VSCode-PVS Walkthrough

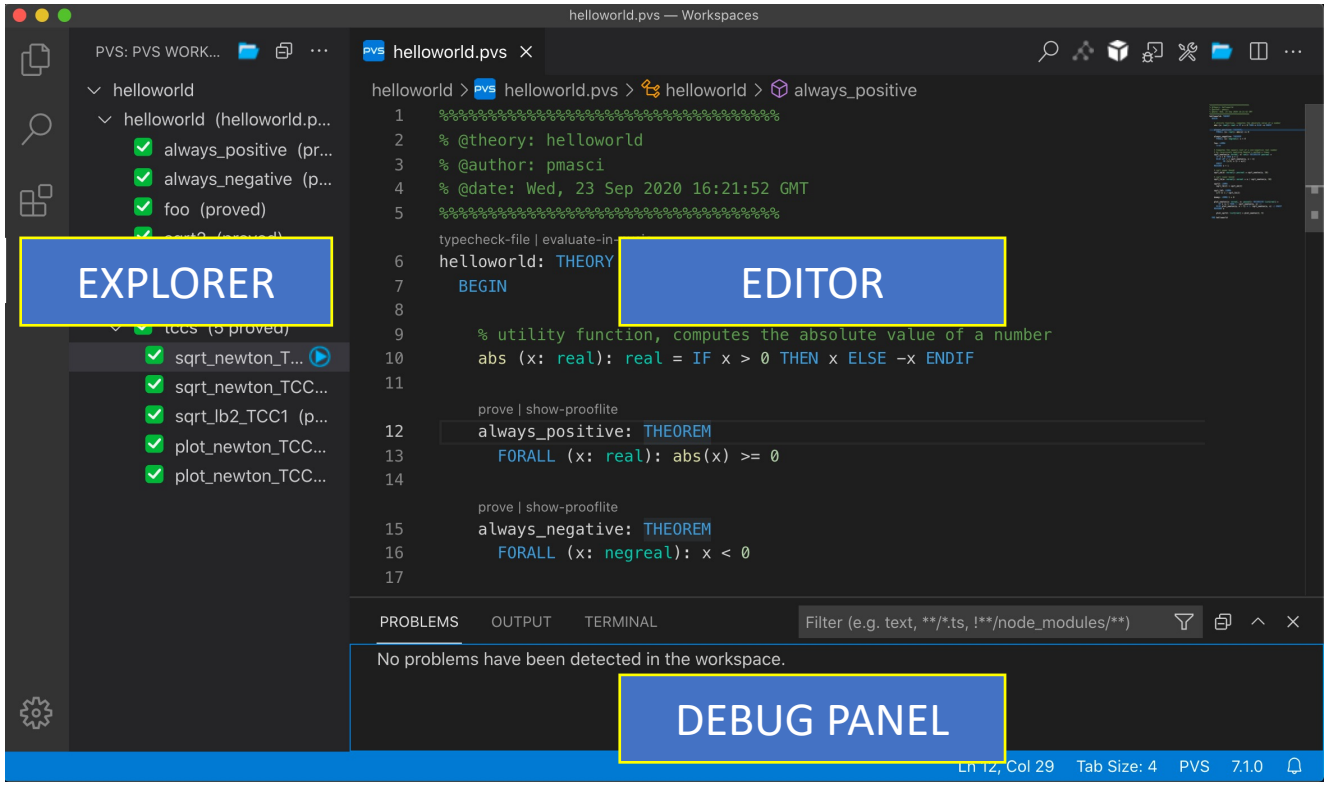
Paolo Masci
paolo.masci@nianet.org
National Institute of Aerospace (NIA)

National Institute of Aerospace

Part 1
Primer on Visual Studio Code

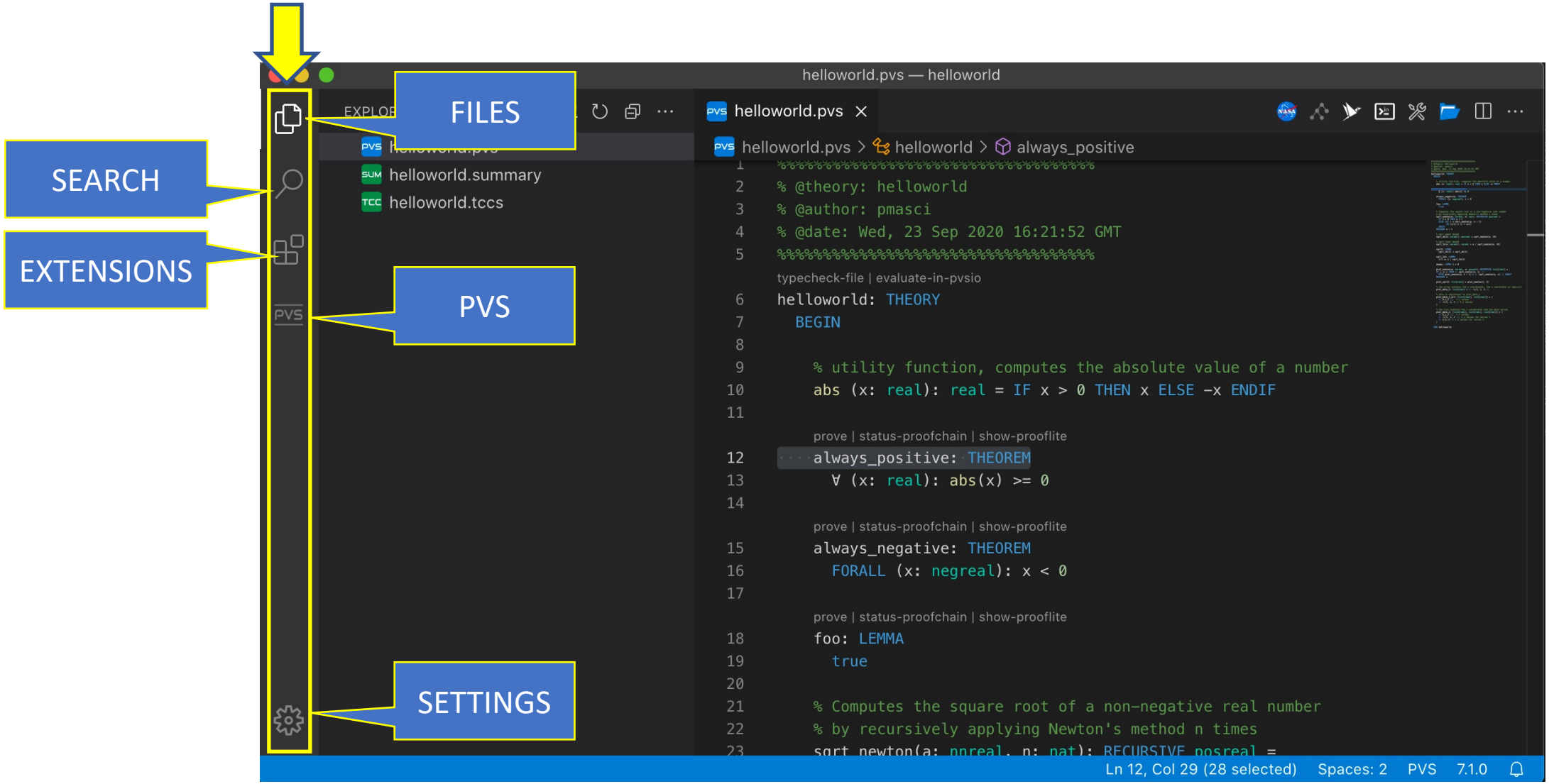
Visual Studio Code Interface

- **Editor**: shows text/code editors and integrated consoles
- **Explorer**: shows information in the form of a tree view, akin to file browsers
- **Debug panel**: shows information about warnings/errors



Explorer

An **activity bar** allows to switch between different views



Editor

Functionalities can be accessed through point-and-click interactions

DRAG & DROP EDITOR PANELS TO ORGANIZE IN DIFFERENT LAYOUTS (E.G., SPLIT-VIEWS)

