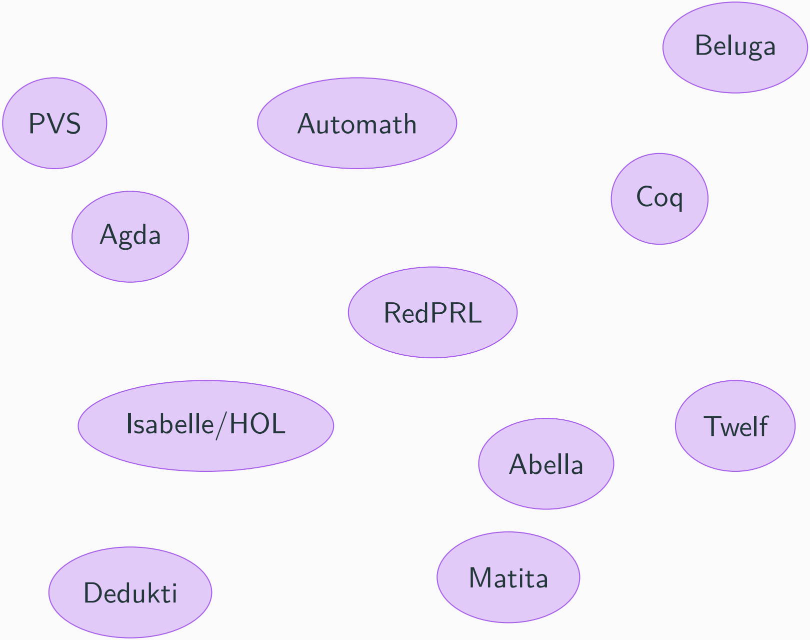
Sharing a Library between Proof Assistants: Reaching out the HOL Family

