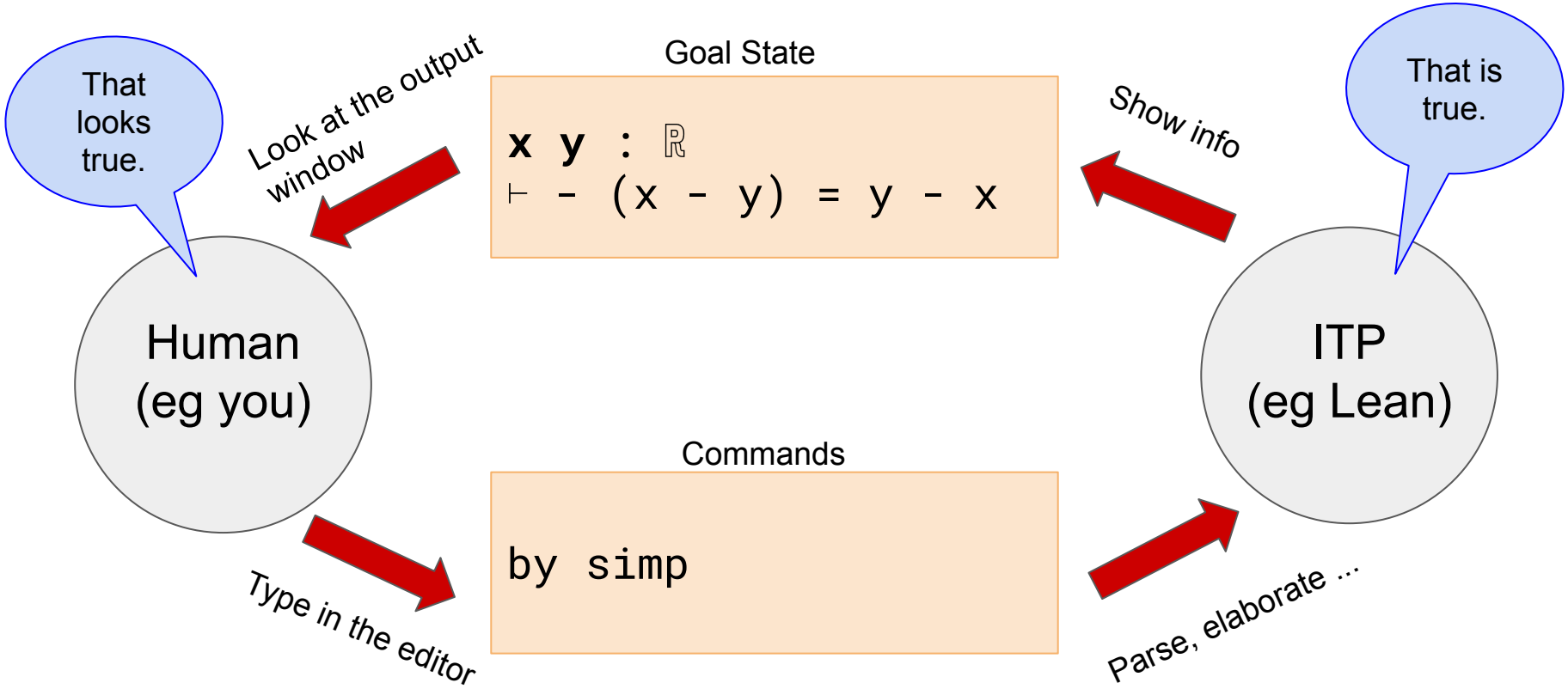
Lean Widgets

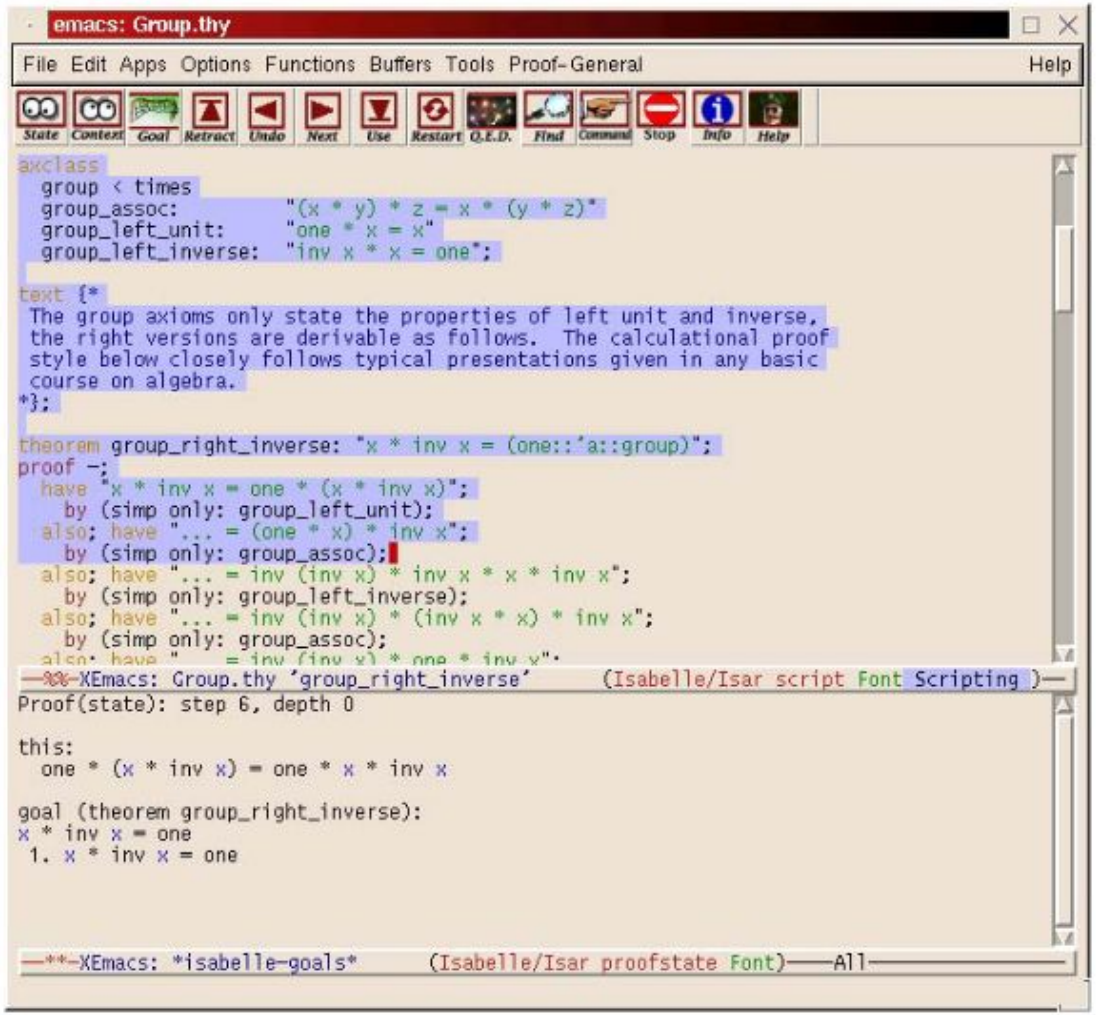
A graphical user interface framework for interactive theorem provers