TOOLBAR

MINIMAP

helloworld.pvs — helloworld

helloworld.pvs

helloworld.pvs
helloworld.summary
helloworld.tccs

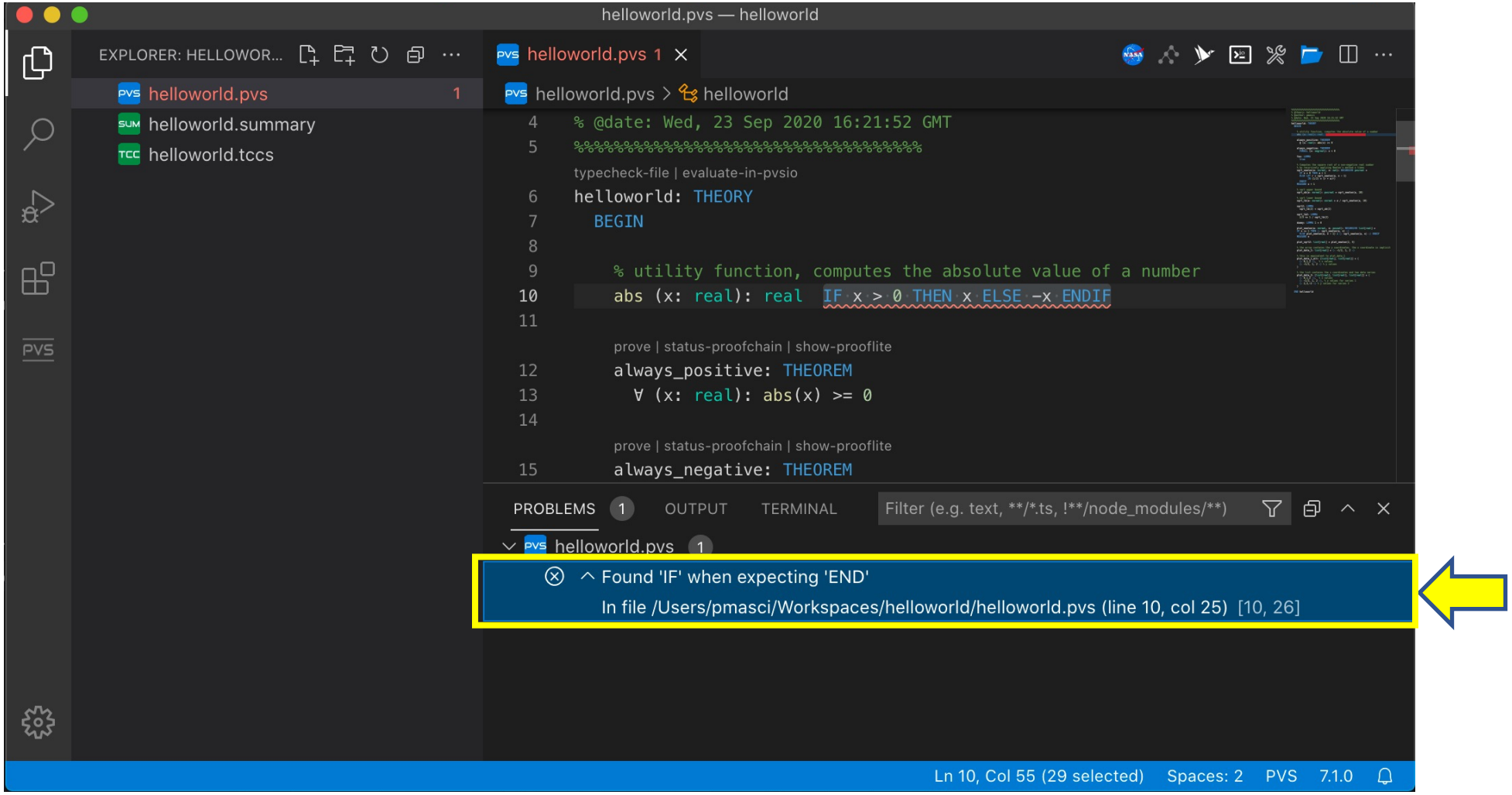
helloworld.pvs > helloworld > always_positive

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @theory: helloworld
% @author: pmasci
% @date: Wed, 23 Sep 2020 16:21:52 GMT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
typecheck-file | evaluate-in-pvsio
helloworld: THEORY
  BEGIN

    % utility function, computes the absolute value of a number
    abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF

    prove | status-proofchain | show-prooflite
    always_positive: THEOREM
      ∀ (x: real): abs(x) >= 0

    prove | status-proofchain | show-prooflite
    always_negative: THEOREM
      FORALL (x: negreal): x < 0