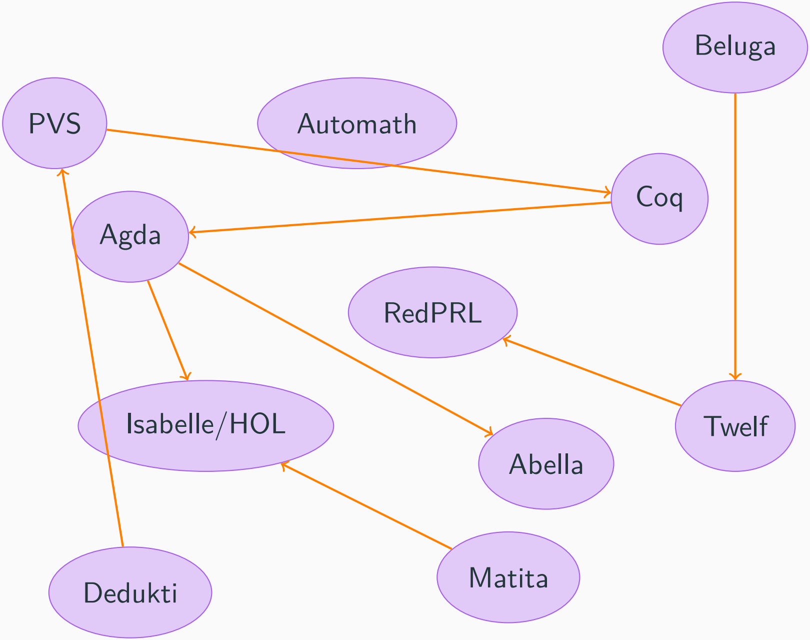
François Thiré

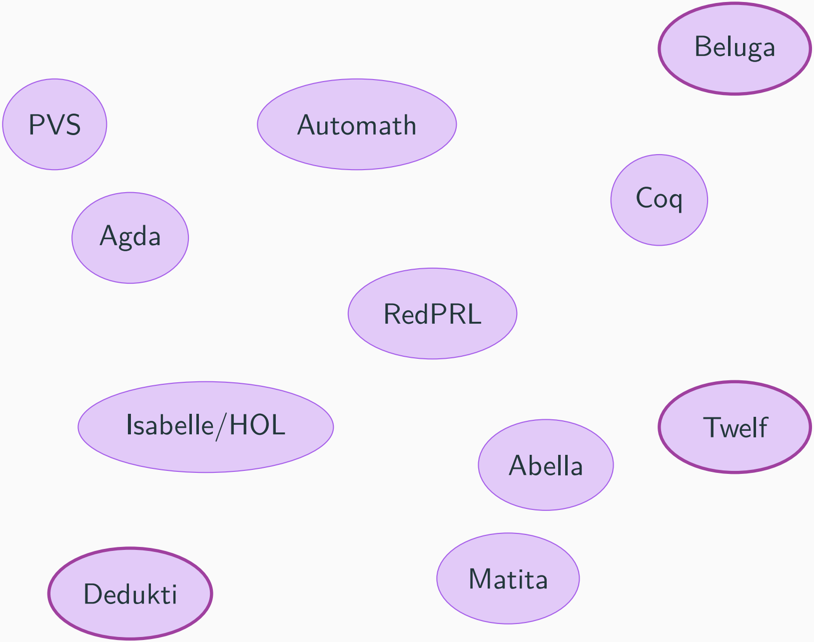
July 7, 2018

LSV, CNRS, Inria, ENS Paris-Saclay

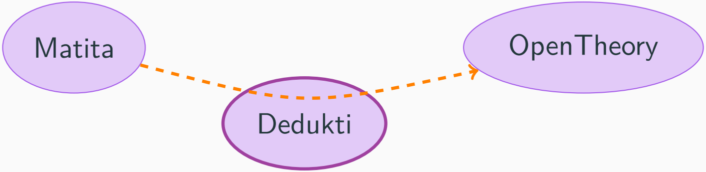
Introduction

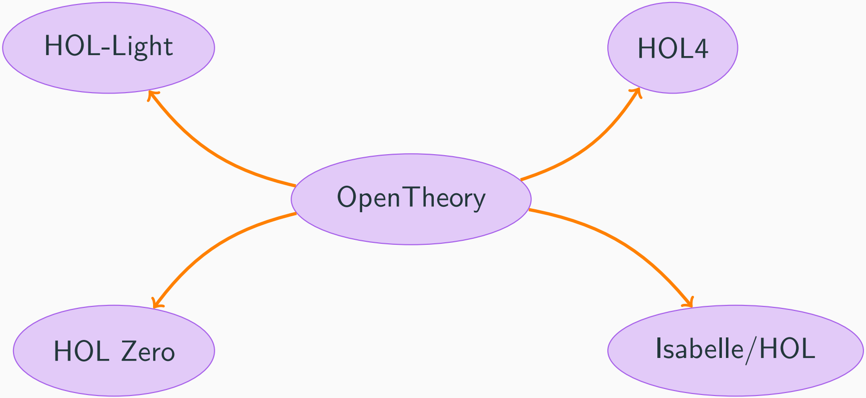

Beluga
PVS
Automath
Coq
Agda
RedPRL
Isabelle/HOL
Twelf
Abella
Dedukti
Matita

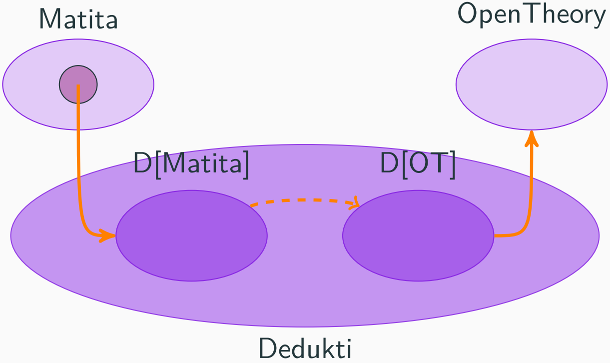

Beluga
PVS
Automath
Coq
Agda
RedPRL
Isabelle/HOL
Twelf
Abella
Dedukti
Matita


Beluga
PVS
Automath
Coq
Agda
RedPRL
Isabelle/HOL
Twelf
Abella
Dedukti
Matita

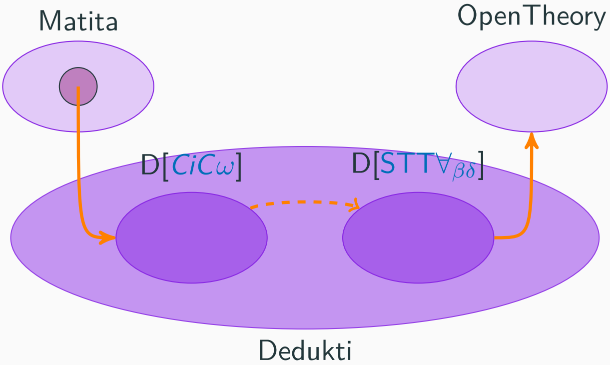

Dedukti

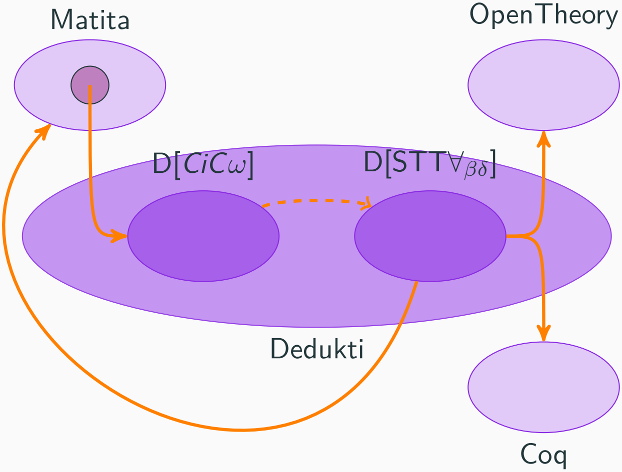

Matita
Dedukti
OpenTheory

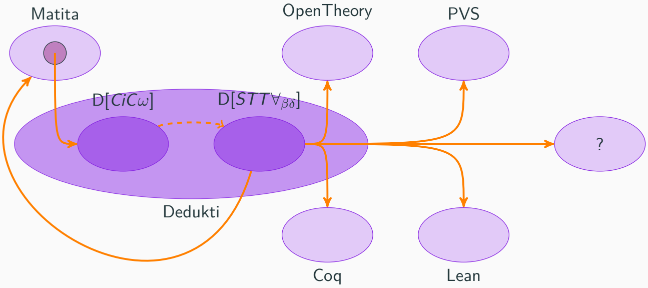

HOL-Light
HOL4
OpenTheory
HOL Zero
Isabelle/HOL



Matita
OpenTheory
D[Matita]
D[OT]
Dedukti

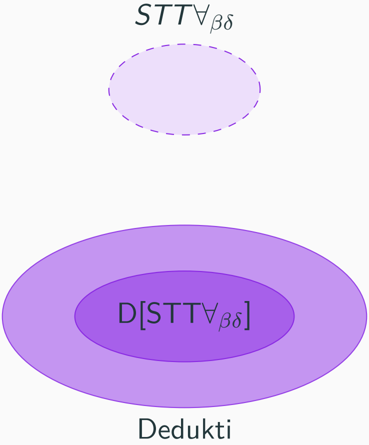

Matita
OpenTheory
D[$CiC\omega$]
D[STT$\forall_{\beta\delta}$]
Dedukti


Matita
OpenTheory
D[$CiC\omega$]
D[STT$\forall_{\beta\delta}$]
Dedukti
Coq

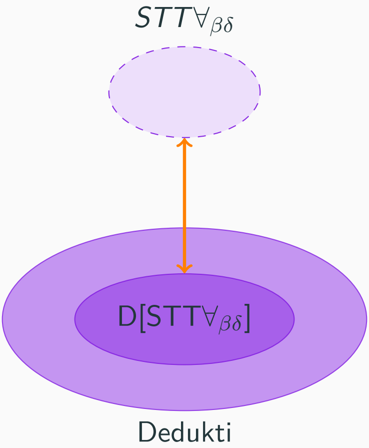

Matita
OpenTheory
PVS
D[$CiC\omega$]
