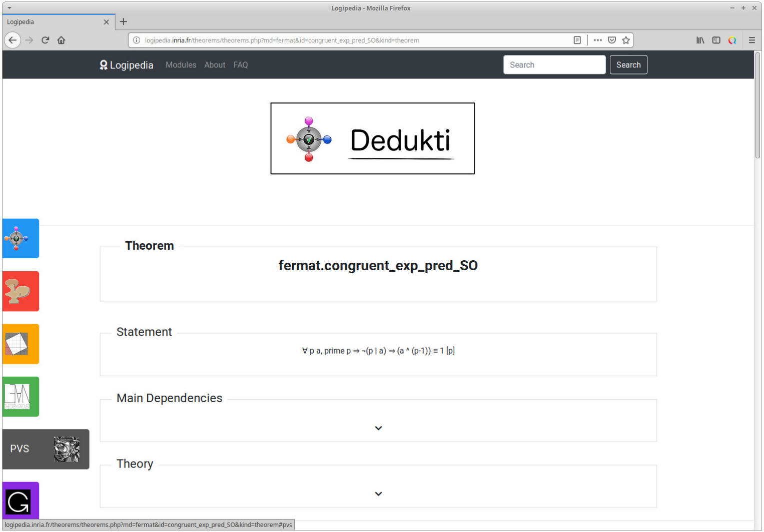
Logipedia

How to use it, and how to contribute to it http://logipedia.science



Logical Frameworks

A logical system (Euclidean geometry, set theory, Simple type theory, the Calculus of constructions...) should not be defined as independent system

They should be expressed in a Logical framework

Logical Frameworks: Predicate logic (1928), λ-Prolog, Isabelle, Pure type systems, the $\lambda\Pi$-calculus (LF), Deduction modulo theory, the $\lambda\Pi$-calculus modulo theory (Dedukti)

Each theory breaks down into a number of axioms / rewrite rules

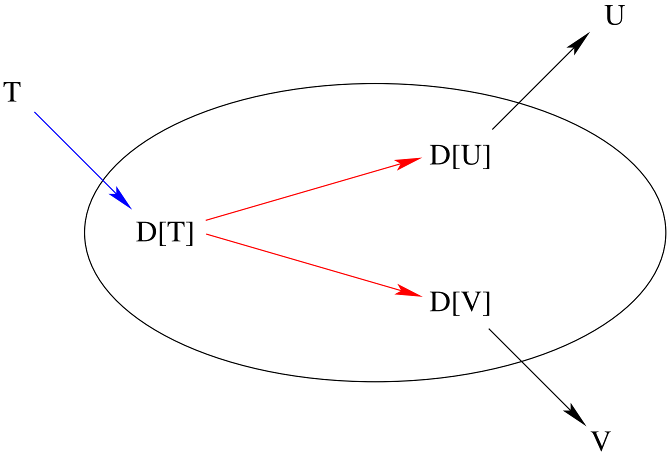
Permits to analyze which proof uses which axiom / rewrite rule (reverse mathematics)