    prove | status-proofchain | show-prooflite
    foo: LEMMA
      true

    % Computes the square root of a non-negative real number
    % by recursively applying Newton's method n times
    sqrt_newton(a: nnreal, n: nat): RECURSIVE posreal =
```

Ln 12, Col 29 (28 selected) Spaces: 2 PVS 7.1.0

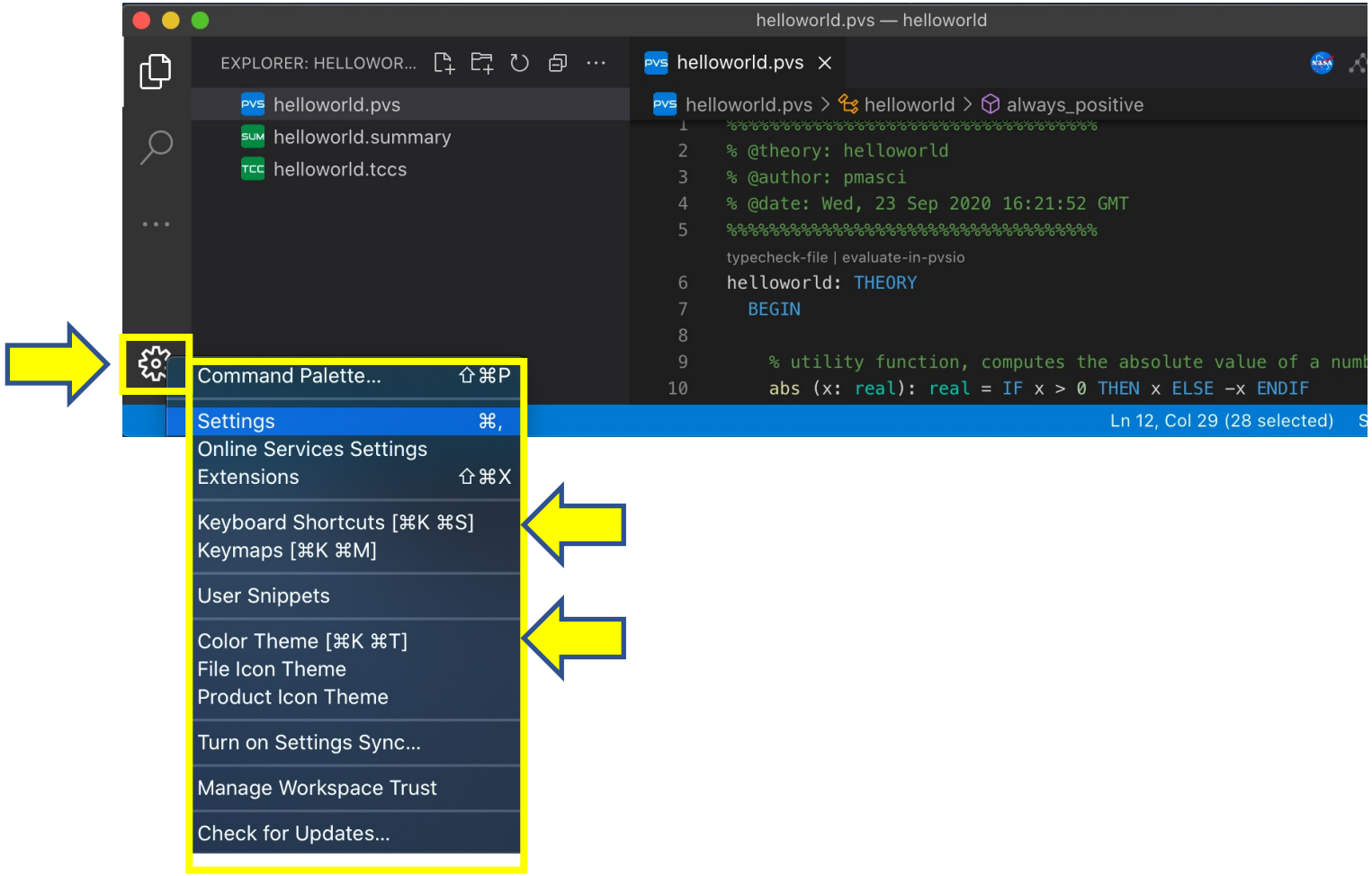
Debug Panel

Click on warnings/errors to jump to the location of the error

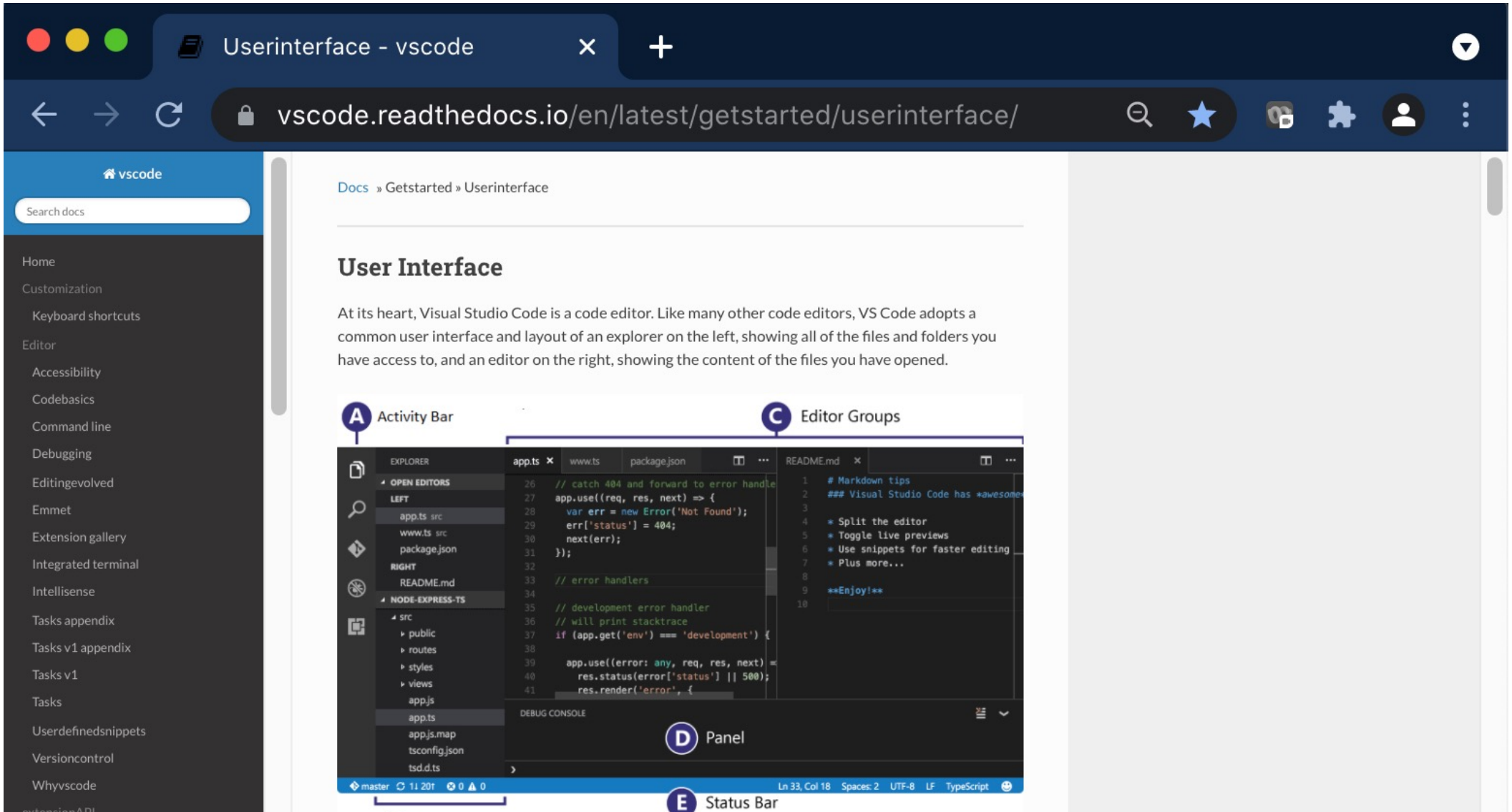


Keyboard Shortcuts & Color Theme

Keyboard shortcuts and Color Theme (e.g., Dark / Light) can be accessed using the wheel icon located in the lower-left corner of the interface



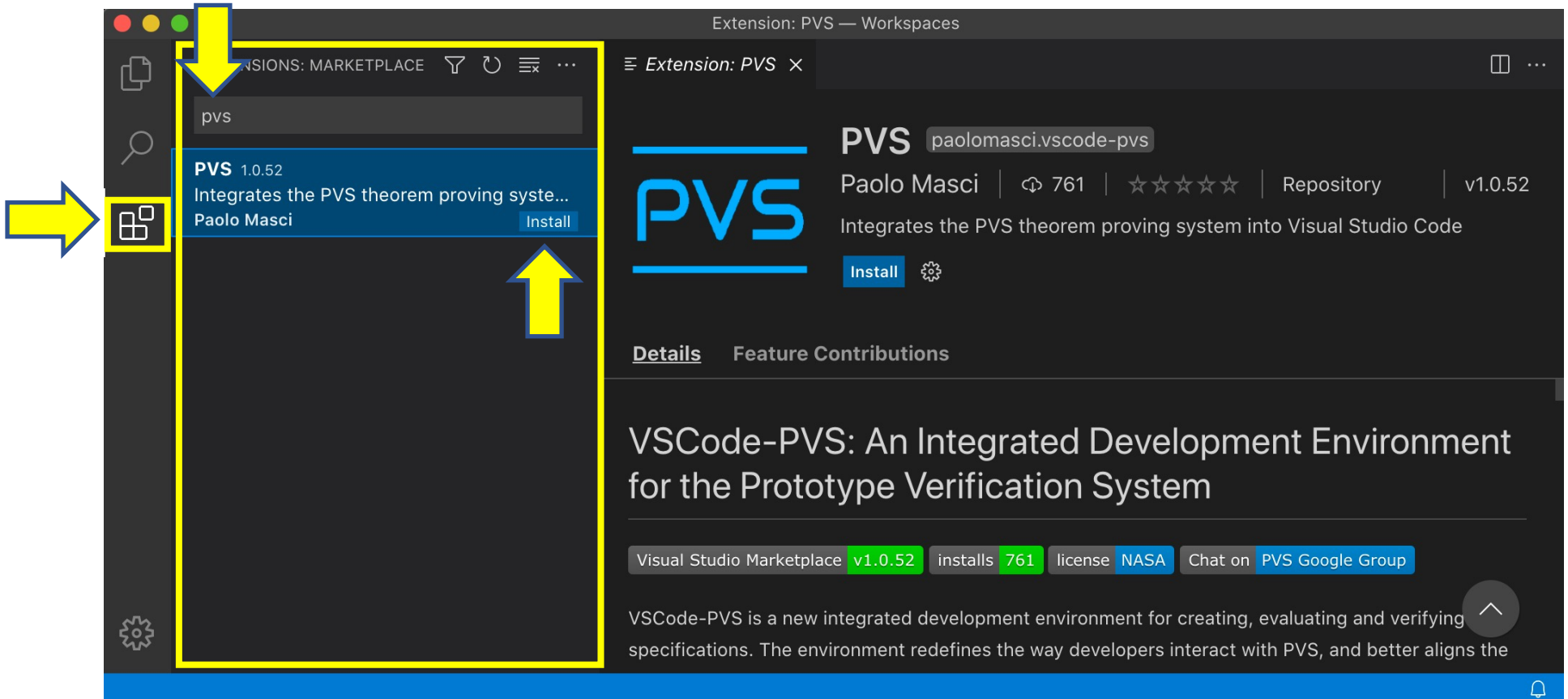
Full Visual Studio Code documentation available online at https://vscode.readthedocs.io/en/latest/getstarted/userinterface/