D[$STT\forall_{\beta\delta}$]
?
Dedukti
Coq
Lean

$\mathit{STT}\forall_{\beta\delta}$

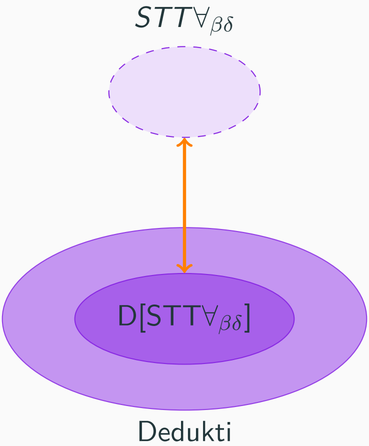
A real implementation of $STT\forall_{\beta\delta}$?



A real implementation of $STT\forall_{\beta\delta}$?



A real implementation of $STT\forall_{\beta\delta}$?



In this talk, Dedukti is abstract!

The encoding is shallow

STT

$$
\begin{array}{llll}
\textbf{Types} & A, B & :\equiv & \iota \mid o \mid A \to B \\
\textbf{Terms} & t, u & :\equiv & x \mid \lambda x^A.\ t \mid t\ u \mid \forall x^A.\ t \mid t \Rightarrow u
\end{array}
$$

STT

$$\textbf{Types} \quad A, B \quad :\equiv \quad \iota \mid o \mid A \to B$$
$$\textbf{Terms} \quad t, u \quad :\equiv \quad x \mid \lambda x^A.\ t \mid t\ u \mid \forall x^A.\ t \mid t \Rightarrow u$$

$$\frac{\mathcal{C} \vdash t : o}{\mathcal{C}, t \vdash t}\ \text{ASSUME} \qquad \frac{\mathcal{C} \vdash t \qquad \mathcal{C} \vdash t \Rightarrow u}{\mathcal{C} \vdash u} \Rightarrow_E \qquad \frac{\mathcal{C}, t \vdash u}{\mathcal{C} \vdash t \Rightarrow u} \Rightarrow_I$$