Logipedia

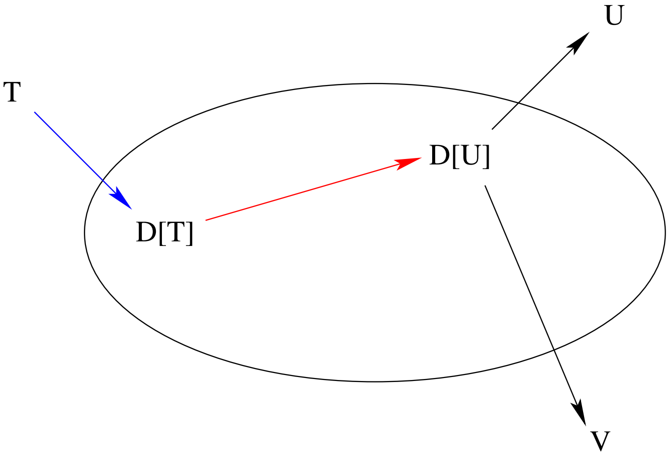
An encyclopedia of proofs expressed

- in various theories
- in DEDUKTI

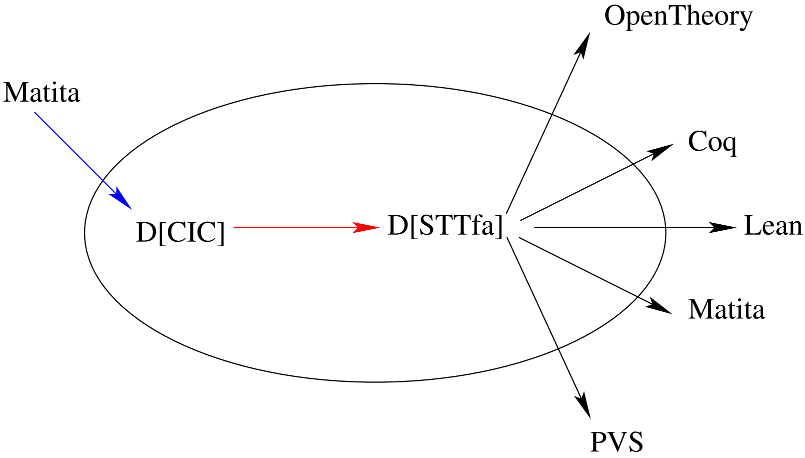
Proof translation



But also



An example



I. Defining a theory in DEDUKTI

No universal method

Depends on the theory
But several "paradigmatic" examples in *Dedukti: a Logical Framework based on the lambda-Pi-Calculus Modulo Theory.*

- Any (finite) theory expressed in Predicate logic
- Axiom schemes
- Simple type theory (without and with polymorphism)
- Pure type systems (CoC...)
- Inductive types
- Universes

Ongoing work

- Inductive types
- Universes (with universe polymorphism)
- Proof irrelevance
- Predicate subtyping

An example: Simple type theory

$$
\begin{array}{rcl}
type & : & \mathit{Type} \\
Te & : & type \rightarrow \mathit{Type} \\
o & : & type \\
nat & : & type \\
arrow & : & type \rightarrow type \rightarrow type \\
Pf & : & (Te\ o) \rightarrow \mathit{Type} \\
\Rightarrow & : & (Te\ o) \rightarrow (Te\ o) \rightarrow (Te\ o) \\
\forall & : & \Pi a : type\ (((Te\ a) \rightarrow (Te\ o)) \rightarrow (Te\ o))
\end{array}
$$

$$
\begin{array}{rcl}
(Te\ (arrow\ x\ y)) & \longrightarrow & (Te\ x) \rightarrow (Te\ y) \\
(Pf\ (\Rightarrow\ x\ y)) & \longrightarrow & (Pf\ x) \rightarrow (Pf\ y) \\
(Pf\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (Te\ x)\ (Pf\ (y\ z))
\end{array}
$$

Examples

Types: $nat \to nat$ expressed as $(arrow\ nat\ nat)$ of type $type$
Then to $(Te\ (arrow\ nat\ nat))$ of type $Type$ that reduces to
$(Te\ nat) \ \to\ (Te\ nat)$

Terms: $\lambda x : nat\ x$ expressed as $\lambda x : (Te\ nat)\ x$ of type
$(Te\ nat) \ \to\ (Te\ nat)$

Propositions: $\forall X : o\ (X \Rightarrow X)$ expressed as
$\forall\ o\ \lambda X : (Te\ o)\ (\Rightarrow\ X\ X)$ of type $(Te\ o)$
Then to $(Pf\ (\forall\ o\ \lambda X : (Te\ o)\ (\Rightarrow\ X\ X)))$ of type $Type$ that reduces to $\Pi X : (Te\ o)\ ((Pf\ X)\ \to\ (Pf\ X))$.

