Object-Oriented Mechanisms for Interoperability between Proof Systems

Raphaël Cauderlier

October 10, 2016

Algorithm – Sieve of Eratosthenes

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Eratosthenes
3rd century BC

Algorithm – Sieve of Eratosthenes

	2	3		5		7		9	
11		13		15		17		19	
21		23		25		27		29	
31		33		35		37		39	
41		43		45		47		49	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

Eratosthenes
3rd century BC

Algorithm – Sieve of Eratosthenes

	2	3		5		7			
11		13				17		19	
		23		25				29	
31				35		37			
41		43				47		49	
		53		55				59	
61				65		67			
71		73				77		79	
		83		85				89	
91				95		97			

Eratosthenes
3rd century BC

Algorithm – Sieve of Eratosthenes

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47		49	
		53						59	
61						67			
71		73				77		79	
		83						89	
91						97			

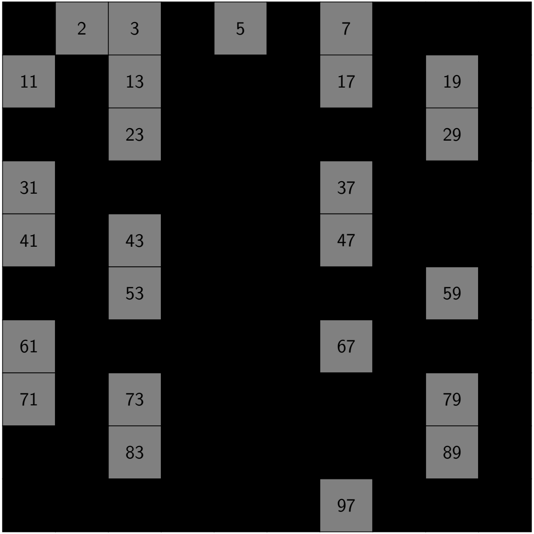
Eratosthenes
3rd century BC

Algorithm – Sieve of Eratosthenes

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

Eratosthenes
3rd century BC

Algorithm – Sieve of Eratosthenes



Eratosthenes
3rd century BC

Implementation

- Algorithms implemented as computer programs
  - + huge instances
  - + speed
  - + reliability
  - - programming errors

Program Specification

The numbers returned by the Sieve of Eratosthenes are the prime numbers below the bound.

Program Specification

The numbers returned by the Sieve of Eratosthenes are the prime numbers below the bound.

$$\forall n \in \mathbb{N}.\ \forall p \in \mathbb{N}.\ (p \in \texttt{eratosthenes}\ n) \Leftrightarrow (p \in \mathbb{P} \wedge p \leq n)$$

Formal Proof

- Automatic theorem provers (ATP)
    - fully automatic
    - weak logics

Formal Proof

- Automatic theorem provers (ATP)
  - fully automatic
  - weak logics
- Interactive theorem provers (ITP)
  - require interaction
  - very expressive logics

Formal Proof

- Automatic theorem provers (ATP)
  - fully automatic
  - weak logics
- Interactive theorem provers (ITP)
  - require interaction
  - very expressive logics
- Proof checkers
  - require detailed proofs

Interoperability

- Proof development is expensive
  - 4-color theorem, Kepler conjecture, Feit-Thomson theorem

Interoperability

- Proof development is expensive
  - 4-color theorem, Kepler conjecture, Feit-Thomson theorem
- Proof systems use incompatible logics
  - Axioms (classical vs. constructive)
  - Empty types
  - ...

Interoperability

- Proof development is expensive
  - 4-color theorem, Kepler conjecture, Feit-Thomson theorem
- Proof systems use incompatible logics
  - Axioms (classical vs. constructive)
  - Empty types
  - ...
- They are specializing
  - Counterexamples, proof by reflection, decision procedures, specialization to theories, ...

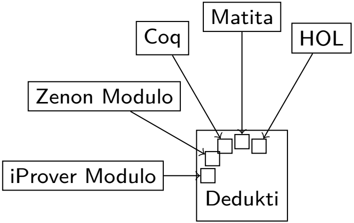
Interoperability

- Proof development is expensive
  - 4-color theorem, Kepler conjecture, Feit-Thomson theorem
- Proof systems use incompatible logics
  - Axioms (classical vs. constructive)
  - Empty types
  - ...
- They are specializing
  - Counterexamples, proof by reflection, decision procedures, specialization to theories, ...
- They do not interoperate
  - Proof exchange between Isabelle and HOL Light harder than it seems