$$\frac{\mathcal{C} \vdash \forall x^A.\ t \qquad \mathcal{C} \vdash u : A}{\mathcal{C} \vdash t[x := u]} \forall_E \qquad \frac{\mathcal{C}, x : A \vdash t \qquad x \notin \mathcal{C}}{\mathcal{C} \vdash \forall x^A.\ t} \forall_I$$

Fig. 1: Proof system

Types $A, B \quad :\equiv \quad \iota \mid o \mid A \to B$

Terms $t, u \quad :\equiv \quad x \mid \lambda x^A.\ t \mid t\ u \mid \forall x^A.\ t \mid t \Rightarrow u$

$$\frac{\mathcal{C} \vdash t : o}{\mathcal{C}, t \vdash t}\ \text{ASSUME} \qquad \frac{\mathcal{C} \vdash t \qquad t \equiv_{\beta\delta} t'}{\mathcal{C} \vdash t'}\ \text{CONV}$$

$$\frac{\mathcal{C} \vdash t \qquad \mathcal{C} \vdash t \Rightarrow u}{\mathcal{C} \vdash u} \Rightarrow_E \qquad \frac{\mathcal{C}, t \vdash u}{\mathcal{C} \vdash t \Rightarrow u} \Rightarrow_I$$

$$\frac{\mathcal{C} \vdash \forall x^A.\ t \qquad \mathcal{C} \vdash u : A}{\mathcal{C} \vdash t[x := u]}\ \forall_E \qquad \frac{\mathcal{C}, x : A \vdash t \qquad x \notin \mathcal{C}}{\mathcal{C} \vdash \forall x^A.\ t}\ \forall_I$$