Proofs: well-known expressed as $\lambda X : (Te\ o)\ \lambda \alpha : (Pf\ X)\ \alpha$ of type $\Pi X : (Te\ o)\ ((Pf\ X)\ \to\ (Pf\ X))$

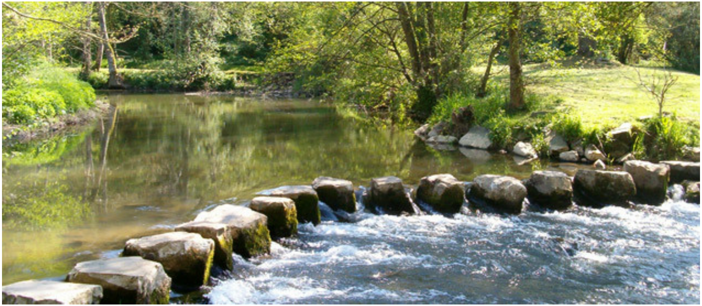
II. Exporting proofs from DEDUKTI

Three types of systems

- Those with explicit proof terms (Automath-like: Coq, Matita, Lean, Agda...)
- Those with predictable tactics (LCF-like: HOL Light, Isabelle/HOL...)
- Those with neither (PVS-like: PVS)

Three types of systems

- Those with explicit proof terms (Automath-like: Coq, Matita, Lean, Agda...)
  Just translate the proof term
- Those with predictable tactics (LCF-like: HOL Light, Isabelle/HOL...)
  Generate tactics (at the level of Natural deduction rules)
- Those with neither (PVS-like: PVS)
  A tree such that a proposition labeling a node is not too difficult to prove from those labeling its children and cut
  Example : a = b, b = a ...
  State $\vdash a = b$
  Cut on $b = a$
  Prove automatically $b = a \vdash a = b$
  Continue with $\vdash b = a$



Easy to do

One day, one week... depending on the system

III. Importing proofs to DEDUKTI

More difficult

Usually requires to instrument the source system

But done with MATITA, HOL LIGHT, FOCALIZE, IPROVER, ZENON, ARCHSAT

- ZENON and ARCHSAT have been designed with a DEDUKTI output
- HOL LIGHT has a output to some proof certificates OPENTHEORY, that we could translate to DEDUKTI

Same three types of systems

- Those with explicit proof terms (Automath-like)
  Just translate the proof term
- Those with a small set of primitive tactics (LCF-like) used to build the others
  Instrument the primitive tactics only
- Those with neither (PVS-like), in particular iProver
  Ford technique (again)
  Output a list of intermediate steps, use an automated theorem prover (that output Dedukti proofs) to fill the gaps, rebuild the puzzle from the pieces

IV. Reverse mathematics in Dedukti

(A slight extension of) the Calculus of constructions as a theory in in Dedukti

$$
\begin{array}{rcl}
type & : & Type \\
Te & : & type \to Type \\
o & : & type \\
nat & : & type \\
arrow & : & \Pi x : type\ (((Te\ x) \to type) \to type) \\
Pf & : & (Te\ o) \to Type \\
\Rightarrow & : & \Pi x : (Te\ o)\ (((Pf\ x) \to (Te\ o)) \to (Te\ o)) \\
\forall & : & \Pi x : type\ (((Te\ x) \to (Te\ o)) \to (Te\ o)) \\
\pi & : & \Pi x : (Te\ o)\ (((Pf\ x) \to type) \to type)
\end{array}
$$

$$
\begin{array}{rcl}
(Te\ (arrow\ x\ y)) & \longrightarrow & \Pi z : (Te\ x)\ (Te\ (y\ z)) \\
(Pf\ (\Rightarrow\ x\ y)) & \longrightarrow & \Pi z : (Pf\ x)\ (Pf\ (y\ z)) \\
(Pf\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (Te\ x)\ (Pf\ (y\ z)) \\
(Te\ (\pi\ x\ y)) & \longrightarrow & \Pi z : (Pf\ x)\ (Te\ (y\ z))
\end{array}
$$