Interoperability

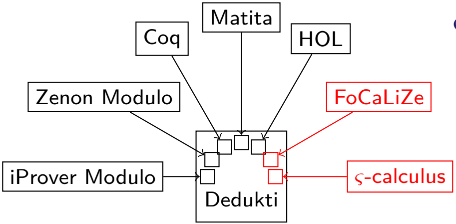
- Proof development is expensive
    - 4-color theorem, Kepler conjecture, Feit-Thomson theorem
- Proof systems use incompatible logics
    - Axioms (classical vs. constructive)
    - Empty types
    - ...
- They are specializing
    - Counterexamples, proof by reflection, decision procedures, specialization to theories, ...
- They do not interoperate
    - Proof exchange between Isabelle and HOL Light harder than it seems
- Deducteam proposes a common formalism: Dedukti

Interoperability in Dedukti



[*Expressing Theories in the* $\lambda\Pi$-*Calculus Modulo Theory and in the* DEDUKTI *System*, Assaf et all, 2016]

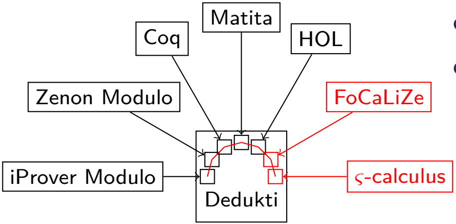
Interoperability in Dedukti



- More translators needed

[*Expressing Theories in the* $\lambda\Pi$-*Calculus Modulo Theory and in the* DEDUKTI *System*, Assaf et all, 2016]

Interoperability in Dedukti



- More translators needed
- Link Dedukti developments

[*Expressing Theories in the λΠ-Calculus Modulo Theory and in the* DEDUKTI *System*, Assaf et all, 2016]

Contents

Contents

Dedukti

- Deduction modulo:
  - Proof search modulo rewriting
  - Computation steps not recorded

Dedukti

- Deduction modulo:
  - Proof search modulo rewriting
  - Computation steps not recorded
- Dependent type theory
  - Basis for proof checkers
  - Logical framework
  - No Deduction modulo

Dedukti

- Deduction modulo:
  - Proof search modulo rewriting
  - Computation steps not recorded
- Dependent type theory
  - Basis for proof checkers
  - Logical framework
  - No Deduction modulo
- Dedukti = Dependent type theory + rewriting

Rewriting – First Example

$$
\begin{aligned}
\mathbb{N} &: \textbf{Type} \\
0 &: \mathbb{N} \\
\mathrm{succ} &: \mathbb{N} \to \mathbb{N} \\
+ &: \mathbb{N} \to \mathbb{N} \to \mathbb{N}
\end{aligned}
$$

$$
\begin{aligned}
0 + n &\hookrightarrow n \\
m + 0 &\hookrightarrow m \\
\mathrm{succ}(m) + n &\hookrightarrow \mathrm{succ}(m + n) \\
m + \mathrm{succ}(n) &\hookrightarrow \mathrm{succ}(m + n)
\end{aligned}
$$

Rewriting – Smart Constructor

$$
\begin{aligned}
&\mathbb{Z} && : \textbf{Type} \\
&\langle \bullet, \bullet \rangle && : \mathbb{N} \to \mathbb{N} \to \mathbb{Z} \\
\\
&\langle \mathsf{succ}(m), \mathsf{succ}(n) \rangle && \hookrightarrow \langle m, n \rangle \\
\\
&| \bullet | && : \mathbb{Z} \to \mathbb{N} \\
&|\langle m, 0 \rangle| && \hookrightarrow m \\
&|\langle 0, n \rangle| && \hookrightarrow n
\end{aligned}
$$

Dependent Type – Vectors

$$
\begin{aligned}
\text{Element} &: \textbf{Type} \\
\text{Vector} &: \mathbb{N} \to \textbf{Type} \\
\text{nil} &: \text{Vector } 0 \\
\text{cons} &: \Pi n : \mathbb{N}.\ \text{Element} \to \text{Vector } n \to \text{Vector } (\text{succ}(n)) \\
\text{append} &: \Pi m : \mathbb{N}.\ \Pi n : \mathbb{N}.\ \text{Vector } m \to \text{Vector } n \to \text{Vector } (m + n)
\end{aligned}
$$

$$
\begin{aligned}
\text{append } 0\ n\ \text{nil}\ w &\hookrightarrow w \\
\text{append } m\ 0\ v\ \text{nil} &\hookrightarrow v \\
\text{append } (\text{succ}(m))\ n\ (\text{cons } m\ e\ v)\ w &\hookrightarrow \\
\quad \text{cons } (m + n)\ e\ (\text{append } m\ n\ v\ w) &
\end{aligned}
$$