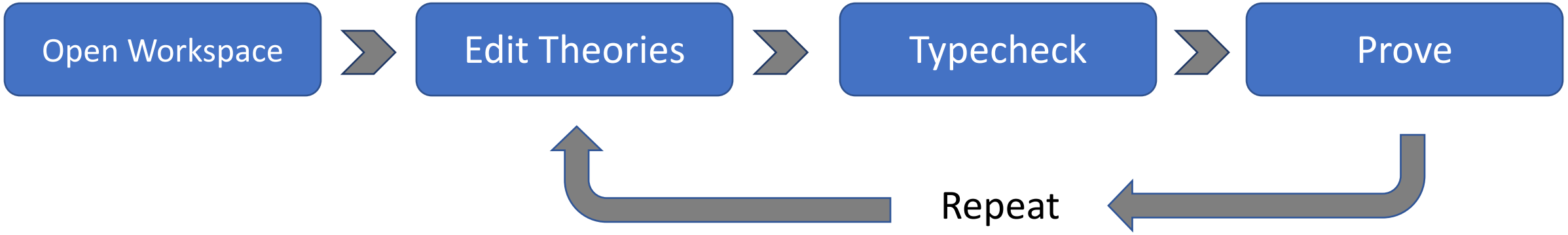
Part 2
VSCode-PVS

Installing VSCode-PVS

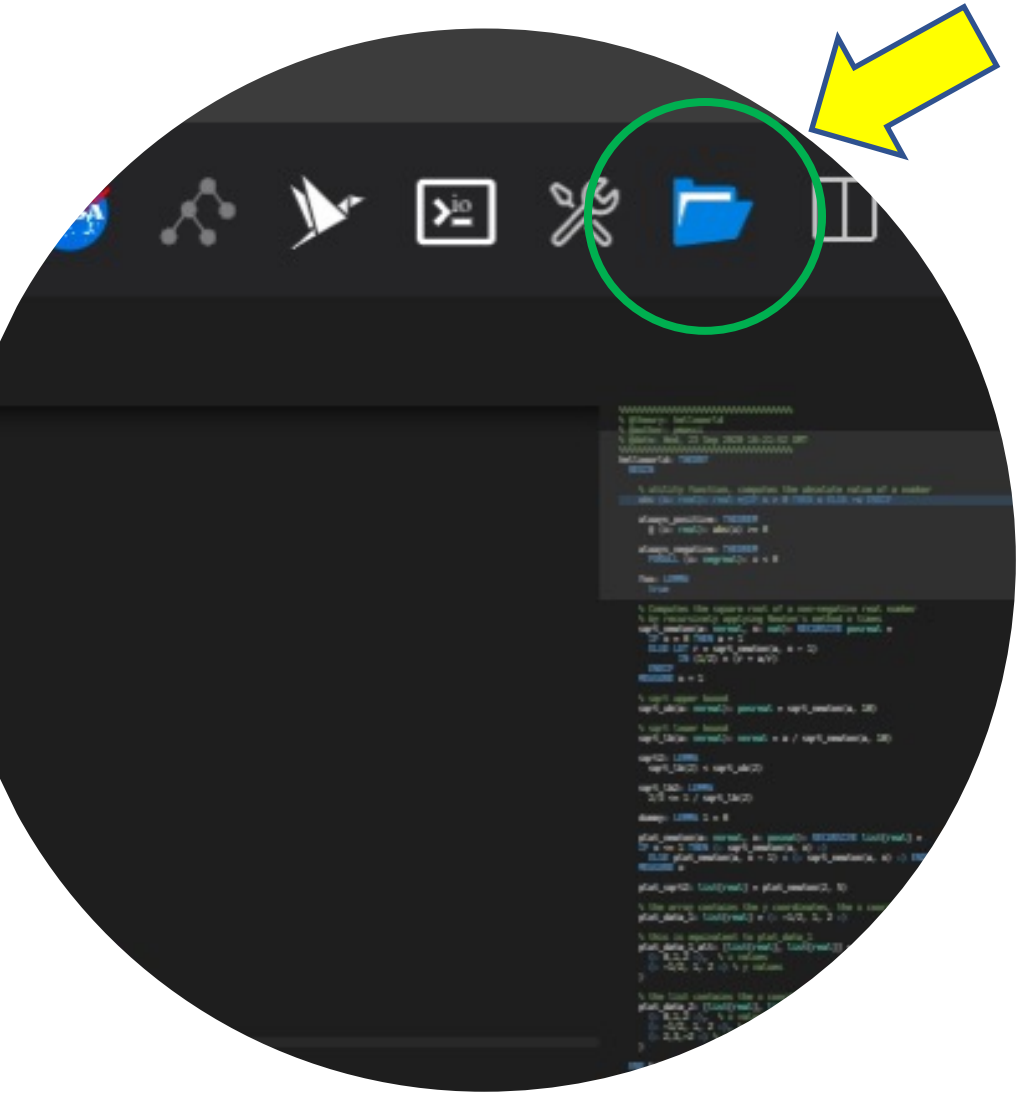
- Select the **Extension Panel**, search **PVS**, and click **install**
- A pop-up window will open, select **Download PVS** to complete the installation process
- Use your home folder as base folder for the installation.



Workflow in VSCode-PVS



Opening a PVS Workspace



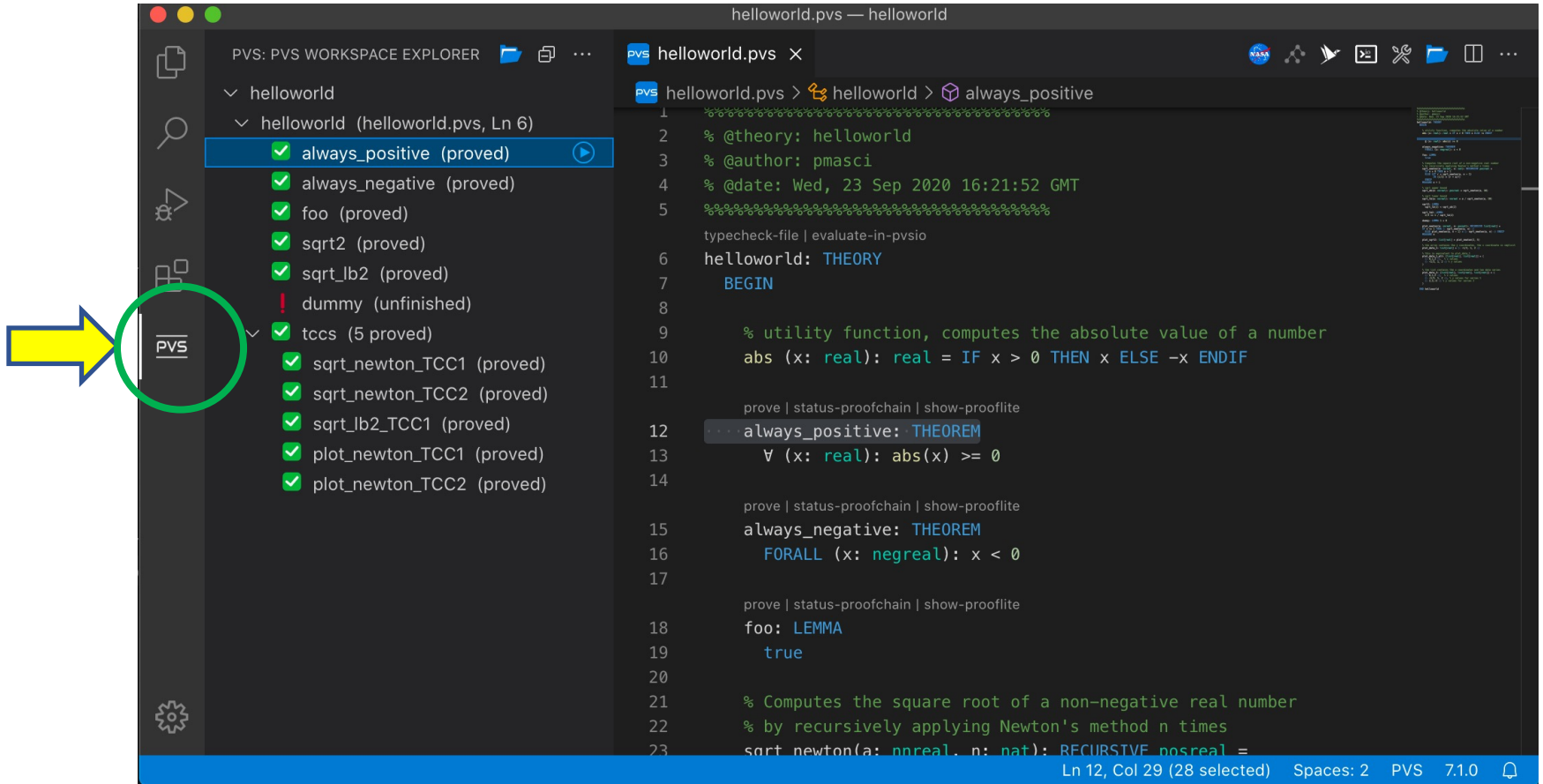
A PVS workspace is a folder that contains (or will contain) PVS files.

To open a PVS workspace:

- Click the **open folder icon** located in the toolbar
- Browse the file system and select an existing folder (or create a new folder) that will be used as PVS workspace.

