

VSCoDe-PVS Walkthrough

Paolo Masci

paolo.masci@nianet.org

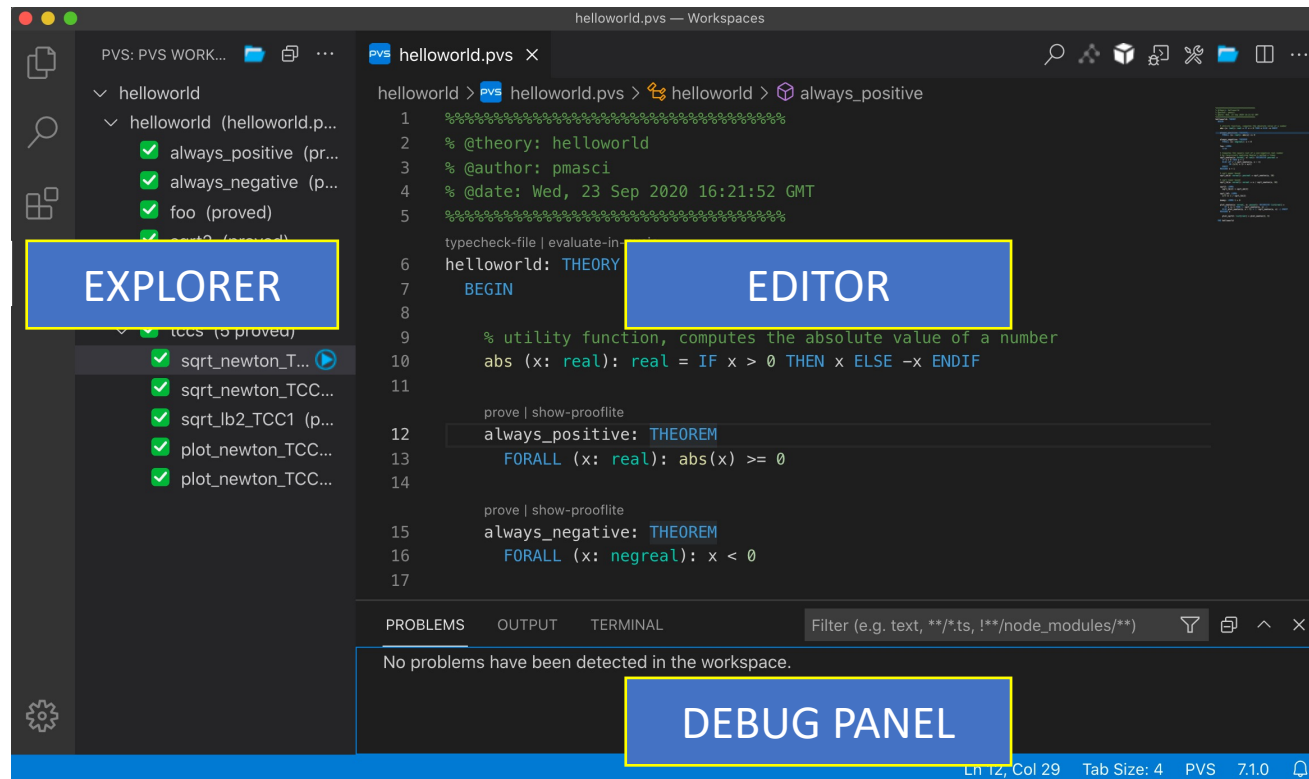
National Institute of Aerospace (NIA)

Part 1

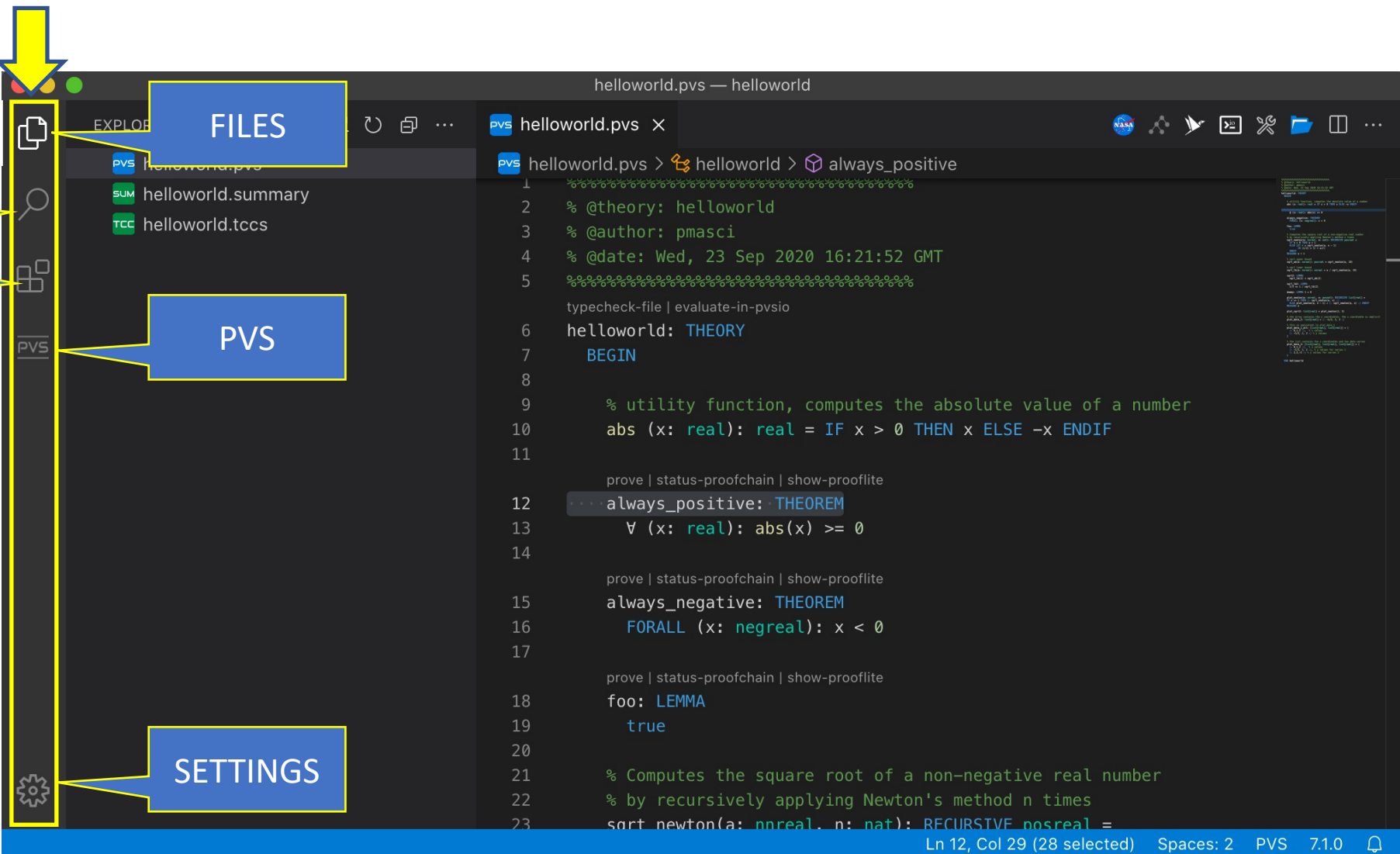
Primer on Visual Studio Code

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



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



The image shows the VS Code Explorer activity bar with several callouts:

- FILES**: Points to the Explorer view icon (a folder icon).
- SEARCH**: Points to the Search view icon (a magnifying glass icon).
- EXTENSIONS**: Points to the Extensions view icon (a puzzle piece icon).
- PVS**: Points to the PVS view icon (a cube icon).
- SETTINGS**: Points to the Settings view icon (a gear icon).

The main editor area displays a PVS file named `helloworld.pvs` with the following content:

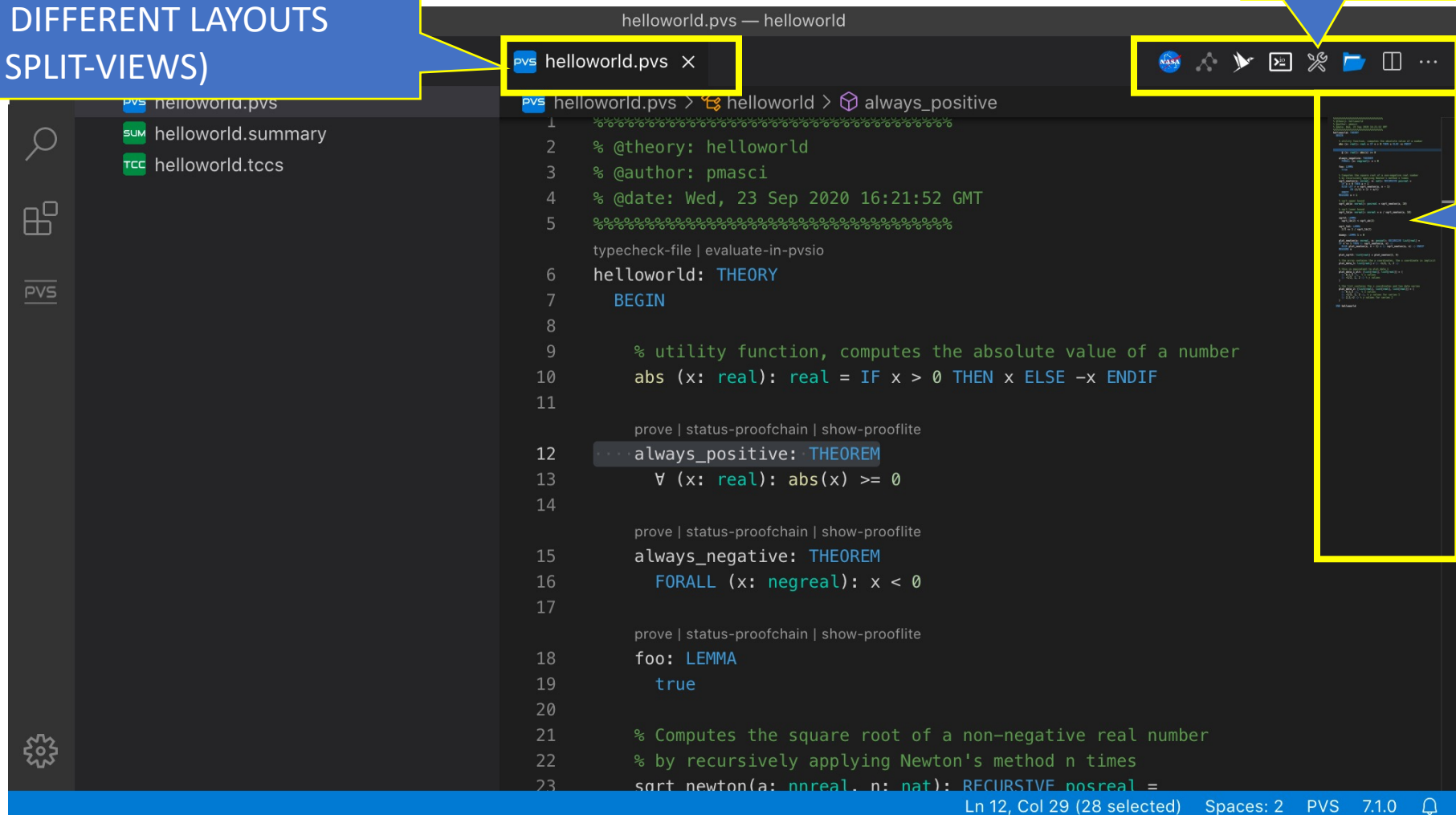
```
helloworld.pvs — helloworld
PVS helloworld.pvs x
PVS helloworld.pvs > helloworld > always_positive
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % @theory: helloworld
3 % @author: pmasci
4 % @date: Wed, 23 Sep 2020 16:21:52 GMT
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 typecheck-file | evaluate-in-pvsio
7 helloworld: THEORY
8 BEGIN
9
10 % utility function, computes the absolute value of a number
11 abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
12
13 prove | status-proofchain | show-prooflite
14 always_positive: THEOREM
15   ∀ (x: real): abs(x) >= 0
16
17 prove | status-proofchain | show-prooflite
18 always_negative: THEOREM
19   FORALL (x: negreal): x < 0
20
21 prove | status-proofchain | show-prooflite
22 foo: LEMMA
23   true
24
25 % Computes the square root of a non-negative real number
26 % by recursively applying Newton's method n times
27 sort newton(a: nreal, n: nat): RECURSIVE noreal =
```