Encoding Logics

$$
\begin{aligned}
\mathsf{Prop} &: \mathbf{Type} \\
\top &: \mathsf{Prop} \\
\bot &: \mathsf{Prop} \\
\wedge &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vee &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\Rightarrow &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
= &: \mathbb{N} \to \mathbb{N} \to \mathsf{Prop} \\
\forall &: (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\exists &: (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop}
\end{aligned}
$$

Encoding Logics

$$
\begin{aligned}
\mathsf{Prop} &: \mathbf{Type} \\
\top &: \mathsf{Prop} \\
\bot &: \mathsf{Prop} \\
\wedge &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vee &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\Rightarrow &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
= &: \mathbb{N} \to \mathbb{N} \to \mathsf{Prop} \\
\forall &: (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\exists &: (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\vdash &: \mathsf{Prop} \to \mathbf{Type}
\end{aligned}
$$

Encoding Logics

$$
\begin{aligned}
\mathsf{Prop} &: \mathbf{Type} \\
\top &: \mathsf{Prop} \\
\bot &: \mathsf{Prop} \\
\wedge &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vee &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\Rightarrow &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
= &: \mathbb{N} \to \mathbb{N} \to \mathsf{Prop} \\
\forall &: (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\exists &: (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\vdash &: \mathsf{Prop} \to \mathbf{Type} \\
\mathsf{refl} &: \Pi n : \mathbb{N}.\ \vdash (n = n)
\end{aligned}
$$

Encoding Logics

$$
\begin{array}{ll}
\mathsf{Prop} & : \mathbf{Type} \\
\top & : \mathsf{Prop} \\
\bot & : \mathsf{Prop} \\
\wedge & : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vee & : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\Rightarrow & : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
= & : \mathbb{N} \to \mathbb{N} \to \mathsf{Prop} \\
\forall & : (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\exists & : (\mathbb{N} \to \mathsf{Prop}) \to \mathsf{Prop} \\
\vdash & : \mathsf{Prop} \to \mathbf{Type} \\
\mathsf{refl} & : \Pi n : \mathbb{N}.\ \vdash (n = n) \\
\mathsf{myproof} & : \vdash (2 + 2 = 4) \\
\mathsf{myproof} & \hookrightarrow \mathsf{refl}\ 4
\end{array}
$$

Encoding Programming Languages

$$\mu : \Pi A : \textbf{Type}.\ \Pi B : \textbf{Type}.\ ((A \to B) \to A \to B) \to A \to B$$
$$\mu\ A\ B\ F\ x \hookrightarrow F\ (\mu\ A\ B\ F)\ x$$

- Shallow encoding: operational semantics preserved

[*Objects and Subtyping in the $\lambda\Pi$-calculus modulo*, TYPES 2014]

Contents

FoCaLiZe, a Formal IDE

- Development of efficient and formally verified programs
- First-order specifications
- Automated proofs
- Modularity (inheritance and parameterization)
- Compilation to OCaml and Coq

Programming Language: ML

- Types : polymorphism and algebraic datatypes
- First-class functions, recursion, pattern-matching

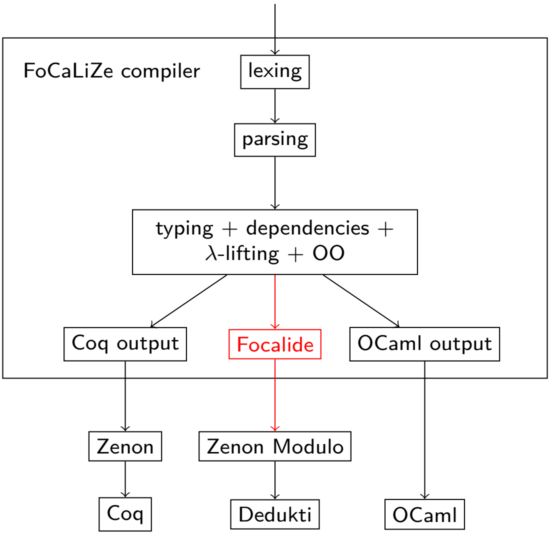
- Clear semantics
- Easy to translate to OCaml and Coq

Object-Oriented Mechanisms

- Species
- Multiple inheritance
- Parameterization
- Redefinition

- Static mechanisms

Compilation Scheme



Pattern-Matching: Naive Translation

```
let f (x) =
  match x with
  | 0 -> 0
  | n -> n-1;
```

