

Use of two proof assistants in an introduction to proof course: an experiment

Frederic Tran-Minh

`frederic.tran-minh@grenoble-inp.fr`

Esisar, Valence, France

ThEdu23, Rome, Italy - July 5th, 2023

Contents

- 1 Two proof assistants : why ?
- 2 Edukera and Lean3
- 3 Pedagogical setup and choices
- 4 A quick poll
- 5 Questions?
- 6 Bibliography

Contents

- 1 Two proof assistants : why ?
- 2 Edukera and Lean3
- 3 Pedagogical setup and choices
- 4 A quick poll
- 5 Questions?
- 6 Bibliography

Where does it all come from ?

- \approx 10 years as dev engineer
- \approx 10 years as teacher (to undergraduates (Esisar engineering school), in maths, C programming)
- A feeling : many students misunderstand the math game we want them to play...¹
- Why do they seem they quicker, better, more motivated at learning programming ? Feedback could be a key...
- Good news : Proof assistants (PA) (ITP) exist...

¹Moore 1994; Weber 2001.

Proof assistants for teaching

- Pedagogical use : mostly in
 - computer science, Hoare logic²
 - formal logics,³
 - geometry⁴
- Few reports of use in undergraduate maths⁵
- Professional PA or tutors ? Controlled natural language⁶
- in Hanna and Yan [2021](#) :
 - Not widespread in this context

“A few universities have offered courses on the use of digital proof assistants, but undergraduate mathematics curricula have neglected them”
 - Few didactical studies

“there does not appear to be any published systematic research that has explored their potential in any educational context”

²Pierce et al. [2017](#); Delahaye, Jaume, and Prevosto [2005](#); Nipkow [2012](#).

³Avigad [2019](#).

⁴Webber et al. [2001](#); Gressier [2006](#); Leduc [2016](#).

⁵Blanc et al. [2007](#); Kerjean et al. [2022](#).

⁶Kerjean et al. [2022](#); Wemmenhove et al. [2022](#).

Two proof assistants

- Primary naïve question (hope!) : would the use of PA in the classroom improve math proving skills ?
- Doubts :
 - Appropriation of a new language (difficulty, effort, time, benefit)
 - Transition to pen/paper proofs
- Different characteristics of PA could impact differently proof learning (Bartzia, Meyer, and Narboux 2022)
- During one semester, the same group of students trained with two different proof assistants
 - Lean3
 - Edukera

Contents

- 1 Two proof assistants : why ?
- 2 **Edukera and Lean3**
- 3 Pedagogical setup and choices
- 4 A quick poll
- 5 Questions?
- 6 Bibliography

Lean3 and Edukera

Comparison

according to some criteria from (Bartzia, Meyer, and Narboux 2022)

	Lean3	Edukera
Type of PA	Professional	Educational
Type of user input	text-based, strict syntax	point-and-click
Imperative/Declarative	Imperative (in tactics mode)	imp input, decl output
Math Lib	Comprehensive	Micro-worlds
Create new defs/lemmas?	Yes	No

Edukera (formalization)

“ A player must be registered in order to play and any beaten player is eliminated ”

Formaliser la phrase suivante :

Un joueur doit être inscrit pour pouvoir jouer et tout joueur battu est éliminé.

$(\forall x, ((\exists y, j(x, y)) \Rightarrow i(x)) \wedge \forall x, ((\exists y, b(x, y)) \Rightarrow e(x)))$ ✓

Les prédicats disponibles sont :

$j(x, y)$	x a joué contre y
$b(x, y)$	x a battu y
$i(x)$	x est inscrit au tournoi
$e(x)$	x est éliminé du tournoi

Les constantes disponibles sont :

A	Alain
B	Bernard

Edukera (formalization)

“ A player must be registered in order to play and any beaten player is eliminated ”

Formaliser la phrase suivante :

Un joueur doit être inscrit pour pouvoir jouer et tout joueur battu est éliminé.

$(\forall x, ((\exists y, j(x, y)) \Rightarrow i(x)) \wedge \forall x, ((\exists y, b(y, x)) \Rightarrow e(x)))$ ✓

Les prédicats disponibles sont :

$j(x, y)$	x a joué contre y
$b(x, y)$	x a battu y
$i(x)$	x est inscrit au tournoi
$e(x)$	x est éliminé du tournoi

Les constantes disponibles sont :

A	Alain
B	Bernard

A simple example

Definition

Let E et F be two sets. Let f be a map from E to F . f is said to be injective if and only id $\forall x \in E, \forall y \in E, f(x) = f(y) \implies x = y$

Theorem

Let E , F and G be three sets, and $f : E \rightarrow F$ et $g : F \rightarrow G$ two maps. If $g \circ f$ is injective, then f is injective.