Functionalities can be accessed through point-and-click interactions

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

TOOLBAR

MINIMAP



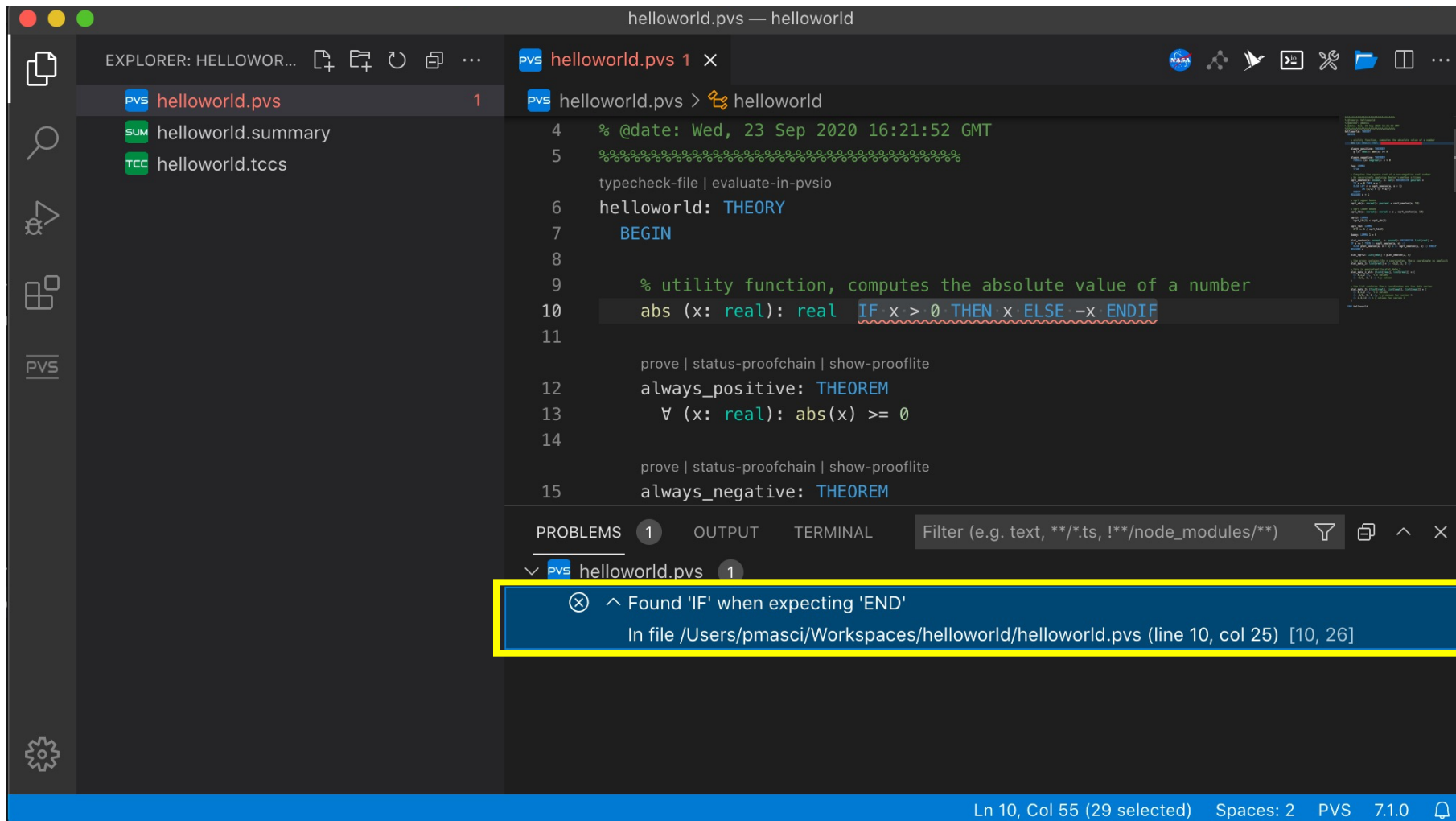
The screenshot displays the PVS editor interface. The main editor window shows the following code:

```
helloworld.pvs — helloworld
pvs helloworld.pvs x
pvs helloworld.pvs > helloworld > always_positive
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % @theory: helloworld
3 % @author: pmasci
4 % @date: Wed, 23 Sep 2020 16:21:52 GMT
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 typecheck-file | evaluate-in-pvsio
7 helloworld: THEORY
8 BEGIN
9
10 % utility function, computes the absolute value of a number
11 abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
12
13 prove | status-proofchain | show-prooflite
14 always_positive: THEOREM
15   ∀ (x: real): abs(x) >= 0
16
17 prove | status-proofchain | show-prooflite
18 always_negative: THEOREM
19   FORALL (x: negreal): x < 0
20
21 prove | status-proofchain | show-prooflite
22 foo: LEMMA
23   true
24
25 % Computes the square root of a non-negative real number
26 % by recursively applying Newton's method n times
27 sort newton(a: nreal, n: nat): RECURSIVE noreal =
```

The interface includes a toolbar with icons for search, save, undo, redo, and other editor functions. A minimap on the right side provides a visual overview of the code structure. The status bar at the bottom indicates the current cursor position: 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



The screenshot shows a code editor window titled "helloworld.pvs — helloworld". The editor displays the following code:

```
4  % @date: Wed, 23 Sep 2020 16:21:52 GMT
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  helloworld: THEORY
7  BEGIN
8
9  % utility function, computes the absolute value of a number
10 abs (x: real): real IF x > 0 THEN x ELSE -x ENDIF
11
12 prove | status-proofchain | show-prooflite
13   always_positive: THEOREM
14      $\forall (x: \text{real}): \text{abs}(x) \geq 0$ 
15
16 prove | status-proofchain | show-prooflite
17   always_negative: THEOREM
```