Pattern-Matching: Naive Translation

```
let f (x) =
  match x with
  | 0 -> 0
  | n -> n-1;
```

$$
\begin{array}{ll}
\mathsf{f} & : \mathbb{Z} \to \mathbb{Z} \\
 & \\
\mathsf{f}\ 0 & \hookrightarrow 0 \\
\mathsf{f}\ n & \hookrightarrow n - 1
\end{array}
$$

Pattern-Matching: Naive Translation

```
let f (x) =
  match x with
  | 0 -> 0
  | n -> n-1;
```

$$\begin{aligned} \mathsf{f} &: \mathbb{Z} \to \mathbb{Z} \\ \mathsf{f}\ 0 &\hookrightarrow 0 \\ \mathsf{f}\ n &\hookrightarrow n - 1 \end{aligned}$$

- f 0 reduces to both 0 and -1.

Pattern-Matching: Naive Translation

```
let f (x) =
  match x with
  | 0 -> 0
  | n -> n-1;
```

$$\begin{aligned} \mathsf{f} &: \mathbb{Z} \to \mathbb{Z} \\ \mathsf{f}\ 0 &\hookrightarrow 0 \\ \mathsf{f}\ n &\hookrightarrow n - 1 \end{aligned}$$

- f 0 reduces to both 0 and -1.
- Inconsistency!: we can prove $0 = -1$

Pattern-Matching: Destructors

```
let f (x) =
  match x with
  | 0 -> 0
  | n -> n-1;
```

$\leadsto$

```
let f (x) =
  if x = 0 then 0 else
    let n = x in n - 1;
```

Pattern-Matching: Destructors

```
let f (x) =
  match x with
  | 0 -> 0
  | n -> n-1;
```

⇝

```
let f (x) =
  if x = 0 then 0 else
    let n = x in n - 1;
```

```
let f (x) =
  match x with
  | 0 -> 0
  | S(n) -> n;
```

⇝

```
let f (x) =
  match x with
  | 0 -> 0
  | _ -> match x with
         | S(n) -> n
         | _ -> ERROR;
```

Recursion: Naive Translation

```
let rec fact (x) =
  if x <= 1
   then 1
   else x * fact(x - 1)
```

Recursion: Naive Translation

```
let rec fact (x) =
  if x <= 1
   then 1
   else x * fact(x - 1)
```

$$\mathsf{fact} : \mathbb{Z} \to \mathbb{Z}$$

$$\mathsf{fact}\ x \hookrightarrow \texttt{if}\ (x \leq 1)\ \texttt{then}\ 1\ \texttt{else}\ (x \times \mathsf{fact}(x - 1))$$

Recursion: Naive Translation

$$\forall x.\ \mathsf{fact}(x) > 0 \hookrightarrow$$

Recursion: Naive Translation

$$\forall x.\ \mathsf{fact}(x) > 0 \hookrightarrow$$
$$\forall x.\ (\mathsf{if}\ x \leq 1\ \mathsf{then}\ 1\ \mathsf{else}\ x \times \mathsf{fact}(x-1)) > 0 \hookrightarrow$$

Recursion: Naive Translation

$$\forall x.\ \mathsf{fact}(x) > 0 \hookrightarrow$$
$$\forall x.\ (\mathsf{if}\ x \leq 1\ \mathsf{then}\ 1\ \mathsf{else}\ x \times \mathsf{fact}(x-1)) > 0 \hookrightarrow$$
$$\forall x.\ (\ldots \mathsf{fact}(x-2)\ldots) > 0 \hookrightarrow$$
$$\ldots$$

Recursion

```
let rec fact (x) =
  if x <= 1
   then 1
   else x * fact(x - 1)
```

$\text{fact} : \mathbb{Z} \to \mathbb{Z}.$

$\text{fact'} : \mathbb{Z} \to \mathbb{Z}.$

$\text{fact'}\ \langle m, n \rangle \hookrightarrow \text{fact}\ \langle m, n \rangle$

$\text{fact}\ x \hookrightarrow$

$\quad \texttt{if}\ (x \leq 1)$

$\quad\quad \texttt{then}\ 1$

$\quad\quad \texttt{else}\ (x \times \text{fact'}(x - 1))$

Evaluation: Focalide Coverage

Library	Size	Coverage (%)
stdlib	160kB	99.42
extlib	155kB	100
contribs	124kB	99.54
iterators	78.4kB	88.33
term-proof	24.4kB	99.62
ejcp	13.7kB	95.16

Evaluation: Time (s)