Proof.

Suppose that $g \circ f$ is injective. Let x and y be elements of E such that $f(x) = f(y)$. Then we have $g(f(x)) = g(f(y))$. Therefore, it follows from the injectivity of $g \circ f$ that $x = y$. \square

Edukera (proof)

	<i>Soit</i> l'ensemble C	<i>déclaration</i>
	<i>Soit</i> $f : A \rightarrow B$	<i>déclaration</i>
	<i>Soit</i> $g : B \rightarrow C$	<i>déclaration</i>
(1)	$g \circ f$ est injective	<i>hypothèse</i>
	<div style="border-left: 1px solid black; padding-left: 10px;"> <p><i>Soit</i> $x \in A$</p> <p><i>Soit</i> $y \in A$</p> </div>	<i>déclaration</i>
(2)	$f(x) = f(y)$	<i>hypothèse</i>
(3)	$g(f(x)) = g(f(y))$	(2) <i>par application de g à gauche et à droite</i>
(4)	$g(f(x)) = g \circ f(y)$	(3) <i>par définition de la composition de fonctions</i>
(5)	$g \circ f(x) = g \circ f(y)$	(4) <i>par définition de la composition de fonctions</i>
(6)	$x = y$	(1) (5) <i>par définition d'une fonction injective</i>
(7)	f est injective	$x \dots$ (6) <i>par définition d'une fonction injective</i>

Edukera (proof)

	<i>Soit</i> l'ensemble C	<i>déclaration</i>
	<i>Soit</i> $f : A \rightarrow B$	<i>déclaration</i>
	<i>Soit</i> $g : B \rightarrow C$	<i>déclaration</i>
(1)	$g \circ f$ est injective	<i>hypothèse</i>
	<div style="border-left: 1px solid black; padding-left: 10px;"> <p><i>Soit</i> $x \in A$</p> <p><i>Soit</i> $y \in A$</p> </div>	<i>déclaration</i>
(2)	$f(x) = f(y)$	<i>hypothèse</i>
(3)	$g(f(x)) = g(f(y))$	(2) <i>par application de g à gauche et à droite</i>
(4)	$g(f(x)) = g \circ f(y)$	(3) <i>par définition de la composition de fonctions</i>
(5)	$g \circ f(x) = g \circ f(y)$	(4) <i>par définition de la composition de fonctions</i>
(6)	$x = y$	(1) (5) <i>par définition d'une fonction injective</i>
(7)	f est injective	$x \dots$ (6) <i>par définition d'une fonction injective</i>

Lean (proof)

The screenshot shows the Lean IDE interface. The main editor displays a Lean script with the following content:

```

78_small_example_injectivity.lean > _
variables {E F G : Type}
definition injective (f : E → F) : Prop := ∀ (u:E), ∀ (v:E), ( f u = f v → u = v )

theorem ex_6_4_21_1' : ∀ (f : E → F) (g : F → G), (injective (g ∘ f)) → (injective f) :=
  assume (f : E → F) (g : F → G),
  assume (h1 : injective (g ∘ f)),
  assume (x y : E),
  assume (h2 : f x = f y),
  have h3 : g (f x) = g (f y), from congr_arg g h2,
  h1 x y h3

```

On the right side, the 'Lean Infoview' panel shows the 'Tactic state' for the current goal:

```

▼ 78_small_example_injectivity.lean:24:12
▼ Tactic state
expected type:
E F G : Type
f : E → F
g : F → G
h1 : injective (g ∘ f)
x y : E
h2 : f x = f y
h3 : g (f x) = g (f y)
┆ x = y
► All Messages (0)

```

Lean (proof)

The screenshot shows the Lean IDE interface. The main editor displays a Lean script with the following content:

```

78_small_example_injectivity.lean > ...
variables {E F G : Type}
definition injective (f : E → F) : Prop := ∀ (u:E), ∀ (v:E), ( f u = f v → u = v )

theorem ex_6_4_21_1' : ∀ (f : E → F) (g : F → G), (injective (g ∘ f)) → (injective f) :=
  assume (f : E → F) (g : F → G),
  assume (h1 : injective (g ∘ f)),
  assume (x y : E),
  assume (h2 : f x = f y),
  have h3 : g (f x) = g (f y), from congr_arg g h2,
  h1 x y h3

```

On the right side, the 'Lean Infoview' panel shows the 'Tactic state' for the current goal:

```

▼ 78_small_example_injectivity.lean:24:12
▼ Tactic state
expected type:
E F G : Type
f : E → F
g : F → G
h1 : injective (g ∘ f)
x y : E
h2 : f x = f y
h3 : g (f x) = g (f y)
┆ x = y
► All Messages (0)