E.W.Ayers
University of Cambridge

Contents

Proving is inherently interactive





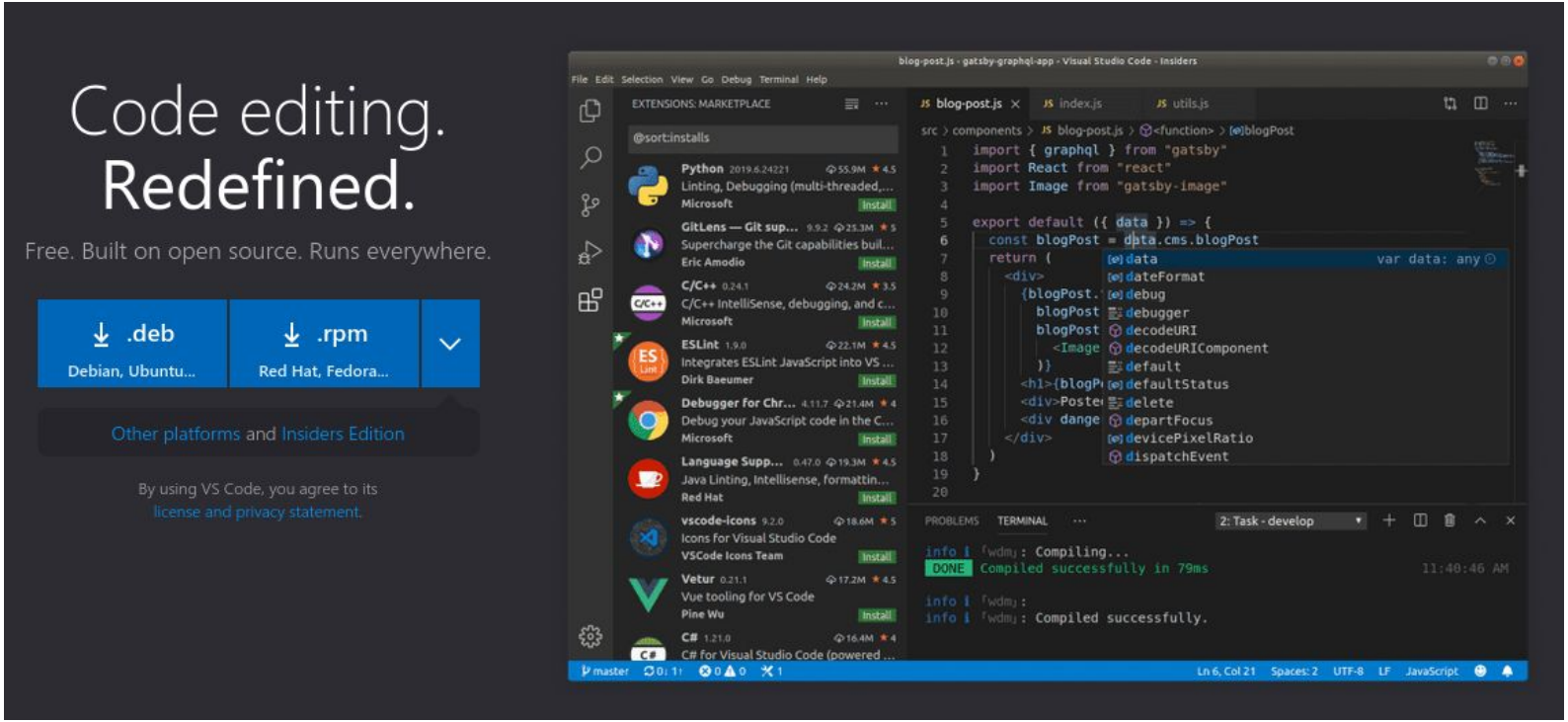
Aspinall, David. "Proof General: A generic tool for proof development." *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, Berlin, Heidelberg, 2000

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, ……...

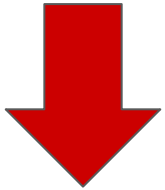
The Lean Infoview is a web browser!



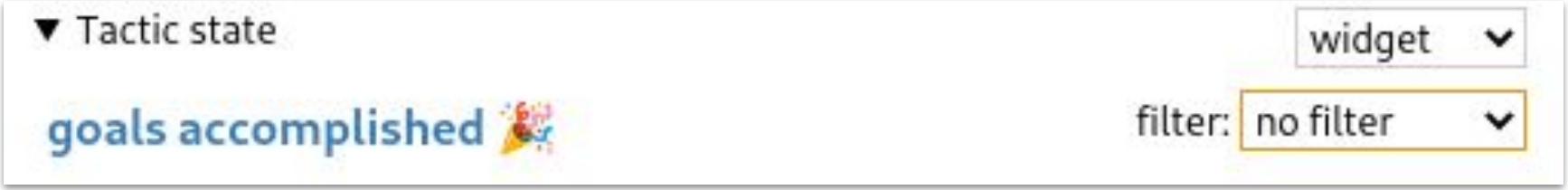
And Lean is full-fledged programming language!