Library	Zen	ZM	Coq	Dk	Zen + Coq	ZM + Dk
stdlib	11.73	32.87	17.41	1.46	29.14	34.33
extlib	9.48	26.50	19.45	1.64	28.93	28.14
contribs	5.38	9.96	26.92	1.17	32.30	11.13
iterators	2.58	3.85	6.59	0.27	9.17	4.12
term-proof	1.10	0.55	24.54	0.02	25.64	0.57
ejcp	0.44	0.86	11.13	0.06	11.57	0.92

Evaluation: Time (s)

Library	Zen	ZM	Coq	Dk	Zen + Coq	ZM + Dk
`stdlib`	11.73	32.87	17.41	1.46	29.14	34.33
`extlib`	9.48	26.50	19.45	1.64	28.93	28.14
`contribs`	5.38	9.96	26.92	1.17	32.30	11.13
`iterators`	2.58	3.85	6.59	0.27	9.17	4.12
`term-proof`	1.10	0.55	24.54	0.02	25.64	0.57
`ejcp`	0.44	0.86	11.13	0.06	11.57	0.92

- Zenon Modulo is slower (?)

Evaluation: Time (s)

Library	Zen	ZM	Coq	Dk	Zen + Coq	ZM + Dk
stdlib	11.73	32.87	17.41	1.46	29.14	34.33
extlib	9.48	26.50	19.45	1.64	28.93	28.14
contribs	5.38	9.96	26.92	1.17	32.30	11.13
iterators	2.58	3.85	6.59	0.27	9.17	4.12
term-proof	1.10	0.55	24.54	0.02	25.64	0.57
ejcp	0.44	0.86	11.13	0.06	11.57	0.92

- Zenon Modulo is slower (?)
- Dedukti is faster

Evaluation: Time (s) – Computation-intensive proof

n	Zenon	Coq	Zenon Modulo	Dedukti
10	31.48	4.63	0.04	0.00
11	63.05	11.04	0.04	0.00
12	99.55	7.55	0.05	0.00
13	197.80	10.97	0.04	0.00
14	348.87	1020.67	0.04	0.00
15	492.72	1087.13	0.04	0.00
16	724.46	$>$ 2h	0.04	0.00
17	1111.10	1433.76	0.04	0.00
18	1589.10	>2h	0.07	0.00
19	2310.48	>2h	0.04	0.00
...	...	...	...	...
2000	>2h	>2h	0.87	1.82

Contents

Functional Implementation of the Sieve of Eratosthenes

```
let rec interval a b =
   if a > b then [] else a :: interval (a+1) b

let rec sieve = function
  | [] -> []
  | a :: l ->
     a :: sieve (List.filter (fun b -> b mod a > 0) l)

let eratosthenes n = sieve (interval 2 n)
```

Correctness Proof

Correctness: $p \in \texttt{eratosthenes}\ n \Leftrightarrow (2 \leq p \leq n \wedge p \in \mathbb{P})$

- Completeness: $(2 \leq p \leq n \wedge p \in \mathbb{P}) \Rightarrow p \in \texttt{eratosthenes}\ n$
- Soundness 1: $p \in \texttt{eratosthenes}\ n \Rightarrow 2 \leq p \leq n$
- Soundness 2: $p \in \texttt{eratosthenes}\ n \Rightarrow p \in \mathbb{P}$
  - `List.filter` preserves ordering
  - $2 \leq p \leq n$ (Soundness 1)
  - there is a prime number d divising p
    - $d \leq p \leq n$
    - $d \in \texttt{eratosthenes}\ n$ (Completeness)
    - if $d = p$ then $p \in \mathbb{P}$
    - if $d < p$ then d is before p in `eratosthenes` n, absurd

Correctness Proof

Correctness: $p \in \texttt{eratosthenes}\ n \Leftrightarrow (2 \leq p \leq n \wedge p \in \mathbb{P})$

- Completeness: $(2 \leq p \leq n \wedge p \in \mathbb{P}) \Rightarrow p \in \texttt{eratosthenes}\ n$
- Soundness 1: $p \in \texttt{eratosthenes}\ n \Rightarrow 2 \leq p \leq n$
- Soundness 2: $p \in \texttt{eratosthenes}\ n \Rightarrow p \in \mathbb{P}$
  - `List.filter` preserves ordering
  - $2 \leq p \leq n$ (Soundness 1)
  - there is a prime number d divising p
    - $d \leq p \leq n$
    - $d \in \texttt{eratosthenes}\ n$ (Completeness)
    - if $d = p$ then $p \in \mathbb{P}$
    - if $d < p$ then d is before p in $\texttt{eratosthenes}\ n$, absurd