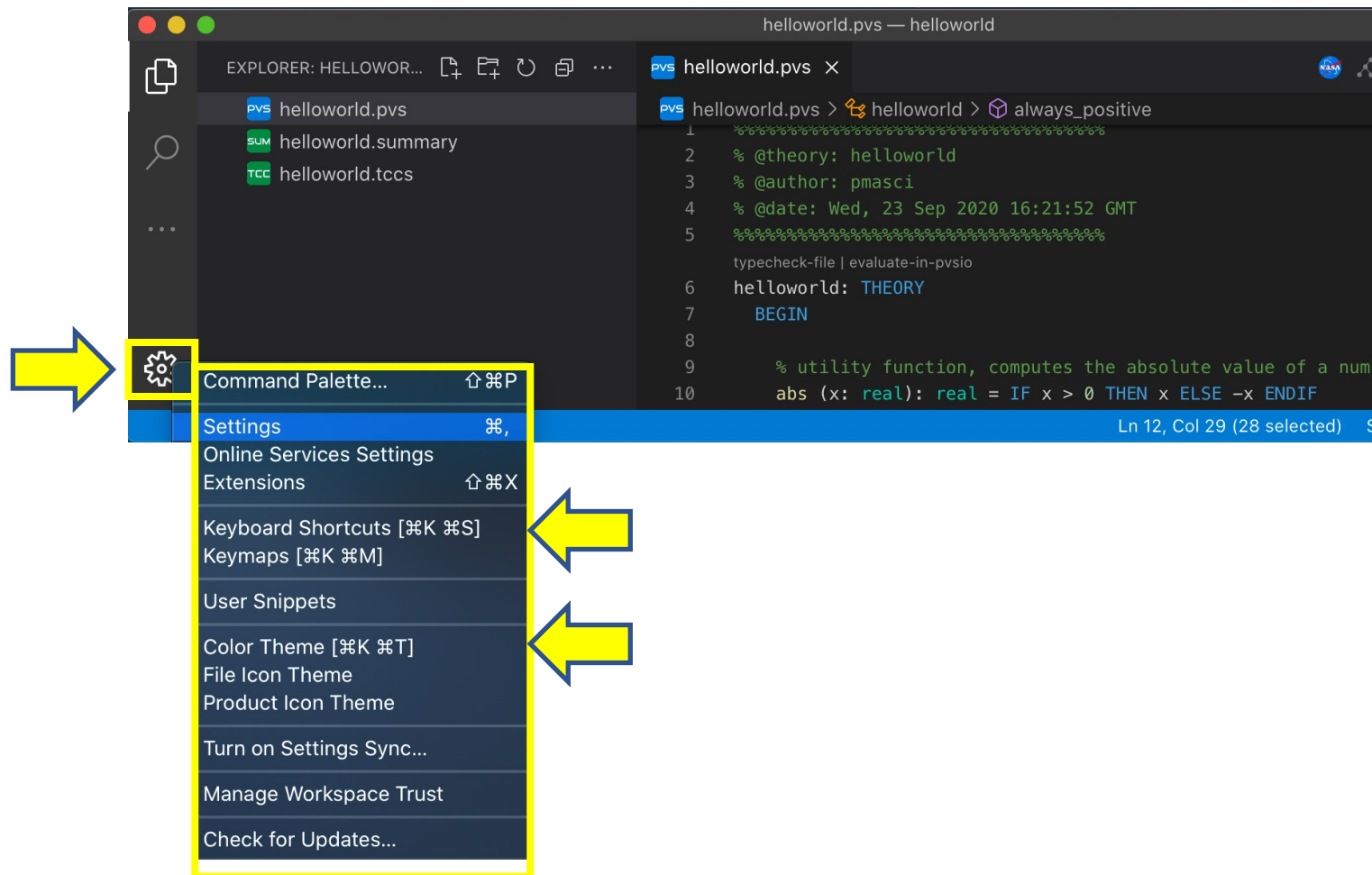
The error message in the PROBLEMS panel is:

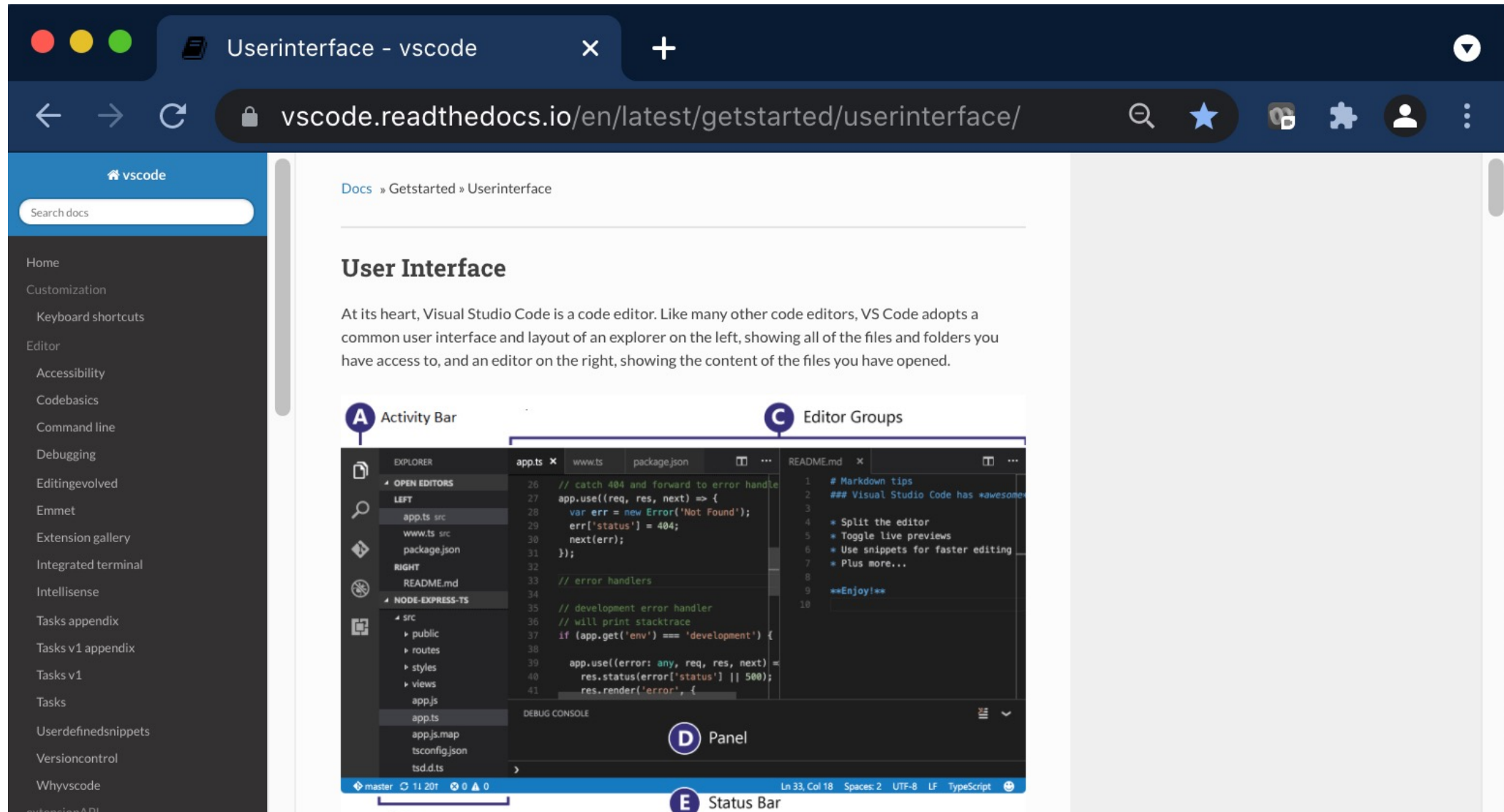
```
⊗ ^ Found 'IF' when expecting 'END'
   In file /Users/pmasci/Workspaces/helloworld/helloworld.pvs (line 10, col 25) [10, 26]
```

A yellow arrow points to the error message. The status bar at the bottom indicates "Ln 10, Col 55 (29 selected) Spaces: 2 PVS 7.1.0".