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



Main use: tactic states in the new Infoview

File Edit Selection View Go Run Terminal Help — Lean Infoview - mathlib - Visual Studio Code

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) : (⊤ : sieve X) f := trivial

/-- Generate the smallest sieve containing the given set of arrows. -/
```

Lean Infoview

▼ sieves.lean:172:0

▼ Tactic state

widget

```
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 ⊓ S | Y | f

sieve X

has_inf.inf | sieve X

semilattice_inf.to_has_inf (sieve X) | R | S

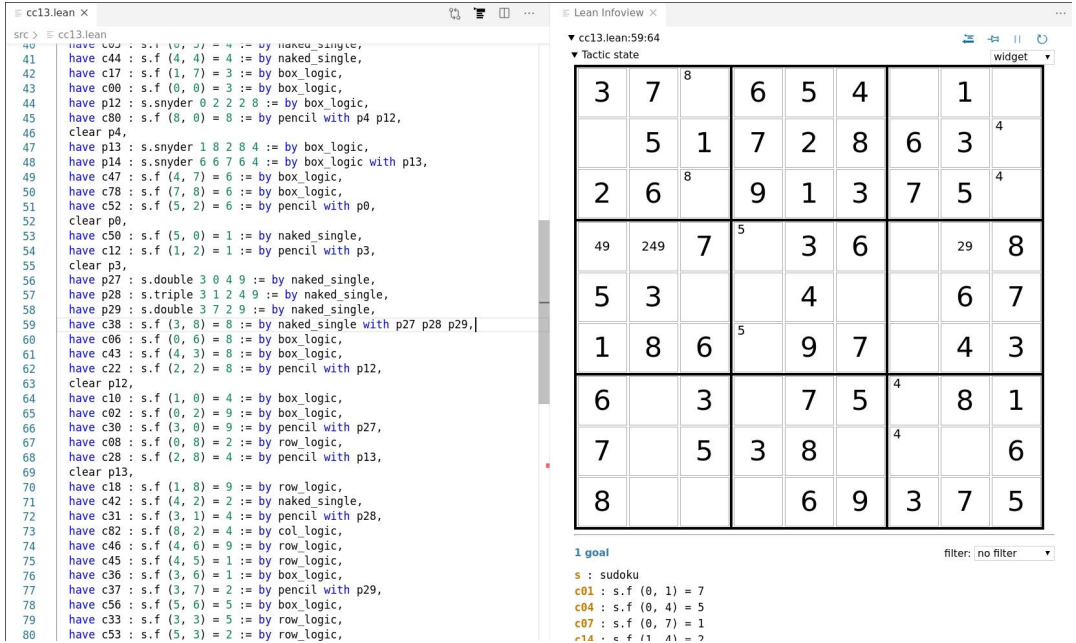
Add a ‘go to definition’ button in widgets.

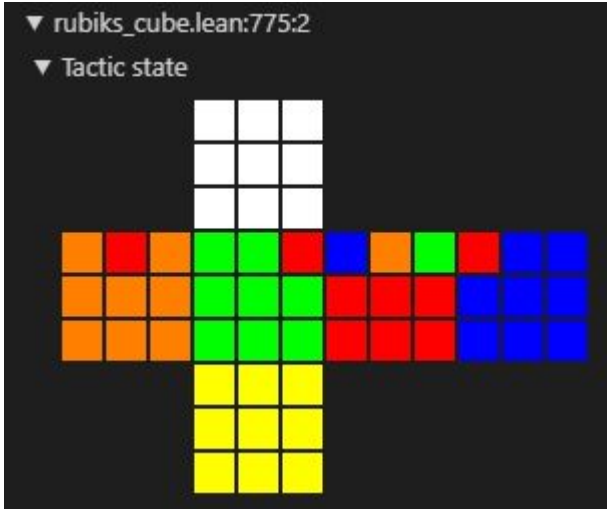
```
/--
Render a 'go to definition' button for a given expression.
If there is no definition available, then returns an empty list.
-/
meta def goto_def_button {γ} : expr → tactic (list (html (action γ)))
| e := (do
    (expr.const n _) ← pure $ expr.get_app_fn e,
    env ← tactic.get_env,
    let file := 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 []
```

