

Lean Widgets

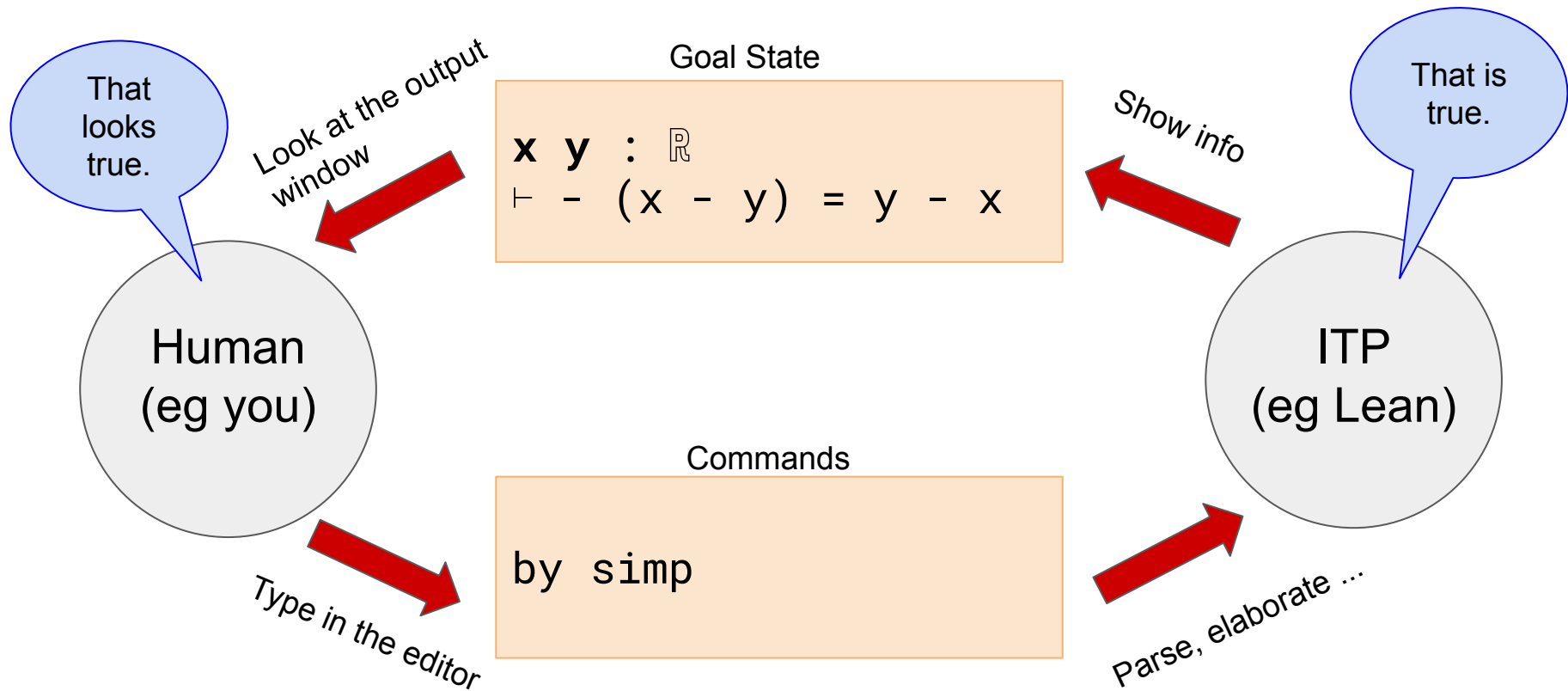
A graphical user interface framework for interactive
theorem provers

E.W.Ayers
University of Cambridge

Contents

1. Why?
2. How it works.
3. How to make your own widgets.

Proving is inherently interactive



```

emacs: Group.thy
File Edit Apps Options Functions Buffers Tools Proof-General Help
State Content Goal Retract Undo Next Use Restart Q.E.D. Find Command Stop Info Help
axclass
  group < times
  group_assoc: "(x * y) * z = x * (y * z)"
  group_left_unit: "one * x = x"
  group_left_inverse: "inv x * x = one";
text {*
  The group axioms only state the properties of left unit and inverse,
  the right versions are derivable as follows. The calculational proof
  style below closely follows typical presentations given in any basic
  course on algebra.
*};
theorem group_right_inverse: "x * inv x = (one::'a::group)";
proof -;
  have "x * inv x = one * (x * inv x)";
  by (simp only: group_left_unit);
  also have "... = (one * x) * inv x";
  by (simp only: group_assoc);
  also have "... = inv (inv x) * inv x * x * inv x";
  by (simp only: group_left_inverse);
  also have "... = inv (inv x) * (inv x * x) * inv x";
  by (simp only: group_assoc);
  also have "... = inv (inv x) * one * inv x";
  --XEmacs: Group.thy 'group_right_inverse' (Isabelle/Isar script Font Scripting)
Proof(state): step 6, depth 0
this:
  one * (x * inv x) = one * x * inv x
goal (theorem group_right_inverse):
x * inv x = one
1. x * inv x = one
--XEmacs: *isabelle-goals* (Isabelle/Isar proofstate Font) All

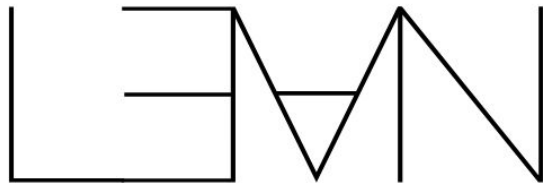
```