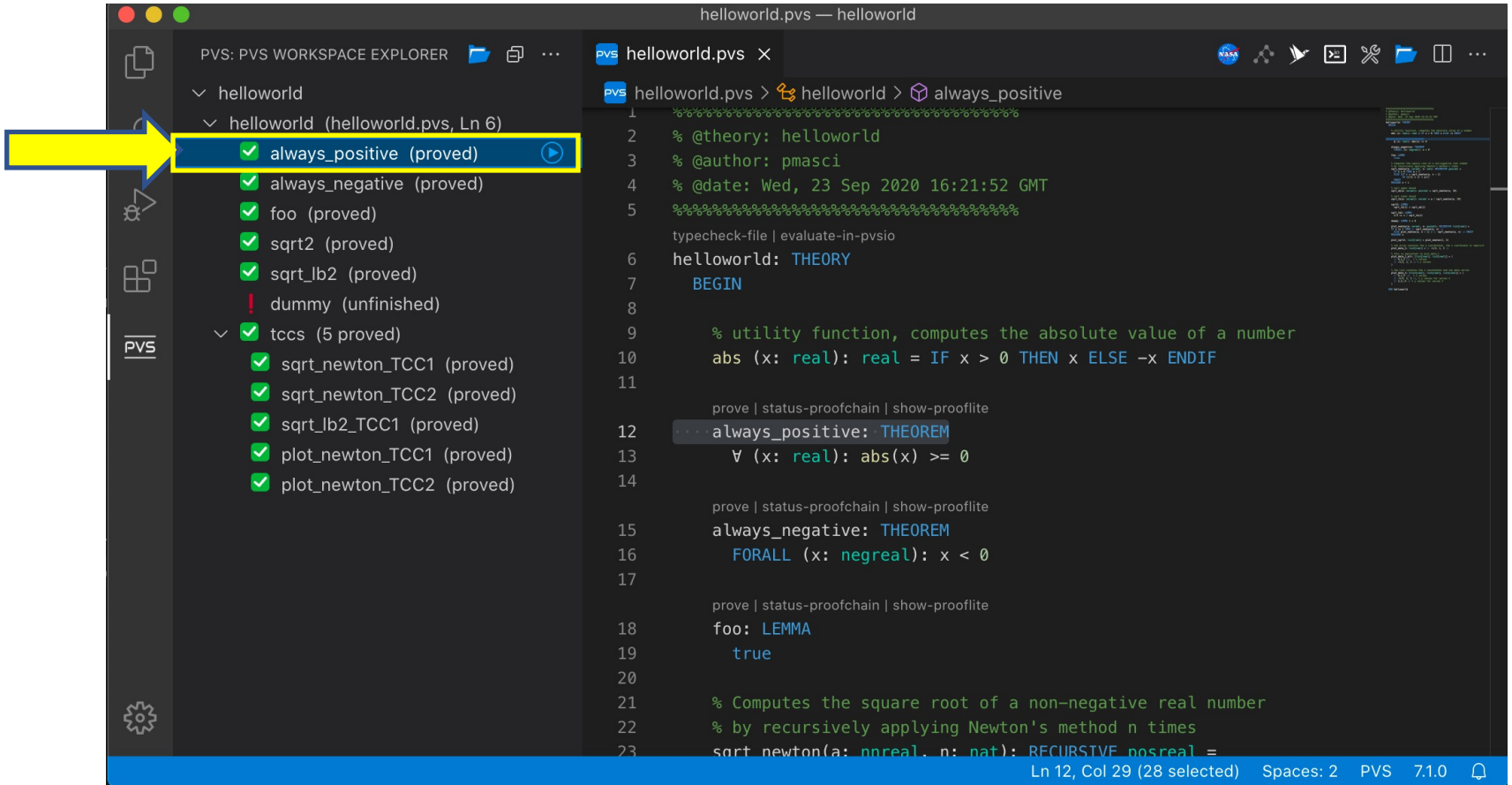
PVS Workspace Explorer

Once a workspace is open, click the PVS icon to switch to the PVS Workspace Explorer and view the list of theories and theorems



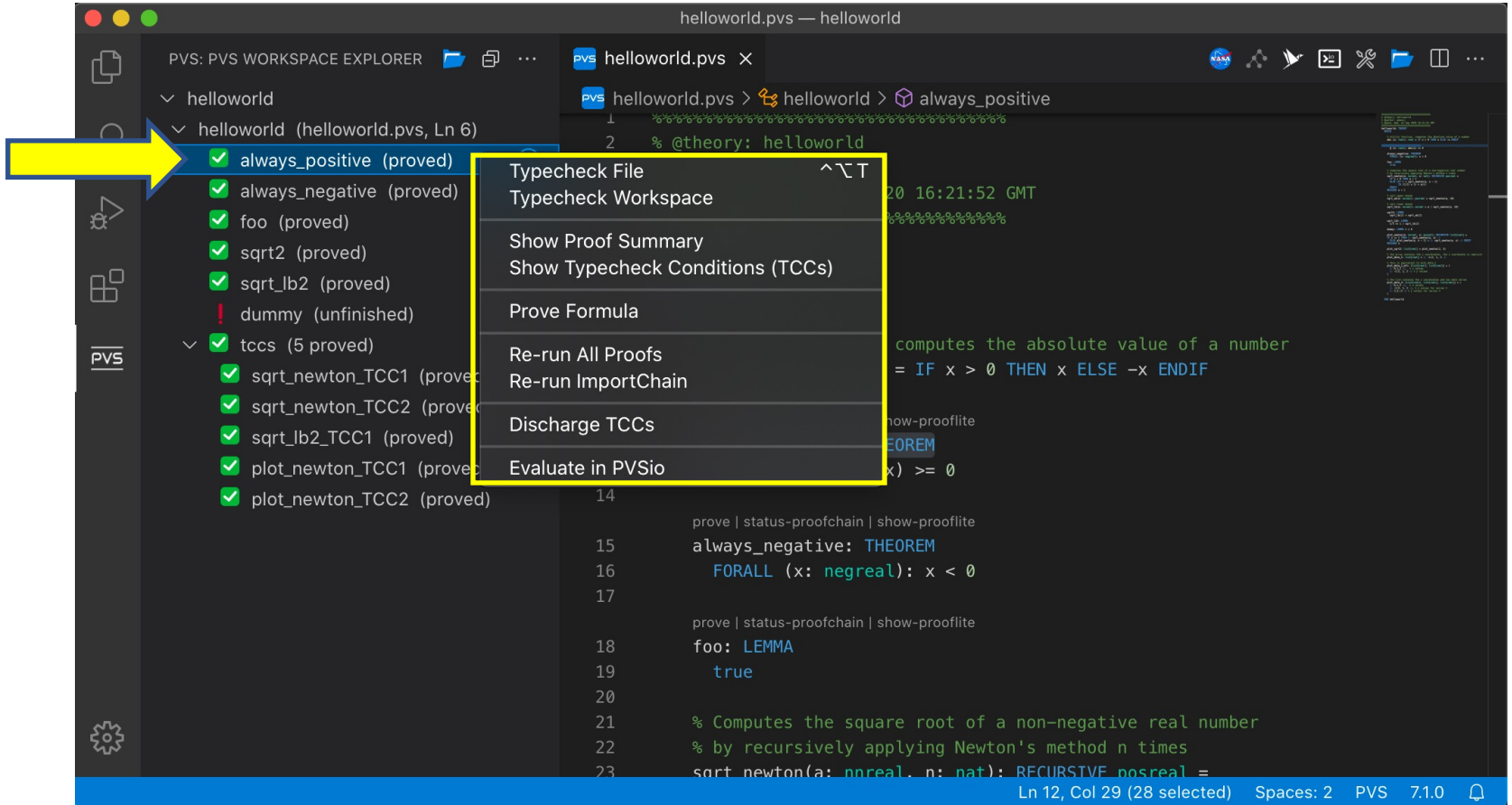
PVS Workspace Explorer (cont'd)

Click on a theory/theorem to open the corresponding PVS file in the editor



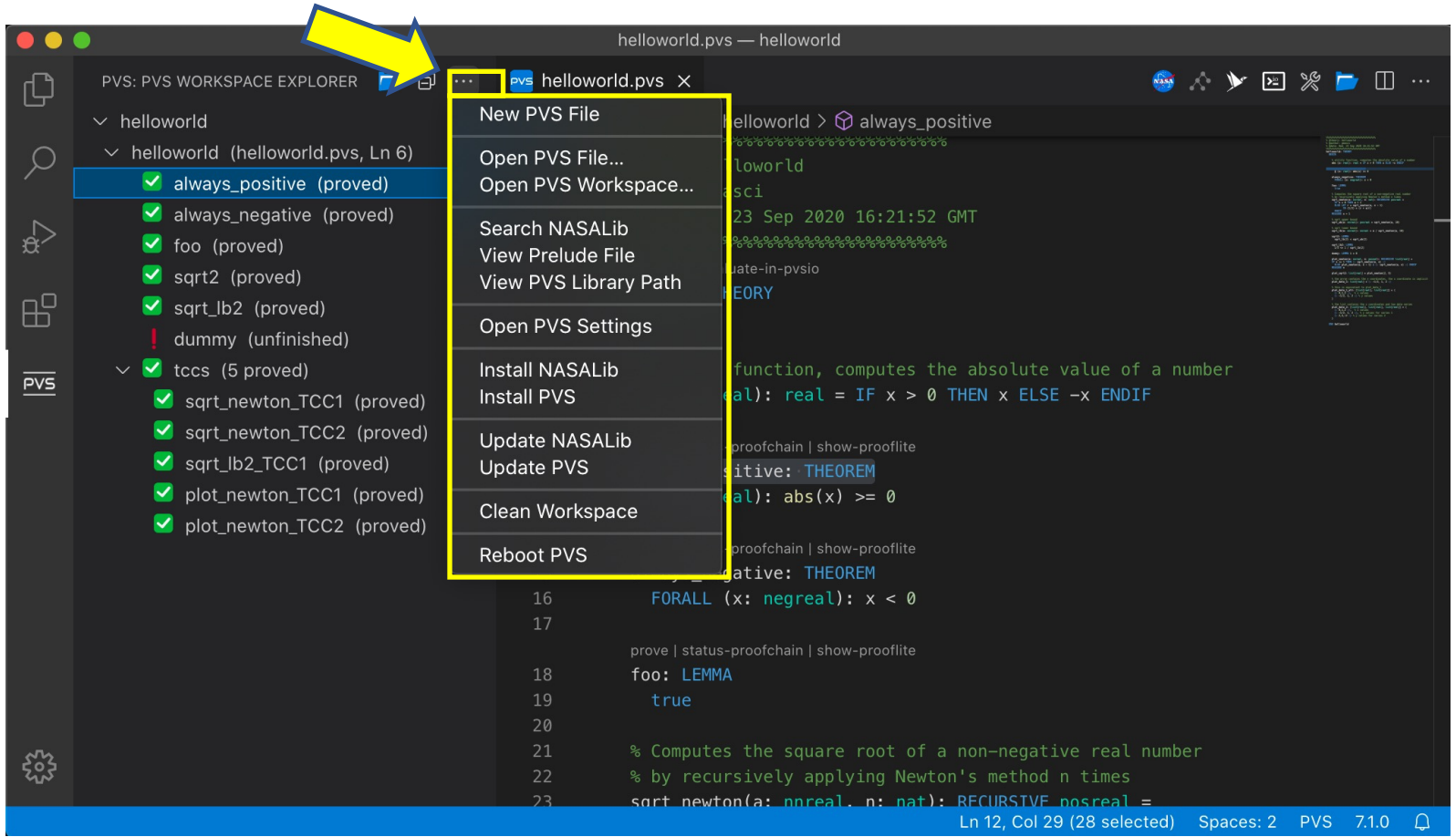
PVS Workspace Explorer (cont'd)

Right click on a theory/theorem to open a menu with actions on files, theories, and theorems



PVS Workspace Explorer (cont'd)

The header menu **...** provides access to workspace actions (e.g., 'New PVS File') and recovery actions (e.g., 'Reboot PVS')



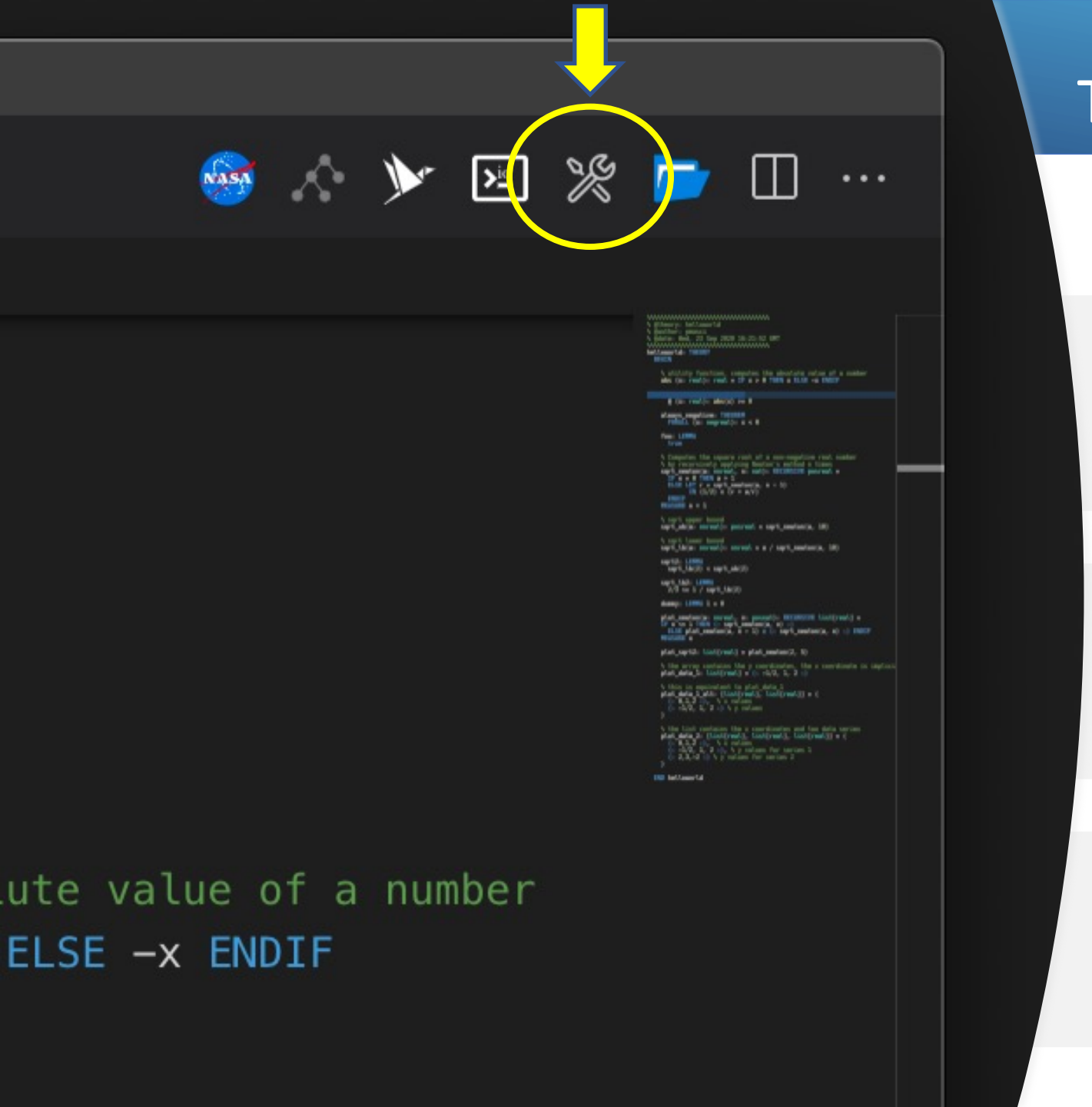
Editing Theories and Theorems