```

Contents

- 1 Two proof assistants : why ?
- 2 Edukera and Lean3
- 3 Pedagogical setup and choices
- 4 A quick poll
- 5 Questions?
- 6 Bibliography

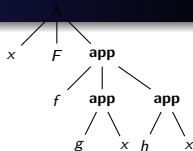
Pedagogical organization

- 64 1st year undergraduates in engineering school
- 6 x 1,5h sessions (in tutorial groups of 32) over 1 semester
- mandatory
- Autonomous use of Edukera
- Lean c.a. 25% lecturing + 75 % practical work with handout
- Evaluation : Edukera Achievement + paper test (MCQ) + Machine Test
- https://github.com/ftranminh/Esisar_MA121_HA_lean_2023



Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

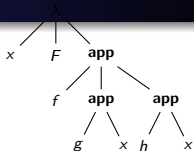
```
assume (f:E→F) (g:F→G),
assume (h1:injective (gof)),
  assume (x y:E),
    assume (h2: f x = f y),
      have h3: g (f x) = g (f y),
        from congr_arg g h2,
      h1 x y h3
```

Tactic mode

```
begin
  intros f g h1 x y h2,
  apply h1,
  simp [function.comp],
  apply congr_arg g,
  assumption
end
```

Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

```
assume (f:E→F) (g:F→G),
assume (h1:injective (gof)),
  assume (x y:E),
    assume (h2: f x = f y),
      have h3: g (f x) = g (f y),
        from congr_arg g h2,
      h1 x y h3
```

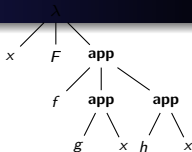
Tactic mode

```
begin
  intros,
  unfold injective at *,
  intros,
  finish,
end
```

More automation

Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

```
assume (f:E→F) (g:F→G),
assume (h1:injective (gof)),
  assume (x y:E),
    assume (h2: f x = f y),
      have h3: g (f x) = g (f y),
        from congr_arg g h2,
      h1 x y h3
```

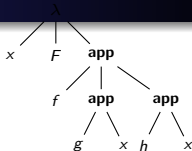
Tactic mode

```
begin
  intros f g h1 x y h2,
  have h3 : g (f x) = g (f y) := by tauto
  tauto,
end
```

Adjust automation

Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

```
assume (f:E  $\rightarrow$  F) (g:F  $\rightarrow$  G),
assume (h1:injective (g  $\circ$  f)),
assume (x y:E),
  assume (h2: f x = f y),
    have h3: g (f x) = g (f y),
      from congr_arg g h2,
    h1 x y h3
```

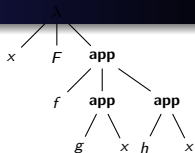
Tactic mode

```
begin
  intros f g h1 x y h2,
  have h3:g (f x)=g (f y):=by congr',
  apply h1,
  assumption
end
```

Adjust automation

Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

```
assume (f:E→F) (g:F→G),
assume (h1:injective (gof)),
  assume (x y:E),
    assume (h2: f x = f y),
      have h3: g (f x) = g (f y),
        from congr_arg g h2,
      h1 x y h3
```

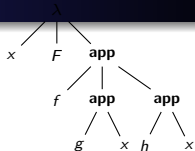
Tactic mode

```
begin
  intro f,
  intro g,
  intro h1,
  intros x y,
  intro h2,
  have h3:g (f x)=g (f y):=by congr
  ',
  apply h1,
  assumption
end
```

More verbose...

Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

```
assume (f:E→F) (g:F→G),
assume (h1:injective (g∘f)),
  assume (x y:E),
    assume (h2: f x = f y),
      have h3: g (f x) = g (f y),
        from congr_arg g h2,
      h1 x y h3
```

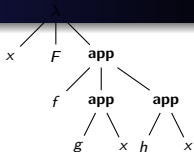
Tactic mode

```
begin
  assume (f:E→F),
  assume (g:F→G),
  assume (h1:injective (g ∘ f)),
  assume (x y:E),
  assume (h2:f x=f y),
  have h3:g (f x) = g (f y):=by
    congr',
  apply h1,
  assumption
end
```

More and more verbose...