Aspinall, David. "Proof General: A generic tool for proof development." *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, Berlin, Heidelberg, 2000

Some solutions to making graphical interactive theorem provers.



- Server protocol
- Go to definition
 - 'Lightbulbs'
 - Completions
 - Type information
 - **Infoview**
 - Type info
 - Try this
 -



But I want to these in the infoview!

- Filtering goals
- "no goals" → "goals accomplished"
- Type information
- Try this,

The Lean Infoview is a web browser!

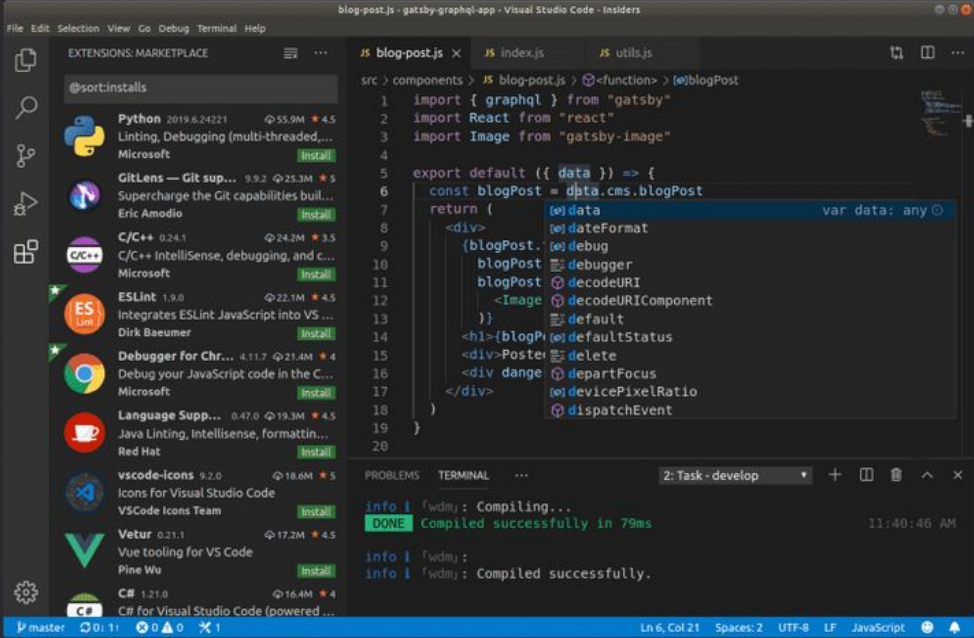
Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

↓ .deb Debian, Ubuntu... ↓ .rpm Red Hat, Fedora... ▾

[Other platforms and Insiders Edition](#)

By using VS Code, you agree to its [license and privacy statement](#).



The screenshot shows the Visual Studio Code interface. On the left, the 'EXTENSIONS: MARKETPLACE' sidebar is open, displaying a list of extensions under the '@sort:installs' filter. Visible extensions include Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, vscode-icons, Vetur, and C#. The main editor area shows a JavaScript file named 'index.js' with the following code:

```
1 import { graphql } from "gatsby"
2 import React from "react"
3 import Image from "gatsby-image"
4
5 export default ({ data }) => {
6   const blogPost = data.cms.blogPost
7   return (
8     <div>
9       {blogPost}
10      <div>
11        {blogPost}
12        {blogPost}
13      </div>
14      <h1>{blogPost}
15      </div>
16      <div>
17        {blogPost}
18      </div>
19    )
20  }
```

The bottom status bar shows the current file is 'index.js' at line 6, column 21, with 2 spaces and UTF-8 encoding. The terminal at the bottom right shows the following output:

```
info | f'wditj : Compiling...
DONE | Compiled successfully in 79ms
info | f'wditj :
info | f'wditj : Compiled successfully.
```

And Lean is full-fledged programming language!