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





The screenshot shows a web browser window with the address bar displaying `vscode.readthedocs.io/en/latest/getstarted/userinterface/`. The page content includes a navigation menu on the left with items like Home, Customization, Keyboard shortcuts, Editor, Accessibility, Codebasics, Command line, Debugging, Editing evolved, Emmet, Extension gallery, Integrated terminal, Intellisense, Tasks appendix, Tasks v1 appendix, Tasks v1, Tasks, Userdefinedsnippets, Versioncontrol, and Whyvscode. The main content area is titled "User Interface" and contains the following text:

Docs » Getstarted » Userinterface

User Interface

At its heart, Visual Studio Code is a code editor. Like many other code editors, VS Code adopts a common user interface and layout of an explorer on the left, showing all of the files and folders you have access to, and an editor on the right, showing the content of the files you have opened.

The image also features a diagram of the VS Code interface with labels A through E:

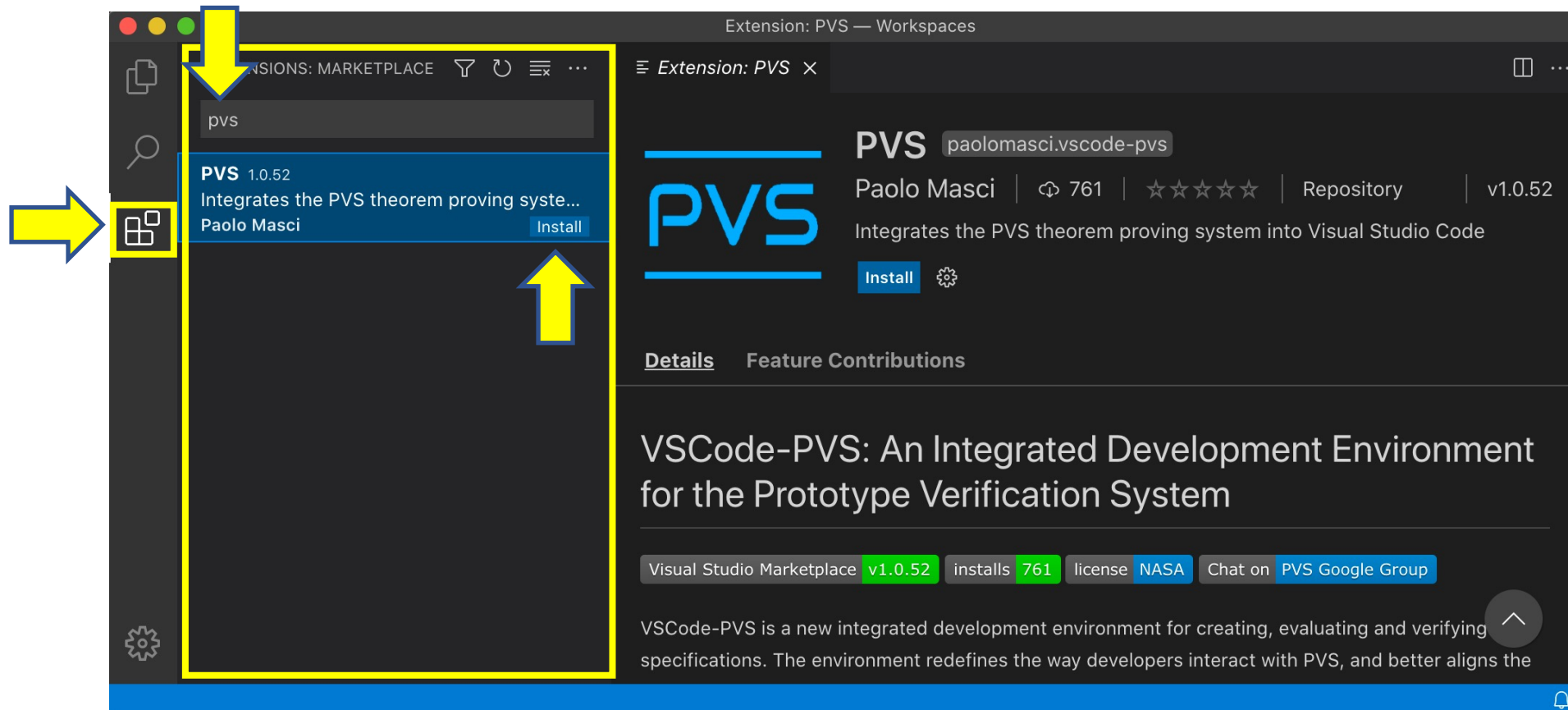
- A Activity Bar**: Points to the left sidebar containing Explorer, Search, Run and Debug, and Source Control.
- C Editor Groups**: Points to the main workspace area where code files are opened and edited.
- D Panel**: Points to the bottom area containing the Debug Console and Output.
- E Status Bar**: Points to the bottom status bar showing the current file, line, column, and encoding information.

Part 2

VSCoDe-PVS

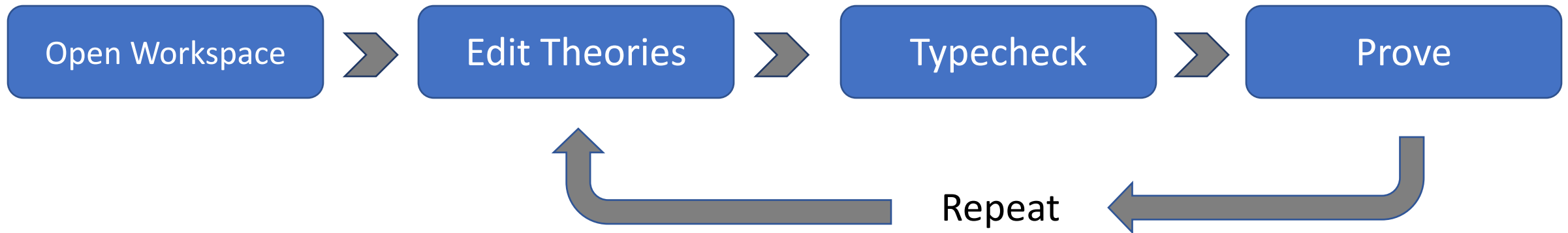
Installing VSCode-PVS

- 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.