Lean: Term-mode and Tactic-mode

- Term-mode : a proof is a λ -term ; carries a structure, e.g $\lambda x : F, f (g x) (h x)$
- Tactic mode : a proof is a list of orders that manipulate goal and hypothesis ; intrinsically sequential ; mostly used for teaching (AFAIK)



```
theorem T1 :  $\forall (f:E \rightarrow F) (g:F \rightarrow G), (\text{injective } (g \circ f)) \rightarrow (\text{injective } f) :=$ 
```

Term mode

```
assume (f:E→F) (g:F→G),
assume (h1:injective (g∘f)),
assume (x y:E),
  assume (h2: f x = f y),
  have h3: g (f x) = g (f y),
    from congr_arg g h2,
  h1 x y h3
```

Tactic mode

```
begin
  assume (f:E→F) (g:F→G),
  assume (h1:injective (g ∘ f)),
  assume (x y:E),
  assume (h2:f x=f y),
  have h3:g (f x)=g (f y),
    from congr_arg g h2,
  apply h1 x y h3
end
```

More and more verbose...

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
 - Proofs on sets, applications
- Lean
 - Propositional Logic
 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)
 - Numerical sequences (limits)

Example

Not all novels were written by the same person

Tous les romans n'ont pas été écrits par une même personne.

$\neg (\exists x, (H(x) \wedge \forall r, (R(r) \Rightarrow E(x, r))))$

Les prédicats disponibles sont :

$L(x, y)$	x a lu y
$E(x, y)$	x a écrit y
$R(x)$	x est un roman
$H(x)$	x est un être humain

Les constantes disponibles sont :

A	Anaïs
VH	Victor Hugo
LM	Les Misérables

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
- Proofs on sets, applications
- Lean
 - Propositional Logic
 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)
 - Numerical sequences (limits)

Example

f is increasing

La fonction f n'est pas croissante sur l'intervalle I.

$\exists x, \exists y, (((x \in I \wedge y \in I) \wedge x \leq y) \wedge \neg (f(x) \leq f(y)))$

La formule ne doit pas démarrer par une négation.

Les prédicats disponibles sont :

$x \leq y$	x est inférieur ou égal à y
$x \in y$	x appartient à y

Les opérateurs disponibles sont :

$x(y)$	x (y)
--------	-------

Les constantes disponibles sont :

f	f
I	I

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
 - Proofs on sets, applications
- Lean
 - Propositional Logic
 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)
 - Numerical sequences (limits)

Example

Prove $g \circ f$ injective $\implies f$ injective

The screenshot shows the Edukera proof assistant interface. The main window displays a formal proof in French. The proof is structured as follows:

$\text{Soit } f : A \rightarrow B$	déclaration
$\text{Soit } g : B \rightarrow C$	déclaration
(1) $g \circ f$ est injective	hypothèse
$\text{Soit } x \in A$	déclaration
$\text{Soit } y \in A$	déclaration
(2) $f(x) = f(y)$	hypothèse
(3) $g(f(x)) = g(f(y))$	(2) par application de g à gauche et à droite
(4) $g \circ f(x) = g \circ f(y)$	(3) par définition de la composition de fonctions
(5) $g \circ f(x) = g \circ f(y)$	(4) par définition de la composition de fonctions
(6) $x = y$	(1) (5) par définition d'une fonction injective
(7) f est injective	$x \dots$ (6) par définition d'une fonction injective

At the bottom right of the interface, there is a blue button labeled "Suivant" (Next) with a checkmark icon.

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
 - Proofs on sets, applications
- Lean
 - Propositional Logic

 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)
 - Numerical sequences (limits)

Example

Prove $(A \vee B) \iff ((A \implies B) \implies B)$

```
theorem T:∀ A B:Prop, A∨B ↔ (A→B)→B :=
  assume A B : Prop,
  iff.intro
    (assume h_A_or_B : A ∨ B ,
     assume h_A_imp_B : A → B,
     or.elim h_A_or_B
     (assume h_A : A,
      show B, from h_A_imp_B h_A
     )
     (assume h_B : B,
      show B, from h_B
     )
    )
  (assume h_A_imp_B_imp_B:(A→B)→B,
   have h_A_or_not_A : A∨¬A, from
     classical.em A,
   or.elim h_A_or_not_A
   (assume h_A : A,
    show A ∨ B, from or.inl h_A
   )
   (assume h_not_A : ¬A,
    have h_A_imp_B : A→B, from
      assume h_A : A,
      show B, from absurd h_A h_not_A,
    have h_B : B, from
      h_A_imp_B_imp_B h_A_imp_B,
    show A ∨ B, from or.inr h_B
   )
  )
```

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
 - Proofs on sets, applications
- Lean

 - Propositional Logic
 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)
 - Numerical sequences (limits)

Example

Prove: if m, n are even then $m + n$ is even

```
def even (n:N):Prop:=∃(k:N),n=2*k
lemma S:∀ m n:N,(even m)^(even n)
      →¬(even (m+n)) :=
assume (m n:N),
assume h1: (even m) ^ (even n),
exists.elim h1.left (
  assume j:N , assume h2: m=2*j,
  exists.elim h1.right (
    assume k:N , assume h2: n=2*k,
    exists.intro (j+k) (
      calc
        m+n=2*j+n :congr_arg (λ z, z+n) h1
        ...=2*j+2*k:congr_arg (λ z, 2*j+z) h2
        ...=2*(j+k):(mul_add 2 j k).symm
      )
    )
  )
)
```