A new solution: turn Lean in to a full-blown UI framework.

```
meta def goals_accomplished_message {α} : html α :=  
h "div" [cn "f5"] ["goals accomplished 🎉"]
```



Lean widgets

▼ Tactic state widget ▼

goals accomplished 🎉 filter: no filter ▼

Main use: tactic states in the new Infoview

sieves.lean

```
lemma mem_inf {Ss : set (sieve X)} {Y} (f : Y → X) :  
  Inf Ss f ↔ ∀ (S : sieve X) (H : S ∈ Ss), S f :=  
  iff.rfl  
  
@[simp]  
lemma mem_Sup {Ss : set (sieve X)} {Y} (f : Y → X) :  
  Sup Ss f ↔ ∃ (S : sieve X) (H : S ∈ Ss), S f :=  
  iff.rfl  
  
@[simp]  
lemma mem_inter {R S : sieve X} {Y} (f : Y → X) :  
  (R ∩ S) f ↔ R f ∧ S f :=  
  iff.rfl  
  
@[simp]  
lemma mem_union {R S : sieve X} {Y} (f : Y → X) :  
  (R ∪ S) f ↔ R f ∨ S f :=  
  iff.rfl  
  
@[simp]  
lemma mem_top (f : Y → X) : (τ : sieve X) f := trivial  
  
/-- Generate the smallest sieve containing the given set of arrows. -/
```

Lean Infoview

▼ sieves.lean:172:0

▼ Tactic state

expected type:

C : Type u

_inst_1 : category C

X : C

R : sieve X

S : sieve X

Y : C

f : Y → X

↑ (R ∩ S) f ↔ ↑ R f ∧ ↑ S f

Prop

coe_fn sieve X sieve.has_coe_to_fun R n S Y f

sieve X

has_inf.inf sieve X

semilattice_inf.to_has_inf (sieve X) R S

Add a 'go to definition' button in widgets.

```
/--  
Render a 'go to definition' button for a given expression.  
If there is no definition available, then returns an empty list.  
-/  
meta def goto_def_button {γ} : expr → tactic (list (html (action γ)))  
| e := (do  
  (expr.const n _) ← pure $ expr.get_app_fn e,  
  env ← tactic.get_env,  
  let file := environment.decl_olean env n,  
  pos ← environment.decl_pos env n,  
  pure $ [h "button" [  
    cn "pointer ba br3 mr1",  
    on_click (λ _, action.effect $ widget.effect.reveal_position file pos),  
    attr.val "title" "go to definition"] ["↵"]  
  ) <|> pure []
```

Community widgets

Sudoku Solver by Markus Himmel; *Rubik's Cube* by Kendall Frey

```
cc13.lean x
src > cc13.lean
40 have c40 : s.f (0, 2) = 4 := by naked_single,
41 have c44 : s.f (4, 4) = 4 := by naked_single,
42 have c17 : s.f (1, 7) = 3 := by box_logic,
43 have c00 : s.f (0, 0) = 3 := by box_logic,
44 have p12 : s.snyder 0 2 2 2 8 := by box_logic,
45 have c80 : s.f (8, 0) = 8 := by pencil_with p4 p12,
46 clear p4,
47 have p13 : s.snyder 1 8 2 8 4 := by box_logic,
48 have p14 : s.snyder 6 6 7 6 4 := by box_logic with p13,
49 have c47 : s.f (4, 7) = 6 := by box_logic,
50 have c78 : s.f (7, 8) = 6 := by box_logic,
51 have c52 : s.f (5, 2) = 6 := by pencil_with p0,
52 clear p0,
53 have c50 : s.f (5, 0) = 1 := by naked_single,
54 have c12 : s.f (1, 2) = 1 := by pencil_with p3,
55 clear p3,
56 have p27 : s.double 3 0 4 9 := by naked_single,
57 have p28 : s.triple 3 1 2 4 9 := by naked_single,
58 have p29 : s.double 3 7 2 9 := by naked_single,
59 have c38 : s.f (3, 8) = 8 := by naked_single with p27 p28 p29,
60 have c06 : s.f (0, 6) = 8 := by box_logic,
61 have c43 : s.f (4, 3) = 8 := by box_logic,
62 have c22 : s.f (2, 2) = 8 := by pencil_with p12,
63 clear p12,
64 have c10 : s.f (1, 0) = 4 := by box_logic,
65 have c62 : s.f (0, 2) = 9 := by box_logic,
66 have c30 : s.f (3, 0) = 9 := by pencil_with p27,
67 have c08 : s.f (0, 8) = 2 := by row_logic,
68 have c28 : s.f (2, 8) = 4 := by pencil_with p13,
69 clear p13,
70 have c18 : s.f (1, 8) = 9 := by row_logic,
71 have c42 : s.f (4, 2) = 2 := by naked_single,
72 have c31 : s.f (3, 1) = 4 := by pencil_with p28,
73 have c82 : s.f (8, 2) = 4 := by col_logic,
74 have c46 : s.f (4, 6) = 9 := by row_logic,
75 have c45 : s.f (4, 5) = 1 := by row_logic,
76 have c36 : s.f (3, 6) = 1 := by box_logic,
77 have c37 : s.f (3, 7) = 2 := by pencil_with p29,
78 have c56 : s.f (5, 6) = 5 := by box_logic,
79 have c33 : s.f (3, 3) = 5 := by row_logic,
80 have c53 : s.f (5, 3) = 2 := by row_logic,
```

