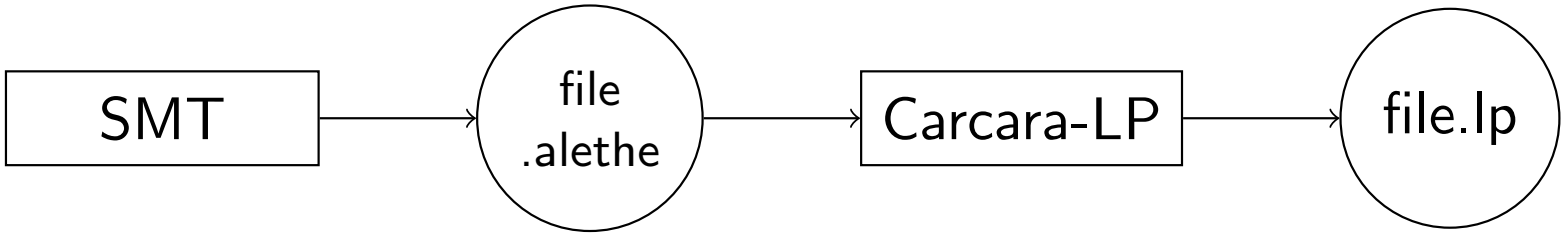
TSTP proof certification

Geoff Sutcliffe's derivation verification tool GDV can be combined with ATPs generating lp proofs [Sutcliffe, B., Burel, 2025]

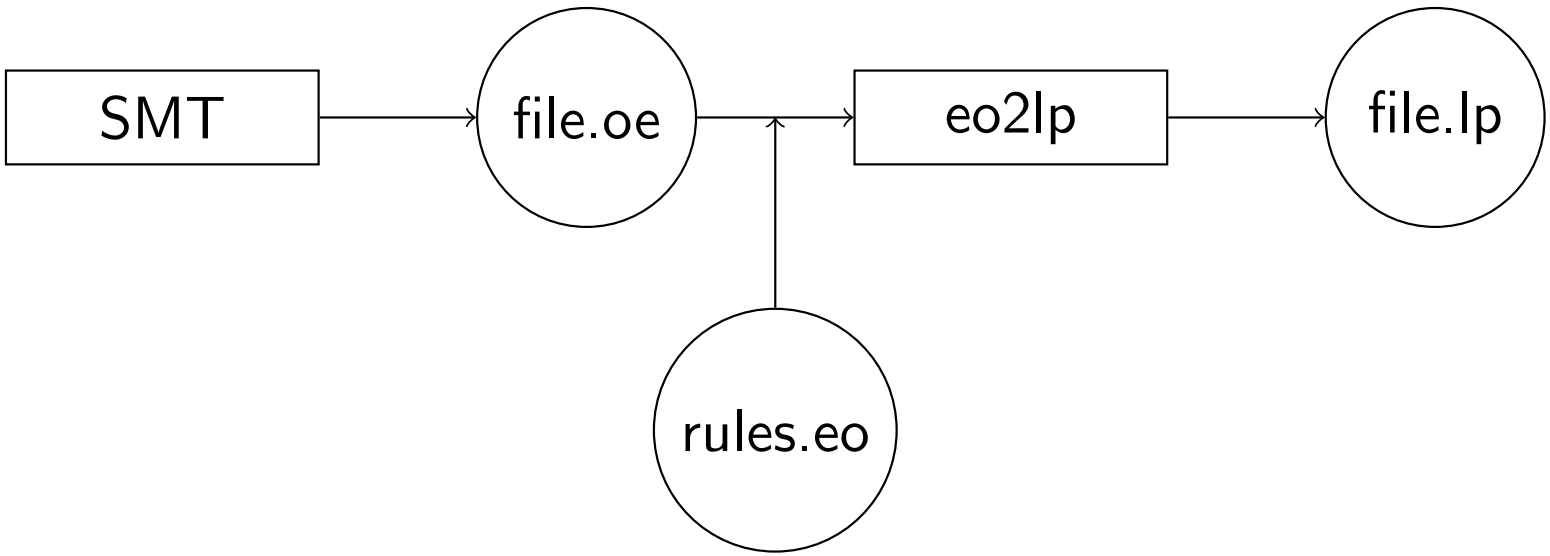


SMT Alethe proof certification

- Alethe proofs can be translated to Lambdapi using a fork of Carcara [Coltellacci, 2024]



- SMT rules in Eunoia (e.g. CPC rules for cvc5) can be translated to Lambdapi checking programs [Dunne, 2026]