Coq-HOL Correctness Proof

- HOL
  - $\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
- Coq
  - Implementation
  - Specification
  - Correctness
- Dedukti
  - Logical linking
- FoCaLiZe
  - Mathematical linking

Scheme

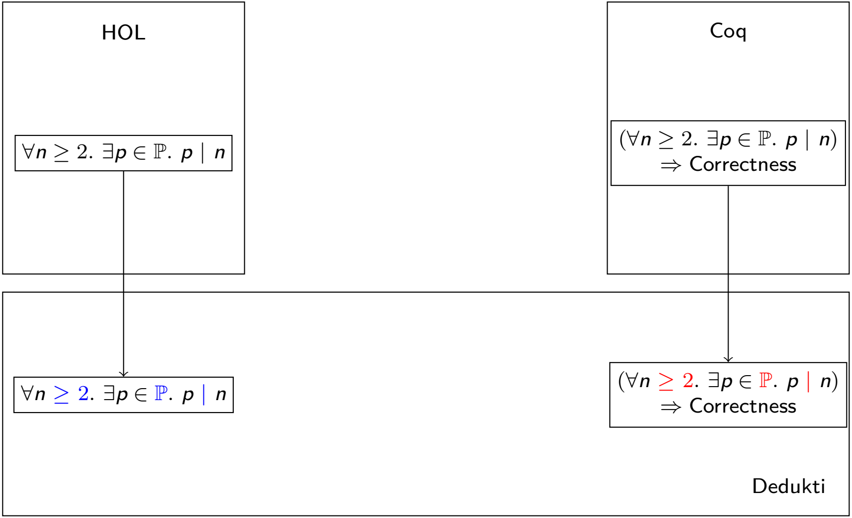
HOL

$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$

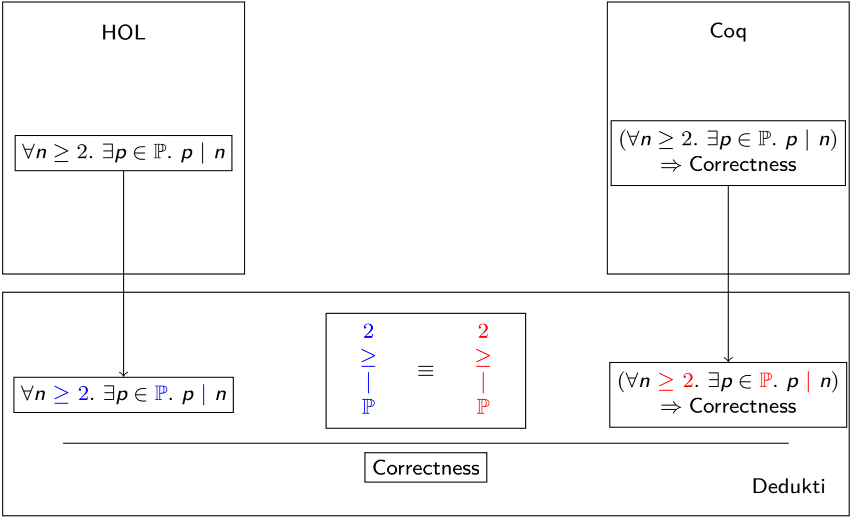
Coq

$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness

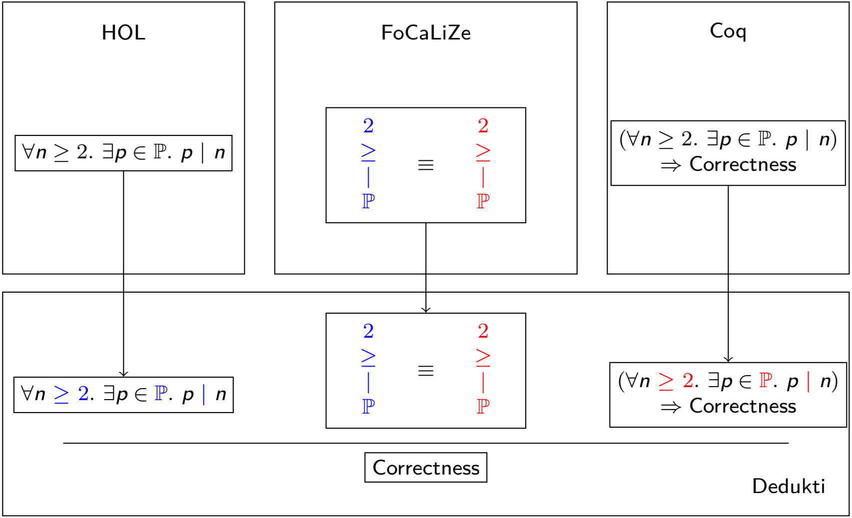
Scheme


HOL
Coq
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness
Dedukti

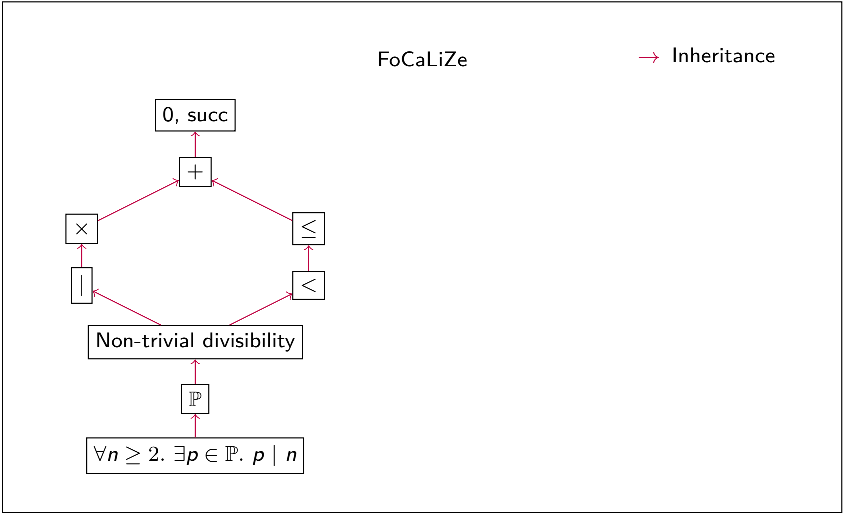
Scheme


HOL
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