Lean Infoview x

cc13.lean:59:64

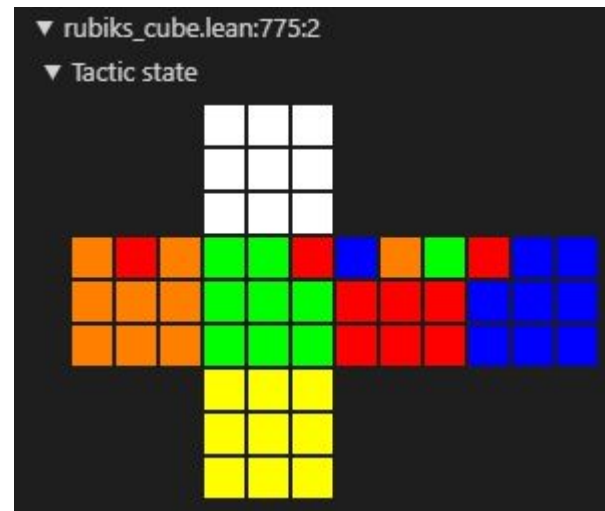
Tactic state

3	7	8	6	5	4	1	
	5	1	7	2	8	6	3 ⁴
2	6	8	9	1	3	7	5 ⁴
49	249	7	5	3	6	29	8
5	3		4			6	7
1	8	6	5	9	7	4	3
6		3	7	5	4	8	1
7		5	3	8	4		6
8			6	9	3	7	5

1 goal

filter: no filter

```
s : sudoku
c01 : s.f (0, 1) = 7
c04 : s.f (0, 4) = 5
c07 : s.f (0, 7) = 1
-14 : s.f (1, 4) = 2
```



Community widgets

Mathematica Bridge by Robert Y. Lewis and Minchao Wu

```
open real
noncomputable def f : ℝ → ℝ := λ x, sin x + cos x

begin_mm_block (unfolding f)

as image
"Plot3D[x^2-y, {x,-3,3}, {y,-3,3}]";

as image
"Plot["(λ y, (sin y)^2 - y)"[x], {x,-10,10}]";

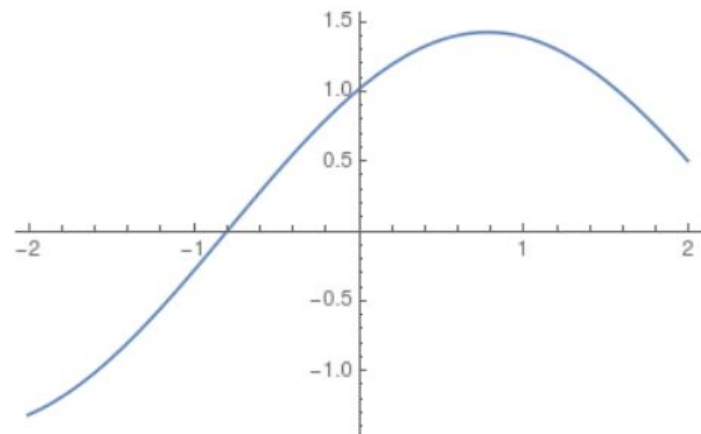
as image
"Plot["f"[y], {y,-2,2}]";

end_mm_block
```

▼ Tactic state

widget ▼

Mathematica output



► All Messages (2)



Demo!

Looking to the future

- Concurrency
- Better support for 'dropping in' widgets.
- How to support 3rd party Javascript libraries (D3, MathJax, ...)?
- Fancy Lean 4 parsing for inline HTML



Thanks!

Gabriel Ebner, Bryan Gen-ge Chen for helping with PRs and collaborating with the VScode extension.

Also thanks to Daniel Fabien for discovering the CSS to get linebreaking to look nice.

The mathlib community in general for just being really constructive and solution oriented.

Angela Li for letting me use her tower of Hanoi widget in the demo.

Overview