https://github.com/leanprover-community/mathlib/pull/3982

Community widgets

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





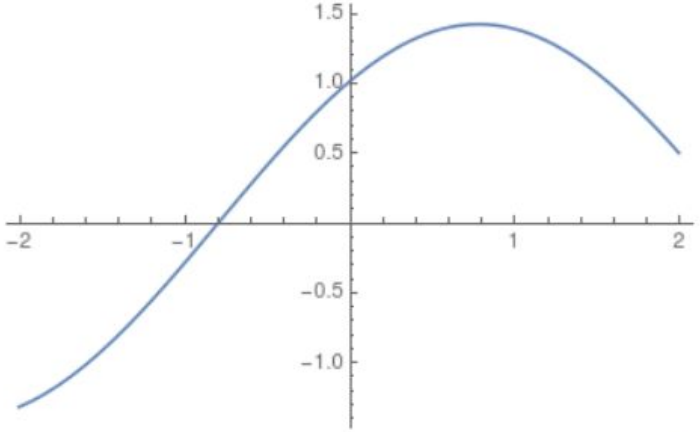
Community widgets

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!

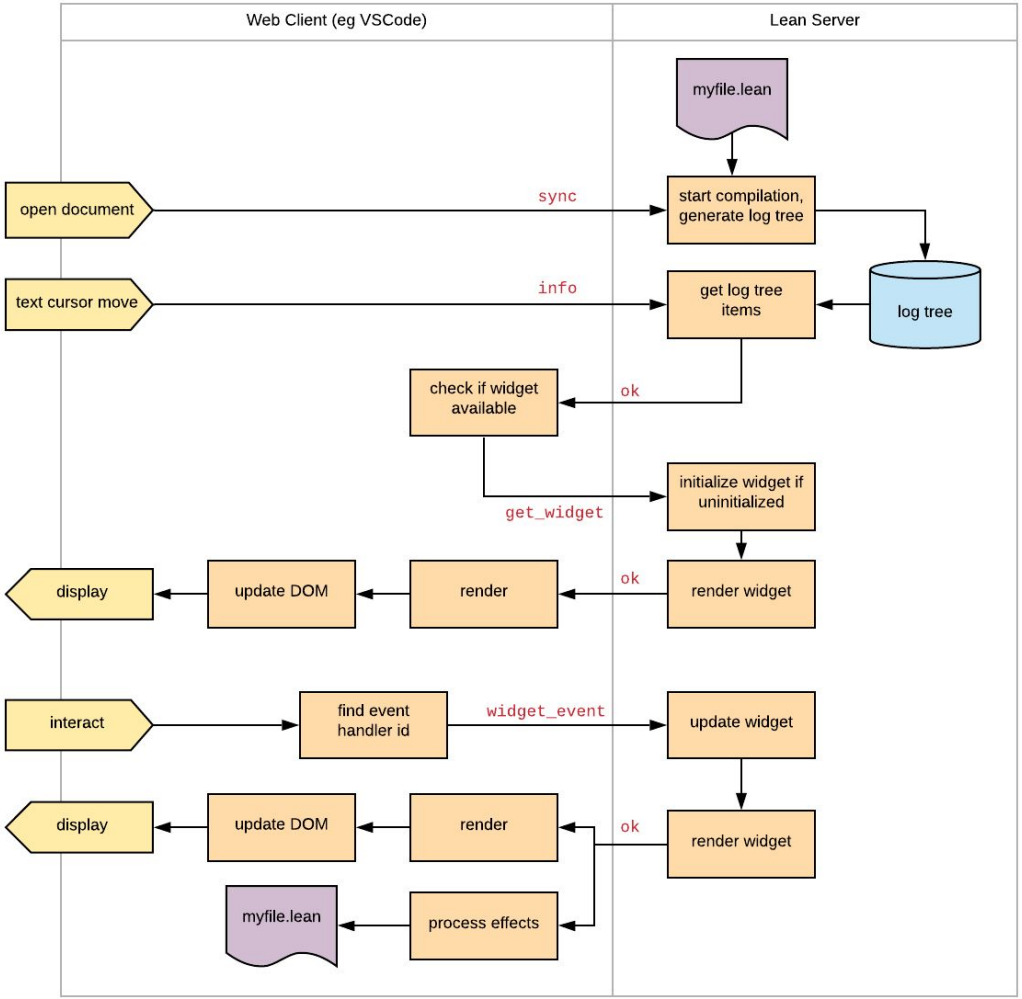
Gabriel Ebner, Bryan Gen-ge Chen for helping with PRs and collaborating with the VScode extension.