ITP to Dedukti translators

- OpenTheory [Assaf, 2015]
- Matita [Assaf, 2015]
- Isabelle [Wenzel, B., Yamada, Dubut, Gontard, 2019]
- Rocq [Férey, 2021]
- PVS (statements only) [Hondet, 2022]
- Agda [Felicissimo, 2024]
- HOL-Light [B., 2025]
- Lean [Vaishnav, 2026]
- Mizar (statements only) [Kornilowicz, 2026]

those in red continue to be developed

Dedukti to ITP translators

- Lambdapi can translate higher-order logic proofs to Rocq/Lean [B., 2024] [Martinez, 2026]
- this is used to translate HOL-Light libraries to Rocq and Lean:
  - `Logic` library on the meta-theory of first-order logic
  - `Multivariate` lib on real and complex analysis ($>$ 20,000 theorems)
- by replacing HOL-Light types/functions by the corresponding Rocq/Lean types/functions already defined in Stdlib/MathComp/Mathlib [B., Gontard, Martinez *et al*, 2025]

  The translated theorems are provided on:
  - https://github.com/Deducteam/lean-hollight
  - https://github.com/Deducteam/rocq-hollight

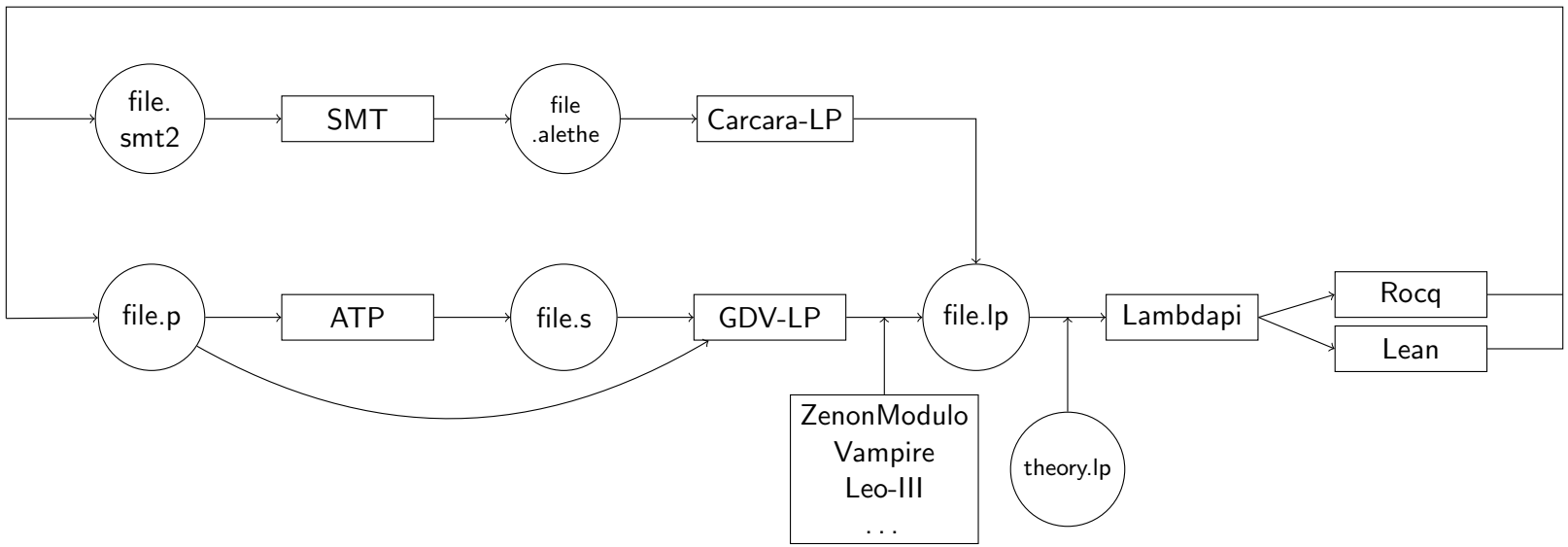
    (Opam package `rocq-hollight`)
- equivalence of HOL-Light and Rocq/Lean definitions for inductive types and primitive recursive functions automatically done by dedicated tactics [Gontard, 2025] [Martinez, 2026]

Indexing and searching Lambdapi libraries and their translations [Sacerdoti, Alidra, 2025]

- Lambdapi provides commands to index, search definitions, rules and theorems, and run a webserver
- Example: `https://lambdapi.saclay.inria.fr/` with an indexing of HOL-Light Multivariate and its Rocq translation (currently not available because of the PHP bug)

  more libraries will be added next year

Future work: ATP/SMT to ITP translation, an alternative to hammers



remark: Lambdapi subgoals can be outsourced to external provers using Why3