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
 - Proofs on sets, applications

- Lean
 - Propositional Logic
 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)

 - Numerical sequences (limits)

Example

Prove: $g \circ f$ injective $\implies f$ injective

```
variables {E F G : Type}
def injective (f:E→F):Prop:=
  ∀u v:E, f u=f v→u=v
theorem T:
  ∀(f:E→F)(g:F→G),injective (g ∘ f)
    → injective f
:=
assume (f:E→F)(g:F→G),
assume (h1:injective (gof)),
assume (x y: E),
assume (h2:f x = f y),
have h3: g (f x)=g (f y),
  from congr_arg g h2,
show x=y,
  from h1 x y h3
```

Topics

- Edukera
 - Formalization of sentences in French
 - Formalization of math statements
 - Proofs on sets, applications
- Lean
 - Propositional Logic
 - 1st order logic (\mathbb{N})
 - Maps (surjectivity, injectivity, preimage,...)
 - Numerical sequences (limits)

Example

Prove that the sum of two convergent sequences converges to the sum of their limits.

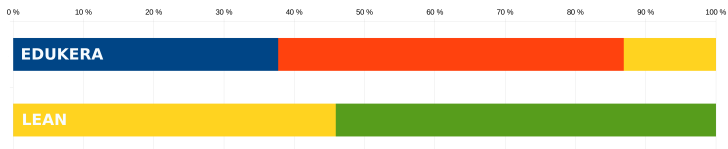
```
def seq := N → ℝ
definition seqadd (u:seq)(v:seq):seq:=
  λ n,(u n) + (v n)
def cv (u:seq)(l:ℝ):Prop:=
  ∀ε:ℝ,ε>0→∃n0:N,∀n:N,n≥n0→|u n-l|≤ε
theorem sumlim (u v:seq) (l l':ℝ):
  ((cv u l)∧(cv v l'))→(cv (u+v) (l+l'))
  :=
  (assume (h1: (cv u l) ∧ (cv v l')),
   assume (ε:ℝ) (hε: ε>0),
   have hε2:(0<(ε/2)),from
     (by simp:(0/2)=(0:ℝ))
     ►(lt_div2 (0:ℝ) ε hε),
   exists.elim (h1.left (ε/2) hε2)
   (assume n0:N,assume h2: ∀ n:N, n≥n0→|u n-
     l|≤ε/2,
   exists.elim (h1.right (ε/2) hε2)
   (assume n1:N,assume h3:∀ n:N,n≥n1→|v n-
     l'|≤ε/2,
   let n2:=(max n0 n1) in exists.intro n2
   assume n:N (h4:n≥n2),
   calc
     |(u+v) n-(l+l')| = |(u n-l)+(v n-l')| : b
     abel
     ... ≤ |(u n-l)|+|(v n-l')| : abs_add
```

Contents

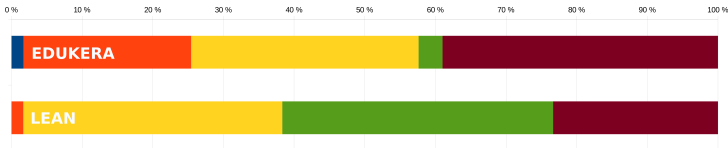
- 1 Two proof assistants : why ?
- 2 Edukera and Lean3
- 3 Pedagogical setup and choices
- 4 A quick poll**
- 5 Questions?
- 6 Bibliography

Quick poll : how difficult did they perceive it?

Getting started



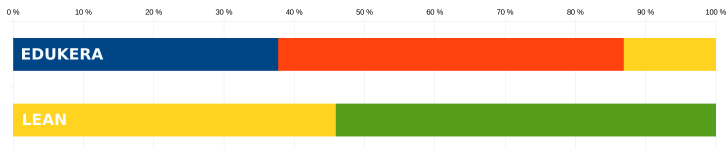
Difficulty of exercises



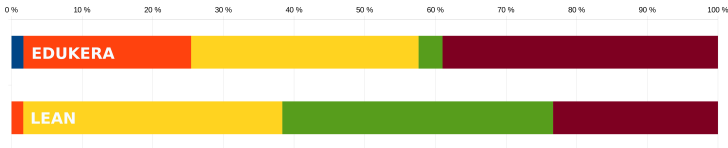
■ Very Easy
 ■ Rather Easy
 ■ Rather Difficult
 ■ Very Difficult
 ■ Heterogeneous Difficulty

Quick poll : how difficult did they perceive it?