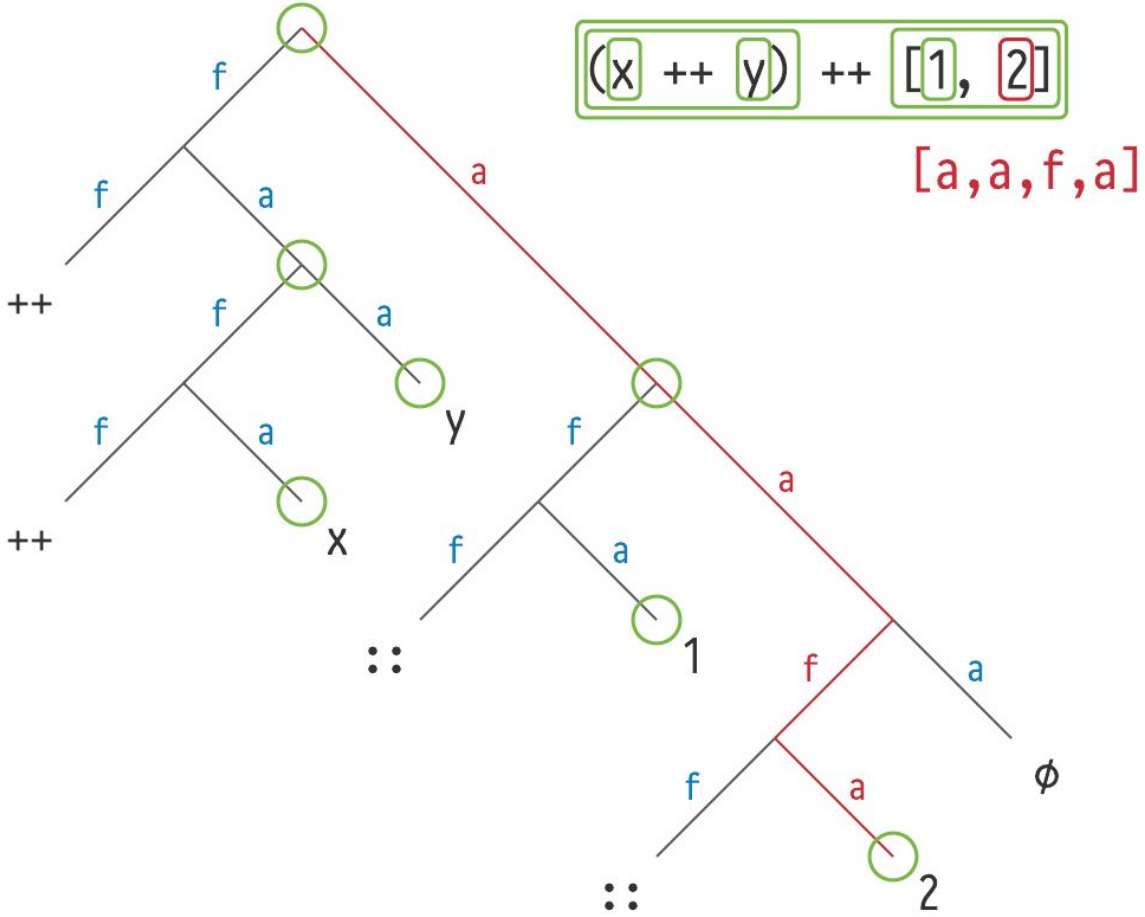
Also thanks to Daniel Fabien for discovering the CSS to get linebreaking to look nice.