```
helloworld.pvs
helloworld > helloworld.pvs > ...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @theory: helloworld
% @author: pmasci
% @date: Wed, 23 Sep 2020 16:21:52 GMT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
typecheck-file | evaluate-in-pvsio
helloworld: THEORY
  BEGIN

    % utility function, computes the absolute value of a number
    abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF

    prove | show-proofl
    always_positi
      FORALL (x:

    prove | show-proofl
    always_negati
      FORALL (x: negreal): x < 0

    prove | show-prooflite
    foo: LEMMA
      true

    % Computes the square root of a non-negative real number
    % by recursively applying Newton's method n times
    sqrt_newton(a: nnreal, n: nat): RECURSIVE posreal =
      IF n = 0 THEN a + 1
      ELSE LET r = sqrt_newton(a, n - 1)
             IN (1/2) * (r + a/r)
```

```
negreal:     NONEMPTY_TYPE = {x: nonpos_real | x < 0
nnreal: TYPE = nonneg_real
Builtin type negreal
prelude (Ln 2019, Col 2)
negreal: TYPE+ = {x: nonpos_real | x < 0} CONTAINING
```

- Hover the mouse on a term to view the definition.
- Right click on a term to access more functions, e.g., "go to definition" and "peek definition"
- Use TAB to autocomplete terms and keywords while editing
- Math symbols are entered using latex-style shortcuts (e.g., \forall).
- Snippets are available for common constructs, e.g., if-then-else
- Inline actionable commands are available, to typecheck theories and prove theorems