Coq
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
2 ≥ | ℙ ≡ 2 ≥ | ℙ
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness
Correctness
Dedukti

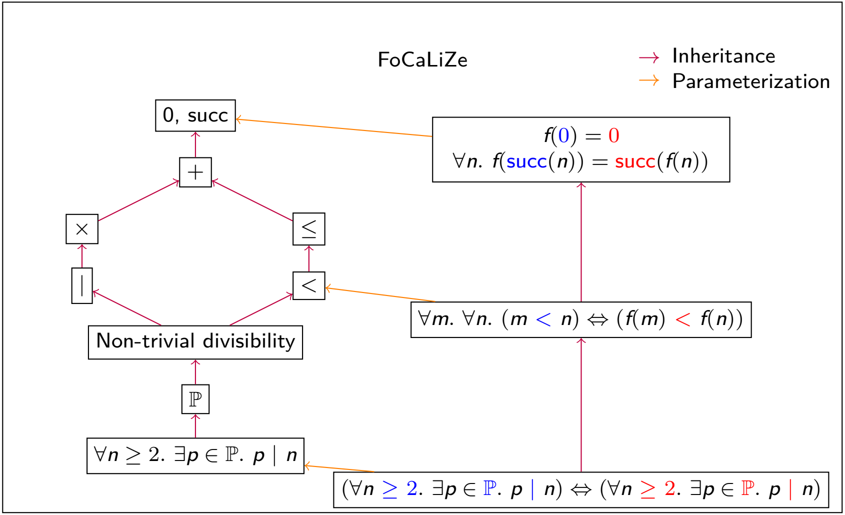
Scheme


HOL
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
FoCaLiZe
$2 \ \geq \ \mid \ \mathbb{P} \equiv 2 \ \geq \ \mid \ \mathbb{P}$
Coq
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n) \Rightarrow$ Correctness
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
$2 \ \geq \ \mid \ \mathbb{P} \equiv 2 \ \geq \ \mid \ \mathbb{P}$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n) \Rightarrow$ Correctness
Correctness
Dedukti