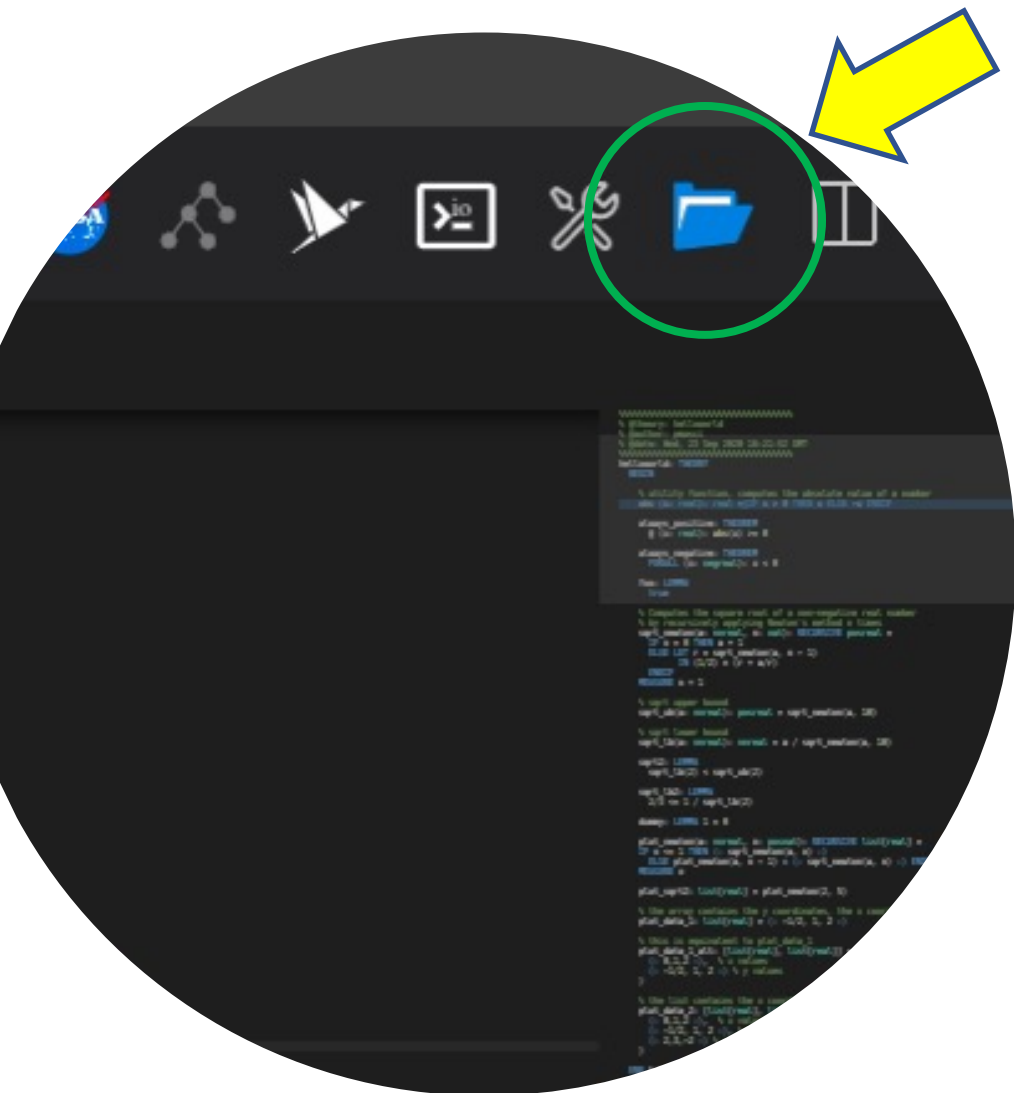


The screenshot shows the Visual Studio Code interface with the Extension Marketplace open. The search bar contains 'pvs'. The search results show the 'PVS' extension by Paolo Masci, version 1.0.52. The extension card is highlighted with a yellow box. A yellow arrow points to the search bar, another yellow arrow points to the extension card, and a third yellow arrow points to the 'Install' button. The extension details page is also visible, showing the 'PVS' logo, the author 'Paolo Masci', and the 'Install' button. The description reads: 'Integrates the PVS theorem proving system into Visual Studio Code'. The version is 'v1.0.52' and it has 761 installs. The license is 'NASA'. There is a link to 'Chat on PVS Google Group'. The extension is described as 'An Integrated Development Environment for the Prototype Verification System'.

Workflow in VSCode-PVS



Opening a PVS Workspace



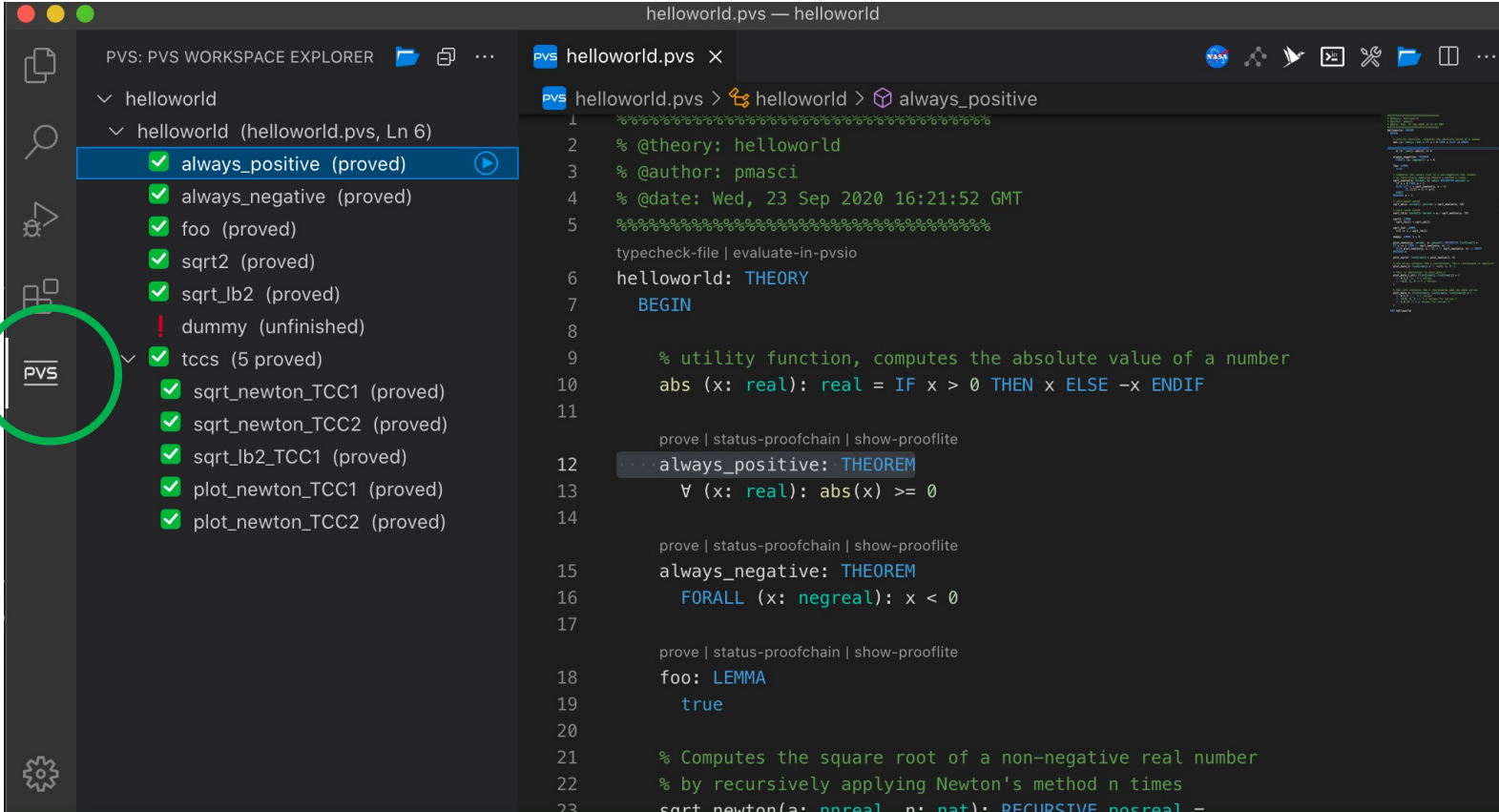
A PVS workspace is a folder that contains (or will contain) PVS files.