- Getting started



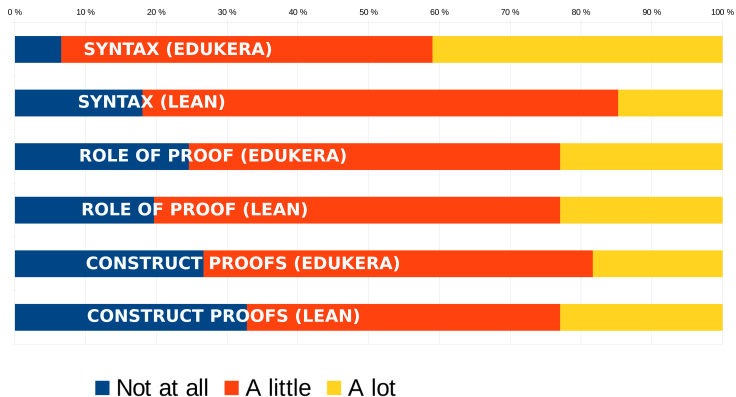
- Difficulty of exercises



clic-based interface: beyond subjective ease : some benefits,
some biases

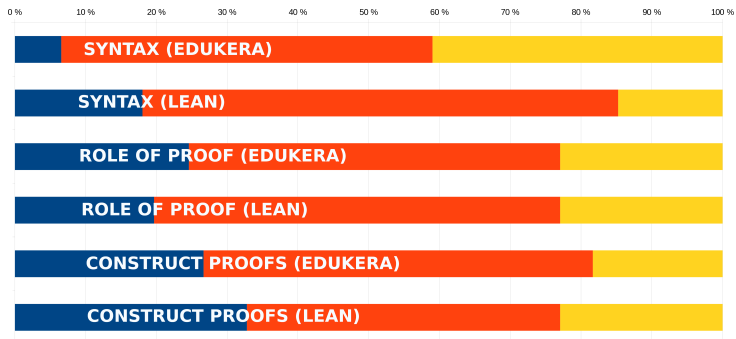
Quick poll : how much do they think it helped?

- How does it help for ...



Quick poll : how much do they think it helped?

- How does it help for ...



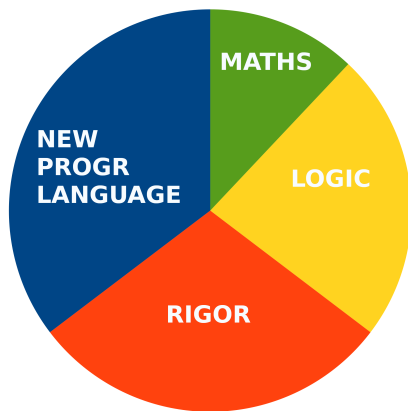
constrained input: could help to construct the idea of a rigid syntax?

Clic-based interface + Constrained input

- ⊕ : quick learning curve, helps to start a proof, to see what is expected, learning by habituation
- ⊖ : few initiative needed in constructing a proof, few opportunities to mistake⁷, possibility to click around without deep understanding
- ? Hybrid tool to accompany them during their first steps with formal proofs?

⁷Lakatos, 1963, Proof and Refutations

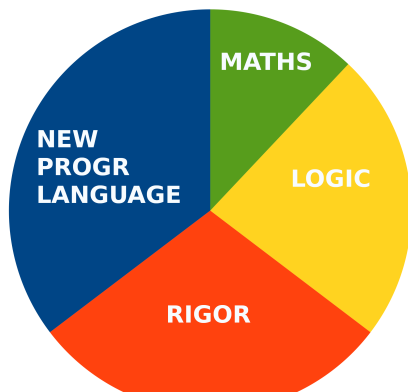
Quick poll : what do they relate their difficulties to ?



The possible difficulties encountered with Lean did they seem to you to be related to : (you may tick several boxes)

- Learning a new programming language
- The rigor required
- Logic skills (how to prove such a statement...)
- Mathematical skills (knowledge of definitions...)
- No difficulties encountered

Quick poll : what do they relate their difficulties to ?



The possible difficulties encountered with Lean did they seem to you to be related to : (you may tick several boxes)

- Learning a new programming language
- The rigor required
- Logic skills (how to prove such a statement...)
- Mathematical skills (knowledge of definitions...)
- No difficulties encountered

Many students attribute their difficulties to the **language syntax rather than mathematical shortcomings**

Distinction language skills vs math skills ?

- “I have a lot more trouble understanding how to write a proof in Lean than understanding what proof to provide”
- “I had great difficulty with the exercises because of my programming errors in Lean when I could do the mathematical proof”
- “It is sometimes annoying because of the syntax and the '()' or ',' which can make our proof wrong for Lean, when it was right”

This partially makes sense, but...