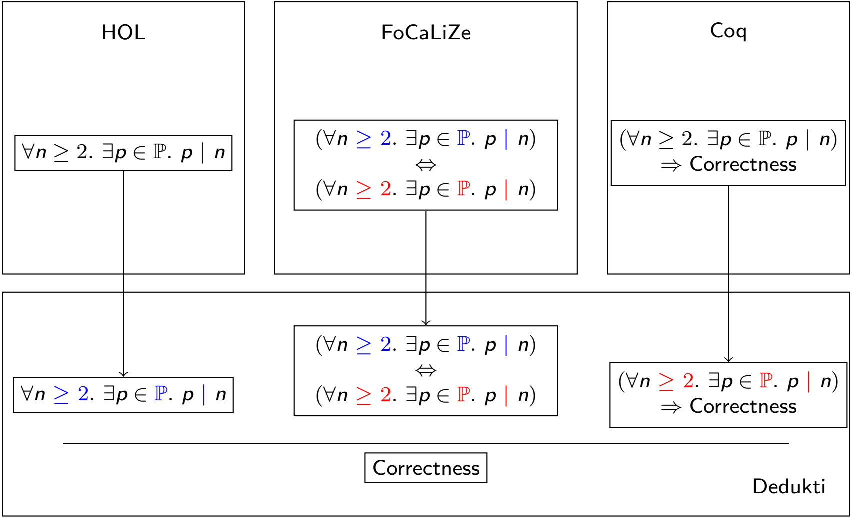
Arithmetical Structures



Morphisms


FoCaLiZe
Inheritance
Parameterization
0, succ
$f(0) = 0$
$\forall n.\ f(\mathsf{succ}(n)) = \mathsf{succ}(f(n))$
$+$
$\times$
$\leq$
$|$
$<$
$\forall m.\ \forall n.\ (m < n) \Leftrightarrow (f(m) < f(n))$
Non-trivial divisibility
$\mathbb{P}$
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n) \Leftrightarrow (\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$

Scheme


HOL
FoCaLiZe
Coq
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Leftrightarrow$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness
$\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Leftrightarrow$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$(\forall n \geq 2.\ \exists p \in \mathbb{P}.\ p \mid n)$
$\Rightarrow$ Correctness
Correctness
Dedukti

Size

	Source	Generated
HOL	3.4M	64M
Coq	31K	829K
FoCaLiZe	61K	495K
Zenon Modulo		886K
Dedukti	9K	

Results

- Working interoperability proof of concept
- Genericity
- Scalability

Conclusion

- Translations to Dedukti of 2 programming paradigms
- Deduction Modulo in FoCaLiZe
- Working example of interoperability
- General methodology based on FoCaLiZe object-oriented mechanisms
- Deviant usage of the tools
  - FoCaLiZe as interoperability framework
  - Dedukti as proof constructivizer ([*A rewrite system for proof constructivization*, LFMTP 2016])

Perspectives

- Axiom elimination
    - Constructivization
    - Extensionality and choice
    - Univalence
- More translations for logical systems
- More translations for programming languages
- Interoperability for bigger examples, involving more systems

Thank you!

Meta-Level Reasoning

- Beyond proof checking, proof transformation

$$
\begin{aligned}
\mathsf{Prop} &: \mathbf{Type} \\
\top &: \mathsf{Prop} \\
\neg &: \mathsf{Prop} \to \mathsf{Prop} \\
\wedge &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vee &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}
\end{aligned}
$$

Meta-Level Reasoning

- Beyond proof checking, proof transformation

$$
\begin{aligned}
\mathrm{Prop} &: \mathbf{Type} \\
\top &: \mathrm{Prop} \\
\neg &: \mathrm{Prop} \to \mathrm{Prop} \\
\wedge &: \mathrm{Prop} \to \mathrm{Prop} \to \mathrm{Prop} \\
\vee &: \mathrm{Prop} \to \mathrm{Prop} \to \mathrm{Prop} \\
\vdash &: \mathrm{Prop} \to \mathbf{Type}
\end{aligned}
$$

Meta-Level Reasoning

- Beyond proof checking, proof transformation

$$
\begin{aligned}
\mathsf{Prop} &: \textbf{Type} \\
\top &: \mathsf{Prop} \\
\neg &: \mathsf{Prop} \to \mathsf{Prop} \\
\wedge &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vee &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\
\vdash &: \mathsf{Prop} \to \textbf{Type} \\
\mathsf{cm} &: \Pi P : \mathsf{Prop}.\ ((\vdash \neg P) \to (\vdash P)) \to (\vdash P)
\end{aligned}
$$

Meta-Level Reasoning

- Beyond proof checking, proof transformation

$$
\begin{array}{ll}
\text{Prop} & : \textbf{Type} \\
\top & : \text{Prop} \\
\neg & : \text{Prop} \to \text{Prop} \\
\wedge & : \text{Prop} \to \text{Prop} \to \text{Prop} \\
\vee & : \text{Prop} \to \text{Prop} \to \text{Prop} \\
\vdash & : \text{Prop} \to \textbf{Type} \\
\text{cm} & : \Pi P : \text{Prop}.\ ((\vdash \neg P) \to (\vdash P)) \to (\vdash P) \\
\text{cm}\ P\ (\lambda\_.\ H_P) & \hookrightarrow H_P
\end{array}
$$

Classical proof → Dedukti → Constructive (?) proof

[*A rewrite system for proof constructivization*, LFMTP 2016]