Fig. 1: Proof system

STT$\forall_{\beta\delta}$ is an extension of STT

$$STT\forall_{\beta\delta} = STT_{\beta\delta} + \text{prenex polymorphism}$$

STT$\forall_{\beta\delta}$ is an extension of STT

monotypes	A, B	$:\equiv$	$o \mid A \to B \mid X \mid p\ A_1\ \ldots\ A_n$
polytypes	T	$:\equiv$	$\forall_K X.\ T \mid A$

- nat
- $\forall_K X.\ \mathit{list}\ X$
- list nat
- $\forall_K X.\ X \to X \to o$

STT$\forall_{\beta\delta}$ is an extension of STT

monotypes	A, B	$:\equiv$	$o \mid A \rightarrow B \mid X \mid p\ A_1\ \ldots\ A_n$
polytypes	T	$:\equiv$	$\forall_K X.\ T \mid A$
monoterms	t,u	$:\equiv$	$\ldots \mid c\ A_1\ \ldots\ A_n \mid \Lambda X.\ t$
polyterms	τ	$:\equiv$	$\forall X.\ \tau \mid t$

- $0\ :\ \mathrm{nat}$
- $\Lambda X.\ \lambda x^X.\ \lambda y^X.\ \forall P^{X \rightarrow o}.\ P\ x \Rightarrow P\ y : \forall_K X.\ X \rightarrow X \rightarrow o$ (eq)
- $\forall X.\ \forall a^X.\ eq\ X\ a\ a$

STT$\forall_{\beta\delta}$ is an extension of STT

monotypes	A, B	$:\equiv$	$o \mid A \to B \mid X \mid p\ A_1\ \ldots\ A_n$
polytypes	T	$:\equiv$	$\forall_K X.\ T \mid A$
monoterms	t,u	$:\equiv$	$\ldots \mid c\ A_1\ \ldots\ A_n \mid \Lambda X.\ t$
polyterms	τ	$:\equiv$	$\forall\!\!\!/ X.\ \tau \mid t$

$$\ldots$$

$$\frac{\mathcal{C} \vdash \forall\!\!\!/ X.\ \tau \qquad \mathcal{C} \vdash A\ \textbf{wf}}{\mathcal{C} \vdash \tau[X := A]}\ \forall\!\!\!/_E \qquad\qquad \frac{\mathcal{C}, X \vdash \tau}{\mathcal{C} \vdash \forall\!\!\!/ X.\ \tau}\ \forall\!\!\!/_I$$

Fig. 2: Rules for STT$\forall_{\beta\delta}$

A reflexivity proof

$$\dfrac{}{eq;\emptyset;\emptyset \vdash \forall X.\ \forall a^{X}.\ eq\ X\ a\ a}\ \forall_I$$

A reflexivity proof

$$\dfrac{\dfrac{}{eq; X; \emptyset \vdash \forall a^{X}.\ eq\ X\ a\ a}\ \forall_I}{eq; \emptyset; \emptyset \vdash \forall X.\ \forall a^{X}.\ eq\ X\ a\ a}\ \forall_I$$

A reflexivity proof

$$\dfrac{\dfrac{\dfrac{}{eq; X, a : X; \emptyset \vdash eq\ X\ a\ a}\ \textsc{conv}}{eq; X; \emptyset \vdash \forall a^{X}.\ eq\ X\ a\ a}\ \forall_I}{eq; \emptyset; \emptyset \vdash \forall X.\ \forall a^{X}.\ eq\ X\ a\ a}\ \forall_I$$