- what is a proof ? what is understand?
- Is it credible when (eg) they formalize def of surjectivity while injectivity is expected ?

Quick poll : Free comments

- Positive impact frequently explicitly reported (44 %)
 - live feedback (41 %)
 - autonomy allowed, self correction (11 %)
 - individualization
 - attention attracted to rigor
 - playful
- Drawbacks
 - **Feedback perceived as binary** (18 %)
 - Lean: difficulty of appropriation (syntax,...) (16 %)
- Asked for more lecturing, tutorial sessions (33 %)

Quick poll : Free comments

- Positive impact frequently explicitly reported (44 %)
 - live feedback (41 %)
 - autonomy allowed, self correction (11 %)
 - individualization
 - attention attracted to rigor
 - playful
- Drawbacks
 - **Feedback perceived as binary** (18 %)
 - Lean: difficulty of appropriation (syntax,...) (16 %)
- Asked for more lecturing, tutorial sessions (33 %)

Quick, even binary feedback perceived as motivating ; produce relevant feedback ?

Produce relevant feedback ?

- Professional PA is not a tutor: feedback is a compiler error message

```

variables {E F G:Type}
definition injective (f: E → F) : Prop := ∀ (u:E), ∀ (v:E), f u =f v → u=v

theorem ex_p_1 : Prop := ∀ f( E → F), ∀ g(F → G), injective f ∧ injective g → injective :=
assume (injective f injective g),
(
  assume (injective (f)),
  assume (injective (g)),
  assume (h:injective f ∧ injective g),
  show injective, from and.import h
)

```

⁸Carl, Lorenzen, and Schmitz 2022.

Produce relevant feedback ?

- Professional PA is not a tutor: feedback is a compiler error message

```

variables {E F G:Type}
definition injective (f:
theorem ex_p_1 : Prop
assume (injective f i
(
  assume (injective (f
    assume (injective
      assume (h:inject
        show injective

```

▼ serrej.lean:31:10
 theorems do not get VM compiled, use 'def' or add 'noncomputable' modifier to

▼ serrej.lean:31:81
 type expected at
 injective
 term has type
 (?m_1 → ?m_2) → Prop

▼ serrej.lean:31:91
 command expected

▼ All Messages (16)

▼ serrej.lean:10:22
 invalid field notation, type is not of the form (C ...) where C is a constant
 h
 has type
 P

⁸Carl, Lorenzen, and Schmitz 2022.

Produce relevant feedback ?

- Professional PA is not a tutor: feedback is a compiler error message

```

variables {E F G:Type}
definition injective (f:
theorem ex_p_1 : Prop
assume (injective f i
(
  assume (injective (f
    assume (injective
      assume (h:inject
        show injective

```

serrej.lean:31:10
 theorems do not get VM compiled, use 'def' or add 'noncomputable' modifier to

serrej.lean:31:81
 type expected at
 injective
 term has type
 (?m_1 → ?m_2) → Prop

serrej.lean:31:91
 command expected

All Messages (16)

serrej.lean:10:22
 invalid field notation, type is not of the form (C ...) where C is a constant
 h
 has type
 P

- Identify and produce different levels of relevant feedback ⁽⁸⁾

⁸Carl, Lorenzen, and Schmitz 2022.

Long term investment as alternative to Natural Controlled Language?

- Natural Controlled Language (ex: LeanVerbose, Waterproof)⁹
 - Could ease appropriation of the PA ?
 - Could help to make an efficient transition to pen/paper proof?
- Other approach
 - Conditions : Long Term Basis (1,2 years...) ; complete integration to regular math course ; CS students
 - May be worth to invest in learning the PA original language
 - Postulate : there are common skills involved in building a pen/paper proof and a formalized proof
 - How to help students relate both ?
 - In : Nipkow 2012

“Teaching Isabelle required the first quarter of the semester. We believe that this initial investment did not just improve the students’ understanding of the logical foundations, it also allowed us to cover the semantics material more quickly than normally because of the solid and uniform foundations that could be taken for granted.”

⁹Kerjean et al. 2022; Wemmenhove et al. 2022.

Contents

- 1 Two proof assistants : why ?
- 2 Edukera and Lean3
- 3 Pedagogical setup and choices
- 4 A quick poll
- 5 Questions?
- 6 Bibliography

Questions?