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

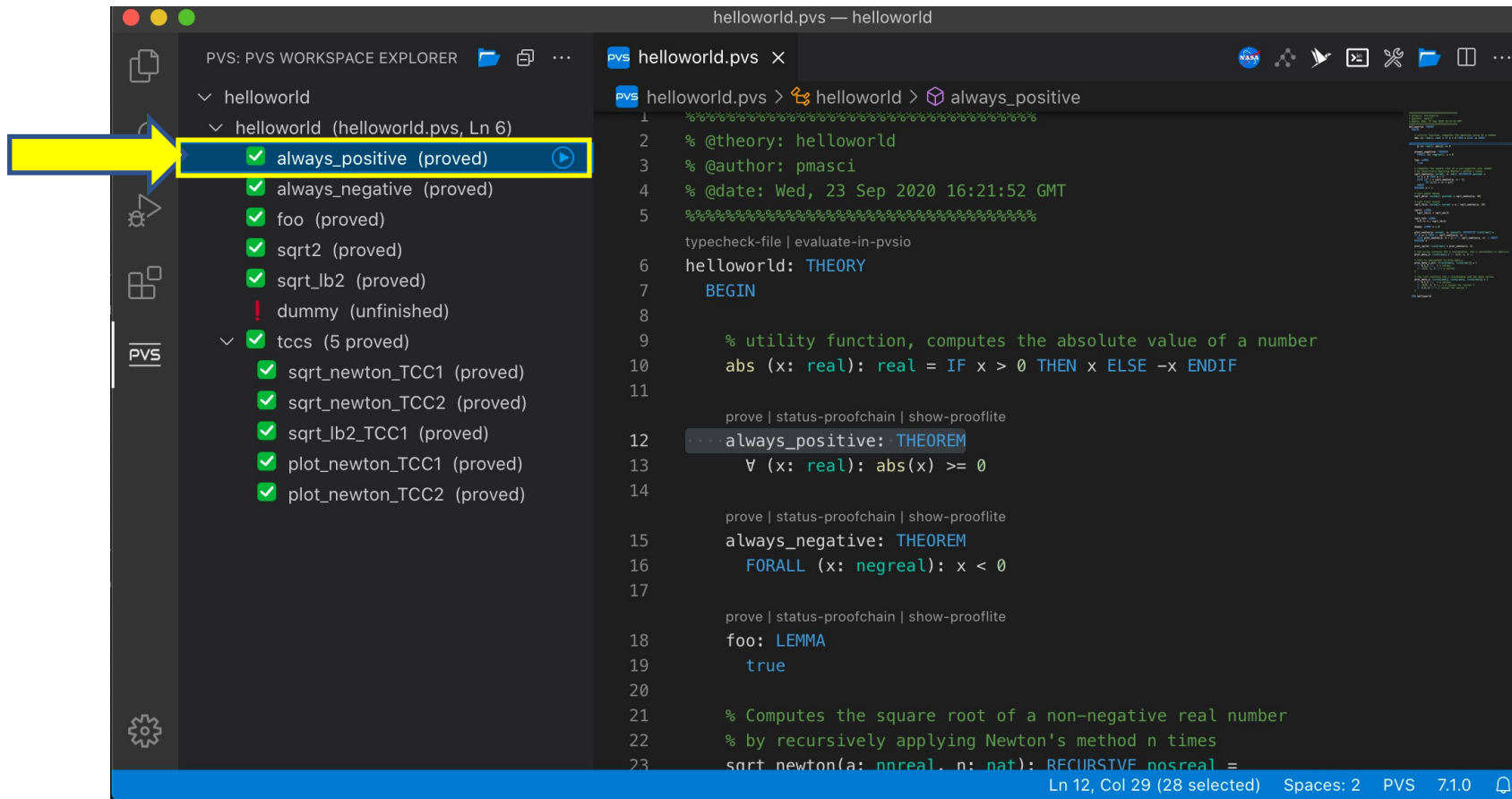


The screenshot shows the PVS Workspace Explorer interface. On the left, a tree view displays a list of theories and theorems under the 'helloworld' workspace. The 'PVS' icon is highlighted with a green circle and a yellow arrow. The main editor area shows the code for the 'always_positive' theory, including comments, a theorem statement, and proof commands.

```
helloworld.pvs — helloworld
PVS: PVS WORKSPACE EXPLORER
helloworld
  helloworld (helloworld.pvs, Ln 6)
    ✓ always_positive (proved)
    ✓ always_negative (proved)
    ✓ foo (proved)
    ✓ sqrt2 (proved)
    ✓ sqrt_lb2 (proved)
    ! dummy (unfinished)
    ✓ tccs (5 proved)
    ✓ sqrt_newton_TCC1 (proved)
    ✓ sqrt_newton_TCC2 (proved)
    ✓ sqrt_lb2_TCC1 (proved)
    ✓ plot_newton_TCC1 (proved)
    ✓ plot_newton_TCC2 (proved)
PVS
helloworld.pvs > helloworld > always_positive
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % @theory: helloworld
3 % @author: pmasci
4 % @date: Wed, 23 Sep 2020 16:21:52 GMT
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 typecheck-file | evaluate-in-pvsio
7 helloworld: THEORY
8 BEGIN
9
10 % utility function, computes the absolute value of a number
11 abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
12
13 prove | status-proofchain | show-prooflite
14 ... always_positive: THEOREM
15   ∀ (x: real): abs(x) >= 0
16
17 prove | status-proofchain | show-prooflite
18 always_negative: THEOREM
19   FORALL (x: negreal): x < 0
20
21 prove | status-proofchain | show-prooflite
22 foo: LEMMA
23   true
24
25 % Computes the square root of a non-negative real number
26 % by recursively applying Newton's method n times
27 sqrt_newton(a: nreal, n: nat): RECURSIVE noreal =
```