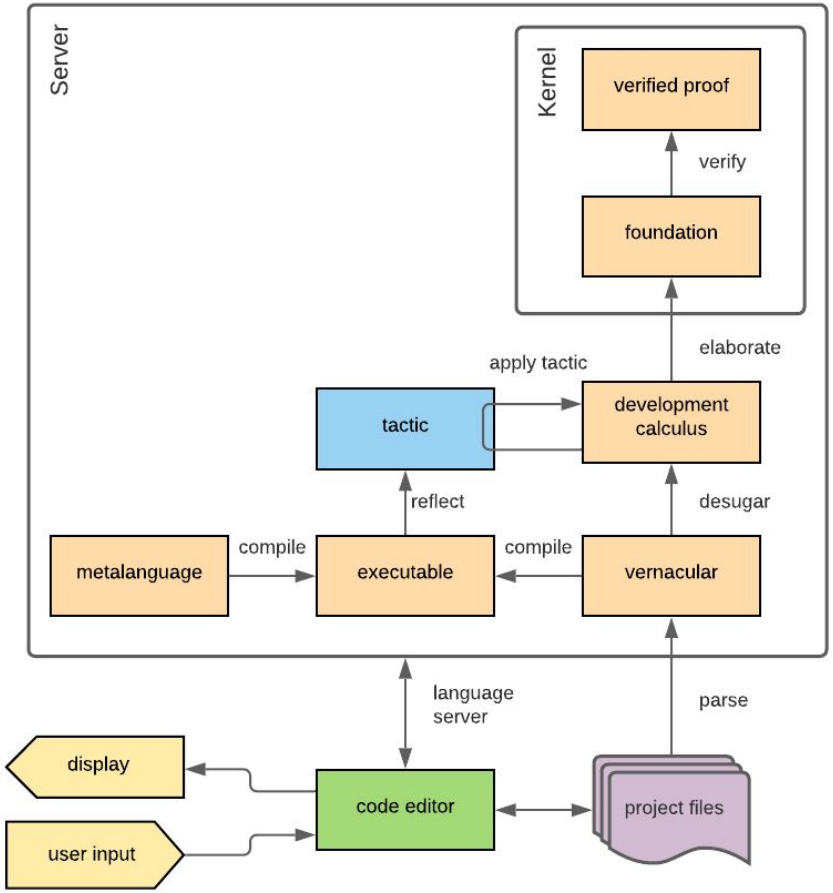
The mathlib community in general for just being really constructive and solution oriented.

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

Overview


Web Client (eg VSCode)
Lean Server
myfile.lean
open document
sync
start compilation, generate log tree
text cursor move
info
get log tree items
log tree
check if widget available
ok
initialize widget if uninitialized
get_widget
display
update DOM
render
ok
render widget
interact
find event handler id
widget_event
update widget
display
update DOM
render
ok
render widget
myfile.lean
process effects


(x ++ y) ++ [1, 2]
[a,a,f,a]
f
a
++
++
x
y
::
1
::
2
ϕ


Server
Kernel
verified proof
verify
foundation
elaborate
apply tactic
tactic
development calculus
reflect
desugar
metalanguage
compile
executable
compile
vernacular
language server
parse
display
code editor
project files
user input