A reflexivity proof

$$\cfrac{\cfrac{\cfrac{\cfrac{}{eq; X, a : X; \emptyset \vdash P\ a \Rightarrow P\ a}\ \Rightarrow_I}{eq; X, a : X; \emptyset \vdash eq\ X\ a\ a}\ \textsc{conv}}{eq; X; \emptyset \vdash \forall a^X.\ eq\ X\ a\ a}\ \forall_I}{eq; \emptyset; \emptyset \vdash \forall X.\ \forall a^X.\ eq\ X\ a\ a}\ \forall_I$$

A reflexivity proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{eq; X, a : X; P\ a \vdash P\ a}\ \text{ASSUME}}{eq; X, a : X; \emptyset \vdash P\ a \Rightarrow P\ a}\ \Rightarrow_I}{eq; X, a : X; \emptyset \vdash eq\ X\ a\ a}\ \text{CONV}}{eq; X; \emptyset \vdash \forall a^X.\ eq\ X\ a\ a}\ \forall_I}{eq; \emptyset; \emptyset \vdash \forall X.\ \forall a^X.\ eq\ X\ a\ a}\ \forall_I$$

$STT\forall_{\beta\delta}$ as a PTS

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash (x : A) \to B : s_3}$$

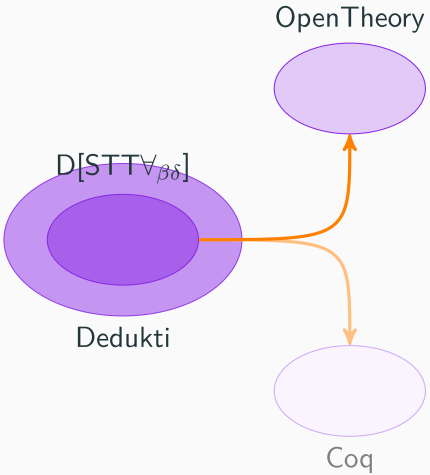
$STT\forall_{\beta\delta}$ as a PTS

$$\mathcal{S}, \mathcal{A} = \textbf{Prop} : \textbf{Type} : \textbf{Kind}$$

$\forall_K$	(**Type**, **Kind**, **Kind**)
$\forall$	(**Type**, **Prop**, **Prop**)
$\Rightarrow$	(**Prop**, **Prop**, **Prop**)
$\rightarrow$	(**Type**, **Type**, **Type**)
$\not\forall$	(**Kind**, **Prop**, **Prop**)

Type $\prec$ **Kind** (subtyping)

OpenTheory


OpenTheory
D[STT∀$_{\beta\delta}$]
Dedukti
Coq

OpenTheory vs STT$\forall_{\beta\delta}$

Terms and types are almost the same!

Three main differences:

In STT$\forall_{\beta\delta}$:

- β and δ extensional
- $\forall$ and $\Rightarrow$ connectives
- $\forall_K$ is explicit

In OpenTheory:

- β and δ intensional
- $=$ connective
- $\forall_K$ is implicit

OpenTheory vs STT$\forall_{\beta\delta}$

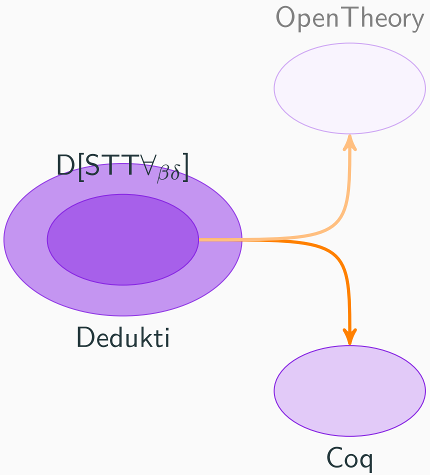
In STT$\forall_{\beta\delta}$:		In OpenTheory:
• β and δ extensional	*hard* →	• β and δ intensional
• $\forall$ and $\Rightarrow$ connectives	*easy* →	• $=$ connective
• $\forall_K$ is explicit	*easy* →	• $\forall_K$ is implicit