(A slight extension of) the Calculus of constructions as a theory in in Dedukti

$$
\begin{array}{rcl}
type & : & Type \\
Te & : & type \to Type \\
o & : & type \\
nat & : & type \\
arrow & : & \Pi x : type\ (((Te\ x) \to type) \to type) \\
Pf & : & (Te\ o) \to Type \\
\Rightarrow & : & \Pi x : (Te\ o)\ (((Pf\ x) \to (Te\ o)) \to (Te\ o)) \\
\forall & : & \Pi x : type\ (((Te\ x) \to (Te\ o)) \to (Te\ o)) \\
\pi & : & \Pi x : (Te\ o)\ (((Pf\ x) \to type) \to type)
\end{array}
$$

$$
\begin{array}{rcl}
(Te\ (arrow\ x\ y)) & \longrightarrow & \Pi z : (Te\ x)\ (Te\ (y\ z)) \\
(Pf\ (\Rightarrow\ x\ y)) & \longrightarrow & \Pi z : (Pf\ x)\ (Pf\ (y\ z)) \\
(Pf\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (Te\ x)\ (Pf\ (y\ z)) \\
(Te\ (\pi\ x\ y)) & \longrightarrow & \Pi z : (Pf\ x)\ (Te\ (y\ z))
\end{array}
$$

Comparing the theories

arrow in Simple type theory

$$\Pi x : type\ (type \to type)$$

in the Calculus of constructions

$$\Pi x : type\ (((Te\ x) \to type) \to type)$$

In the Calculus of constructions, dependent *arrow*: in $A \to B$ (written $\Pi x : A\ B$), B can contain a variable x of type A

Same for $\Rightarrow$
($\forall$ is dependent is both theories)

An extra constant π in the Calculus of constructions: typing functions mapping proofs to terms

Analyzing proofs expressed in the Calculus of constructions

A subset S of the proofs expressed in the Calculus of constructions

- do not use the dependency of *arrow*
- do not use the dependency of the symbol $\Rightarrow$,
- do not use the symbol π

Many proofs expressed in the Calculus of constructions in S

Translating proofs to Simple type theory

A proof in the Calculus of constructions

In S

Translation to Simple type theory:

replace $(arrow\ A\ \lambda x : (Te\ A)\ B)$ with $(arrow\ A\ B)$

(similar for $\Rightarrow$)

Not in S

Genuinely uses a feature of the Calculus of constructions that does not exist in Simple type theory

Cannot be expressed in Simple type theory

Same as in ZFC: genuinely uses the axiom of choice: not in ZF

Weaker and weaker

Currently: the “first” proof of Fermat’s little theorem in constructive Simple type theory (no full polymorphism, no dependent types, no universes...)

Further: predicative constructive Simple type theory

Further?: PA, fragments of PA...

V. Towards concept alignment

Connectives and quantifiers

Inductive types / Q_0
Should be ignored by the library

Making formal the saying: Cauchy sequences or Dedekind cuts immaterial (isomorphic and only structural statements)

But may be: one classical disjunction and one constructive one (Ecumenical systems)

Further

The induction principle

Justified in different ways in different systems (axiom, consequence of the definition of natural numbers...)

Does not matter as long as it is there

Not the first attempt to build a standard or a library

Why will / might it work this time?

- A better understanding of the theories behind the provers (40 years of research in logic)
- Success stories in point to point translations (Coq / HOL Light)
- A logical framework to express these theories (more abstract view)
- Try to accommodate as many people as possible but not all (theories expressed in DEDUKTI, e.g. predicate subtyping: research effort)
- Analyzing the proofs (reverse mathematics) before we share them (partial translations)

First discussion before we go deeper

Which proof libraries should we target?

Which similar effort should we build upon?

What should we expect from an encyclopedia?