Typechecking

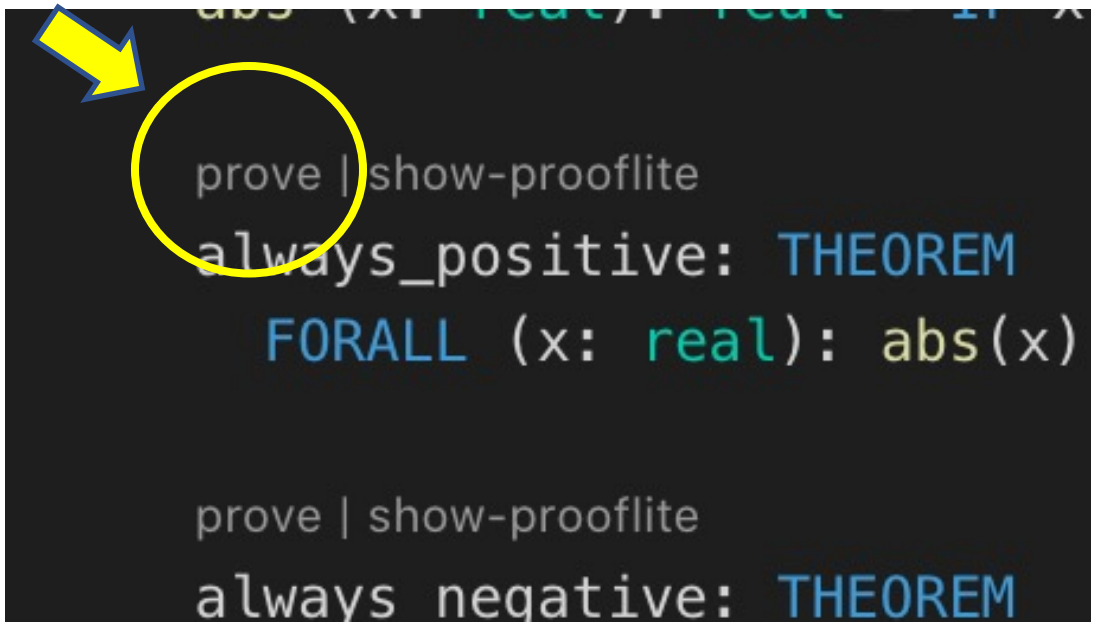


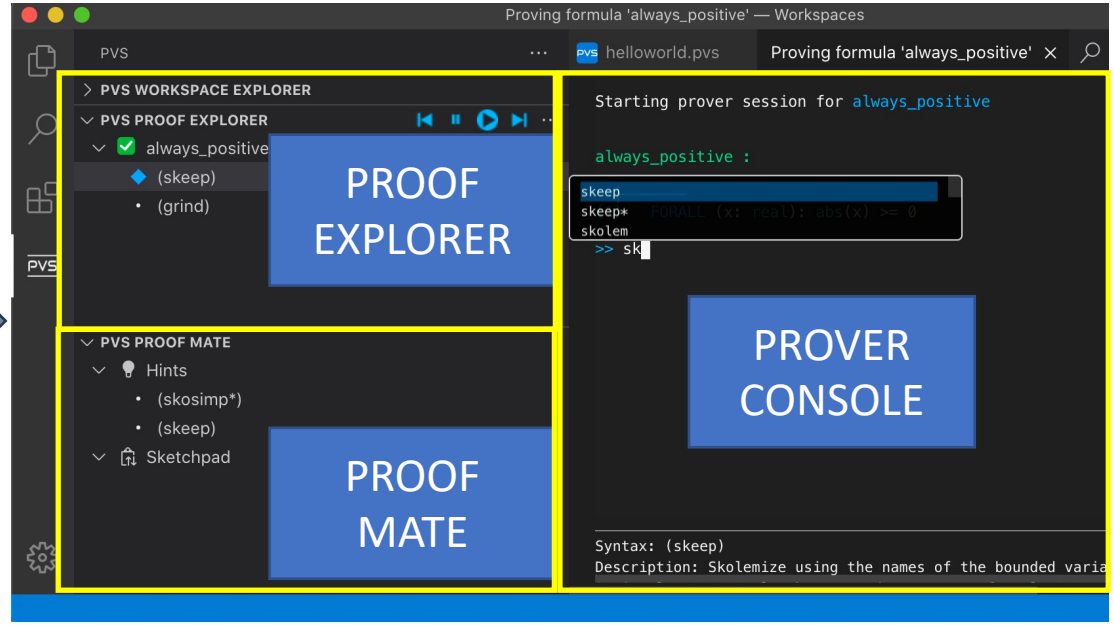
To typecheck the file open in the editor, click the **build icon** in the editor toolbar

If the theory typechecks correctly, you can proceed to proving theorems

Errors will be underlined with a red squiggle and listed in the Problems panel

Proving



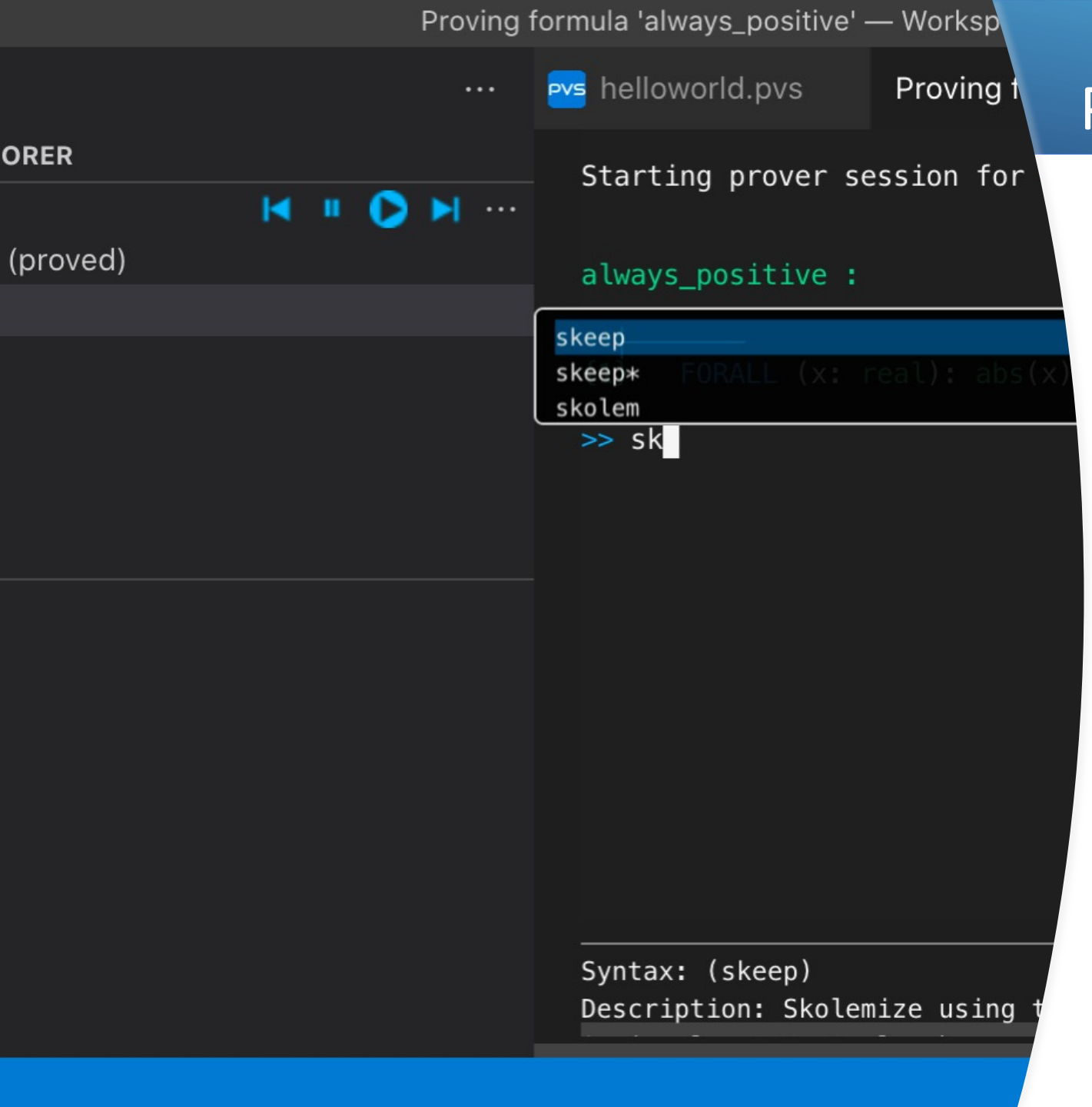


To prove a theorem, click the actionable command **prove** displayed inline above the theorem name.

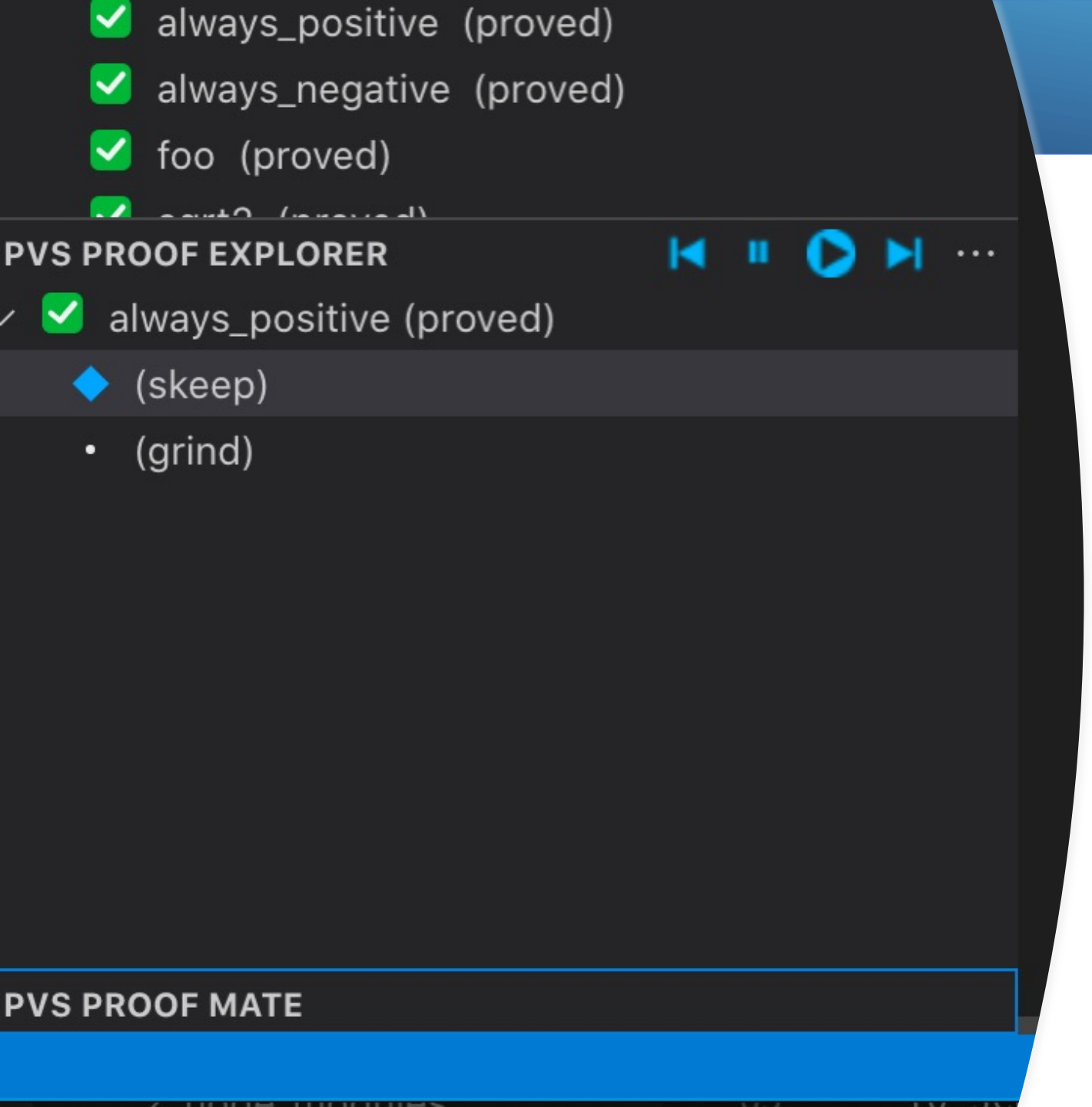
Three new views will be opened:

- Proof Explorer
- Proof Mate
- Prover Console

Prover Console



- The **Prover Console** is the main way to interact with the theorem prover
- Proof commands are entered at the console prompt
- TAB autocompletes commands
- An integrated help shows useful information about the command being entered
- UP/DOWN arrow keys provide access to the command history



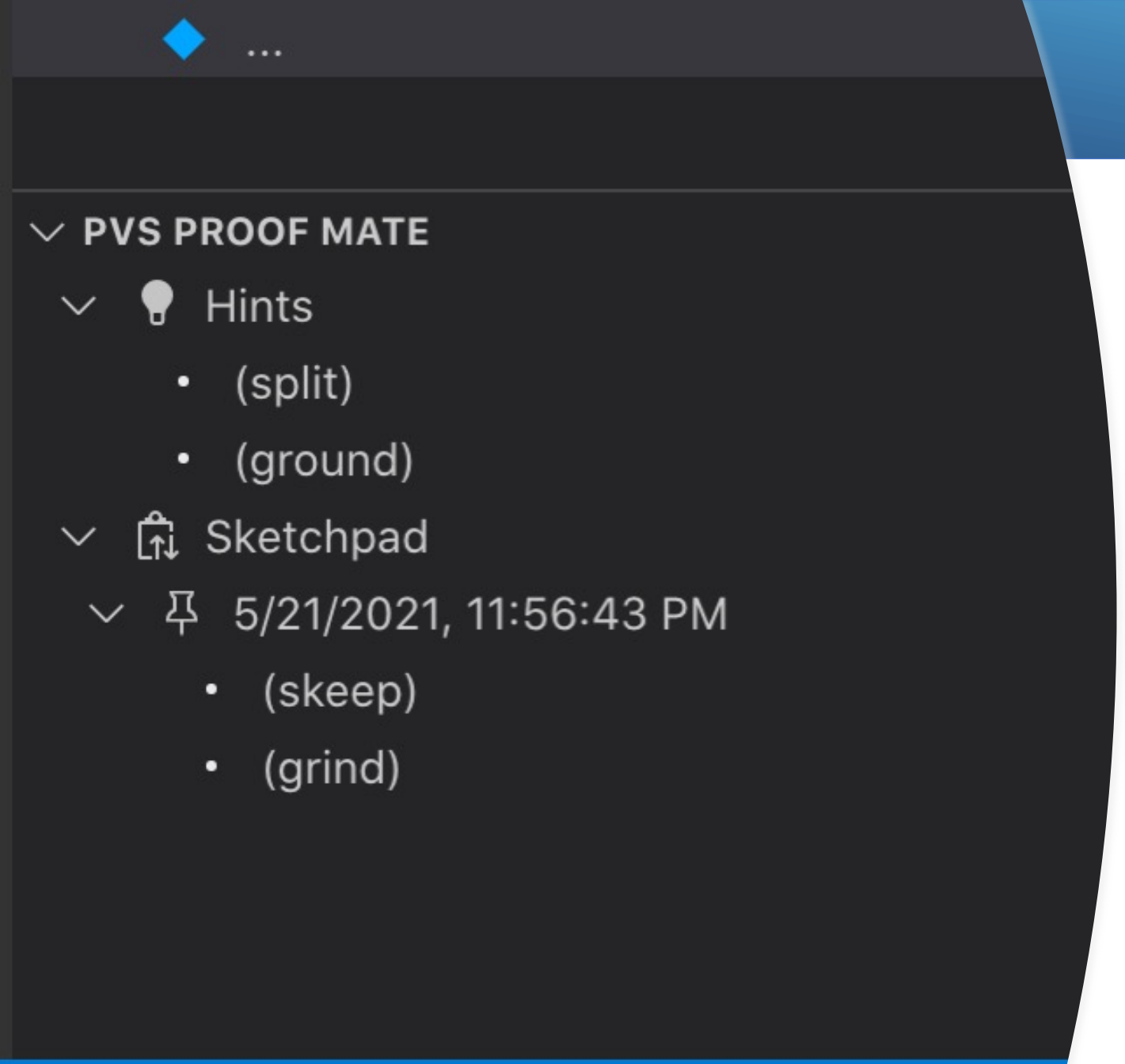
Proof Explorer

Proof Explorer shows the proof tree.

An integrated toolbar has a **play button** for re-running the proof, a **forward button** for stepping proof commands, and a **back button** for rewinding (i.e., undoing) proof commands.

Right click on a proof command to access additional functionalities, e.g., edit/cut/ paste/fast-forward etc.

Proof Mate



Proof Mate is useful when editing/repairing proofs

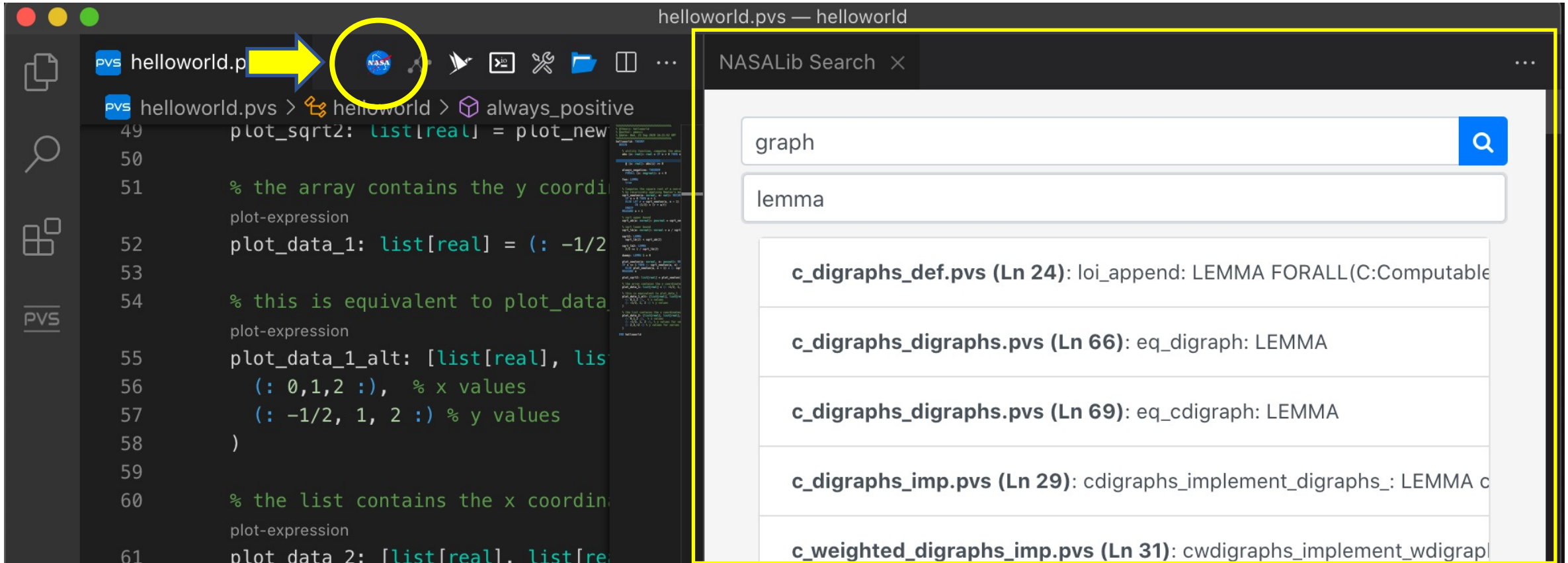
A **sketchpad** collects proof branches that become detached from the proof tree, to facilitate inspection of problems and re-use of proof commands