Contents



- 1 Two proof assistants : why ?
- 2 Edukera and Lean3
- 3 Pedagogical setup and choices
- 4 A quick poll
- 5 Questions?
- 6 Bibliography

Bibliography I



Avigad, Jeremy (2019). “Learning Logic and Proof with an Interactive Theorem Prover”. en. In: *Proof Technology in Mathematics Research and Teaching*. Ed. by Gila Hanna, David A. Reid, and Michael de Villiers. Vol. 14. Series Title: Mathematics Education in the Digital Era. Cham: Springer International Publishing, pp. 277–290. ISBN: 978-3-030-28482-4 978-3-030-28483-1. DOI: [10.1007/978-3-030-28483-1_13](https://doi.org/10.1007/978-3-030-28483-1_13). URL: http://link.springer.com/10.1007/978-3-030-28483-1_13 (visited on 03/30/2023).




Bibliography II

-  Bartzia, Evmorfia, Antoine Meyer, and Julien Narboux (Oct. 2022). “Proof assistants for undergraduate mathematics and computer science education: elements of a priori analysis”. In: *INDRUM 2022: Fourth conference of the International Network for Didactic Research in University Mathematics*. Ed. by María Trigueros. Reinhard Hochmuth. Hanovre, Germany. URL: <https://hal.science/hal-03648357>.
-  Blanc, Jérémy et al. (2007). “Proofs for freshmen with Coqweb”. In: *Proceedings PATE07*, pp. 93–107.




Bibliography III

-  Carl, Merlin, Hinrich Lorenzen, and Michael Schmitz (Feb. 2022). “Natural Language Proof Checking in Introduction to Proof Classes – First Experiences with Diproche”. en. In: *Electronic Proceedings in Theoretical Computer Science* 354, pp. 59–70. ISSN: 2075-2180. DOI: [10.4204/EPTCS.354.5](https://doi.org/10.4204/EPTCS.354.5). URL: <http://arxiv.org/abs/2202.08131v1> (visited on 05/25/2023).
-  Delahaye, David, Mathieu Jaume, and Virgile Prevosto (Nov. 2005). “Coq, un outil pour l’enseignement. Une expérience avec les étudiants du DESS Développement de logiciels srs.”. In: *Technique et Science Informatiques* 24, pp. 1139–1160. DOI: [10.3166/tsi.24.1139-1160](https://doi.org/10.3166/tsi.24.1139-1160).
-  Gressier, Jacques (2006). *Géométrie*.
<http://revue.sesamath.net/spip.php?article42>.
<http://geometrix.free.fr/site/>.




Bibliography IV

-  Hanna, Gila and Xiaoheng Yan (Nov. 2021). “Opening a discussion on teaching proof with automated theorem provers”. In: *For the Learning of Mathematics*.
-  Kerjean, Marie et al. (Aug. 2022). “Utilisation des assistants de preuves pour l’enseignement en L1 - Retours d’expériences”. In: *Gazette SMF*.
-  Leduc, Nicolas (Dec. 2016). “QED-Tutrix : système tutoriel intelligent pour l’accompagnement des élèves en situation de résolution de problèmes de démonstration en géométrie plane”. PhD thesis. École Polytechnique de Montréal. URL: <https://publications.polymtl.ca/2450/>.

Bibliography V

-  Moore, Robert C. (1994). “Making the Transition to Formal Proof”. In: *Educational Studies in Mathematics* 27.3, pp. 249–266. ISSN: 00131954, 15730816. URL: <http://www.jstor.org/stable/3482952> (visited on 03/28/2023).
-  Nipkow, Tobias (Jan. 2012). “Teaching Semantics with a Proof Assistant: No More LSD Trip Proofs”. In: pp. 24–38. ISBN: 978-3-642-27939-3. DOI: [10.1007/978-3-642-27940-9_3](https://doi.org/10.1007/978-3-642-27940-9_3).
-  Pierce, Benjamin C. et al. (2017). *Software Foundations*. Version 5.0. Electronic textbook. URL: <http://www.cis.upenn.edu/~bcpierce/sf>.

Bibliography VI

-  Webber, C. et al. (2001). “The Baghera project: a multi-agent architecture for human learning”. In: *Proceedings of the Workshop Multi-Agent Architectures for Distributed Learning Environments*. Laboratoire Leibniz UJF Grenoble, AIED2001, San Antonio, TX, USA, pp. 12–17.
-  Weber, Keith (Oct. 2001). “Student difficulty in constructing proofs: The need for strategic knowledge”. In: *Educational Studies in Mathematics* 48.1, pp. 101–119. ISSN: 1573-0816. DOI: [10.1023/A:1015535614355](https://doi.org/10.1023/A:1015535614355). URL: <https://doi.org/10.1023/A:1015535614355>.
-  Wemmenhove, Jelle et al. (2022). *Waterproof: educational software for learning how to write mathematical proofs*. arXiv: [2211.13513](https://arxiv.org/abs/2211.13513) [math.HO].