Why is it hard?

$$\frac{\mathcal{C} \vdash t \qquad t \equiv_{\beta\delta} t'}{\mathcal{C} \vdash t'} \text{CONV}$$

- $\equiv_{\beta\delta}$ is the one of Dedukti
- How to annotate proofs? Reduce the term step by step.
- β of $STT\forall_{\beta\delta}$ vs administrative β
- Don't compute the normal form everytime!

Coq


OpenTheory
D[STT$\forall_{\beta\delta}$]
Dedukti
Coq

Trivial: $STT\forall_{\beta\delta}$ is a subsystem of Coq !

DEMO

Arithmetic library

	Dedukti[STT]	OpenTheory	Coq	Matita	Lean	PVS
size (mb)	1.5	41	0.6	0.6	0.6	9
translation time (s)	-	18	3	3	3	3
checking time (s)	0.1	13	6	2	1	$\sim$300

Arithmetic library

	Dedukti[STT]	OpenTheory	Coq	Matita	Lean	PVS
size (mb)	1.5	41	0.6	0.6	0.6	9
translation time (s)	-	18	3	3	3	3
checking time (s)	0.1	13	6	2	1	$\sim$300

- Theorems: 340 (Commutativity of addition, Fermat's little theorem)
- Parameters: 46 (nat, bool, ...)
- Axiom: 71 (equalities generated from recursive definitions,...)
- Definitions: 34 (le,primes,...)

Concept Alignement

Fermat's little theorem

```
Theorem congruent_exp_pred_SO :
forall p a : Nat, prime p -> Not (divides p a) ->
congruent (exp a (pred p)) (S O) p.
```

Fermat's little theorem

```
Theorem congruent_exp_pred_SO :
forall p a : Nat, prime p -> Not (divides p a) ->
congruent (exp a (pred p)) (S O) p.
```

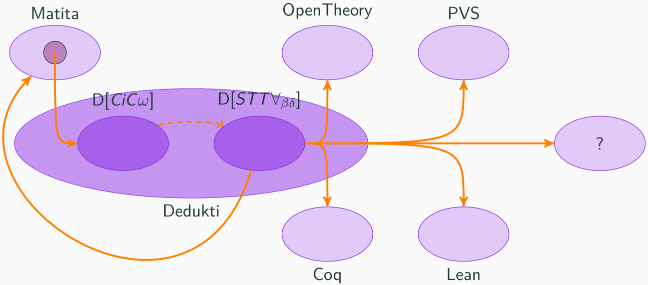
Fermat's little theorem

```
Theorem congruent_exp_pred_SO :
forall p a : Nat, prime p -> Not (divides p a) ->
congruent (exp a (pred p)) (S O) p.

Parameter exp : Nat -> Nat -> Nat.
Axiom axiom_exp_O : forall n : Nat,
      equal Nat (exp n O) (S O).
Axiom axiom_exp_S : forall n m : Nat,
      equal Nat (exp n (S m)) (times (exp n m) n).
```

Conclusion

Conclusion



- A relatively weak logic: $STT\forall_{\beta\delta}$
- An automatic translation of a library to 5 other proof systems

Future work

- Sharing the aritmetic library to other systems (Agda, Idris,...)
- Developing an encyclopedia of proofs: Logipedia
- A standardization of an arithmetic library?

Future work

- Sharing the aritmetic library to other systems (Agda, Idris,...)
- Developing an encylopedia of proofs: Logipedia
- A standardization of an arithmetic library?

Contributions are welcome!

https://github.com/Deducteam/Logipedia