Hints about proof commands that could be used to make progress with the proof (the heuristics are really simple for now, more to come!)

Additional functionalities of VSCode-PVS

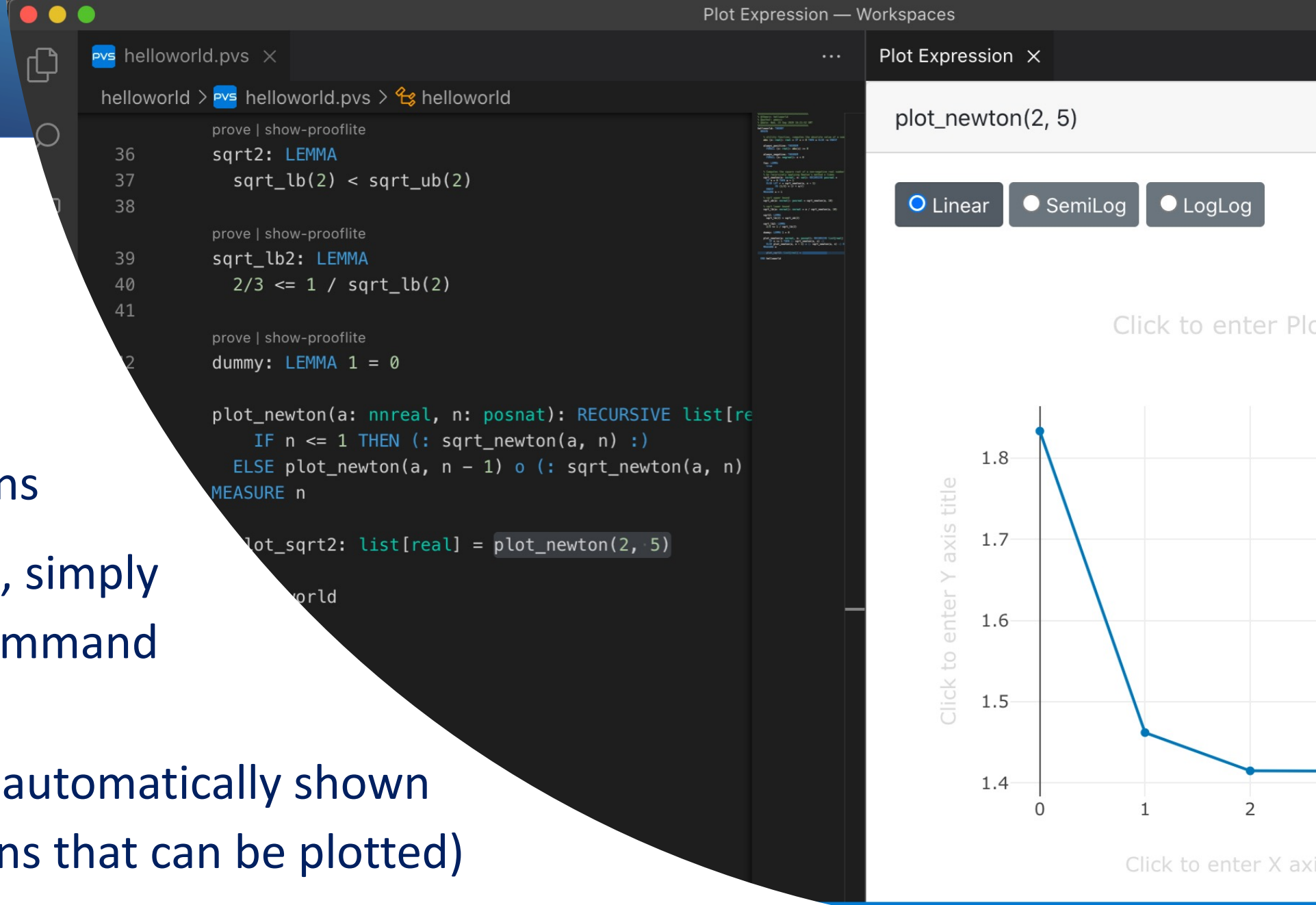
Search NASALib

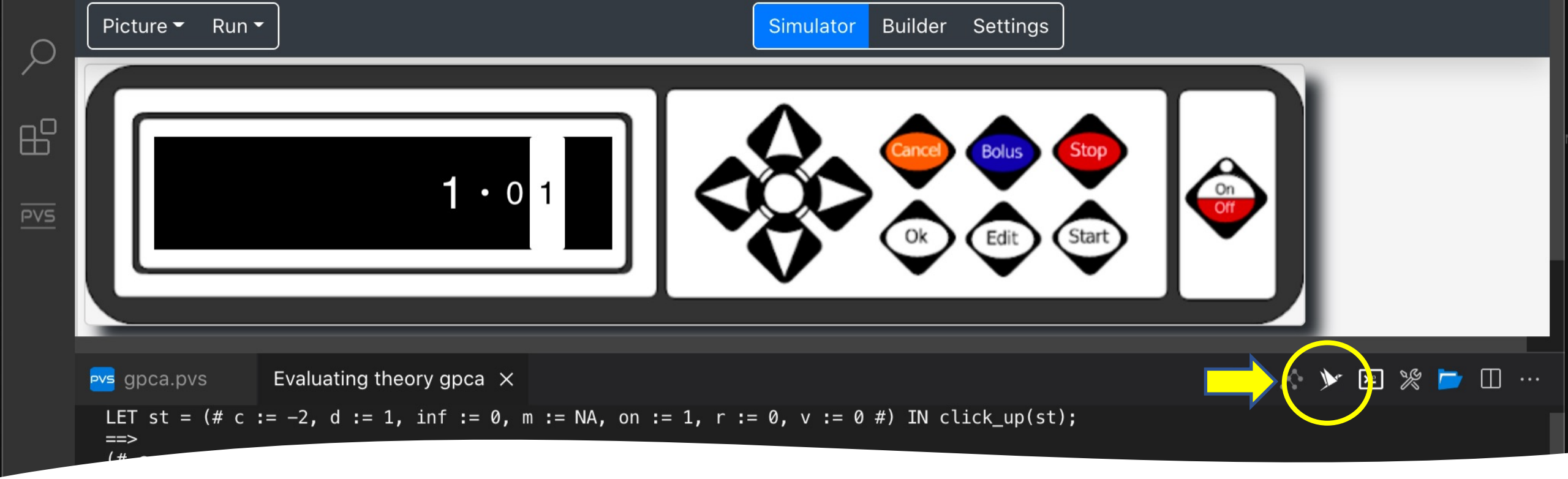
- VSCode-PVS provides a simplified interface to search definitions and lemmas in NASALib, an extensive library created and maintained by our group at NASA Langley.
- To open the search panel, click the NASA meatball logo in the editor toolbar.



Plot

- Create visual diagrams based on PVS expressions
- To plot a diagram, simply click the inline command **plot-expression** (the command is automatically shown next to expressions that can be plotted)





Rapid Prototyping

- Interactive prototypes can be created based on executable PVS specifications
- To create an interactive prototype: open an executable PVS specification, and click on the prototype icon in the editor toolbar to launch the PVSio-web prototype builder and simulator
- Examples of interactive prototypes can be downloaded at https://github.com/pvsioweb/examples

Key references

- Github Repository: https://github.com/nasa/vscode-pvs
- PVS Google Group: https://groups.google.com/g/pvs-group
- Publications: Paolo Masci and César Muñoz, An Integrated Development Environment for the Prototype Verification System Electronic Proceedings in Theoretical Computer Science (EPTCS) Vol. 310, pp. 35-49, 2019