PVS Workspace Explorer (cont'd)

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

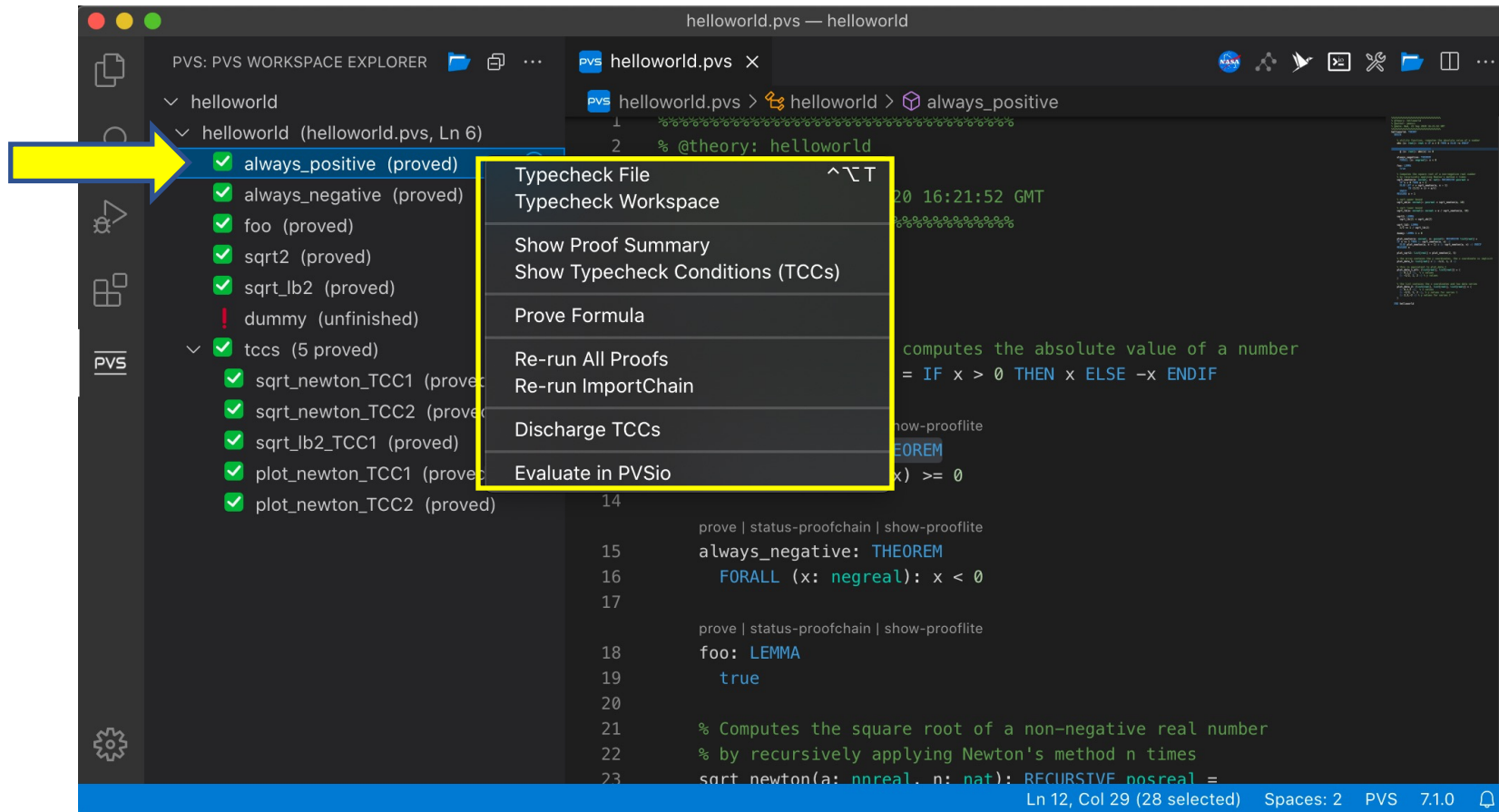


The screenshot displays the PVS Workspace Explorer on the left and the PVS editor on the right. A yellow arrow points to the 'always_positive (proved)' entry in the workspace explorer. The editor shows the corresponding PVS code for the 'always_positive' theorem.


```
helloworld.pvs — helloworld
PVS: PVS WORKSPACE EXPLORER
helloworld
  helloworld (helloworld.pvs, Ln 6)
    ✓ always_positive (proved)
    ✓ always_negative (proved)
    ✓ foo (proved)
    ✓ sqrt2 (proved)
    ✓ sqrt_lb2 (proved)
    ! dummy (unfinished)
  tccs (5 proved)
    ✓ sqrt_newton_TCC1 (proved)
    ✓ sqrt_newton_TCC2 (proved)
    ✓ sqrt_lb2_TCC1 (proved)
    ✓ plot_newton_TCC1 (proved)
    ✓ plot_newton_TCC2 (proved)
PVS
helloworld.pvs > helloworld > always_positive
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % @theory: helloworld
3 % @author: pmasci
4 % @date: Wed, 23 Sep 2020 16:21:52 GMT
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 typecheck-file | evaluate-in-pvsio
7 helloworld: THEORY
8 BEGIN
9
10 % utility function, computes the absolute value of a number
11 abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
12
13 prove | status-proofchain | show-prooflite
14 ... always_positive: THEOREM
15   ∀ (x: real): abs(x) >= 0
16
17 prove | status-proofchain | show-prooflite
18 always_negative: THEOREM
19   FORALL (x: negreal): x < 0
20
21 prove | status-proofchain | show-prooflite
22 foo: LEMMA
23   true
24
25 % Computes the square root of a non-negative real number
26 % by recursively applying Newton's method n times
27 sqrt_newton(a: nreal, n: nat): RECURSIVE noreal =
```

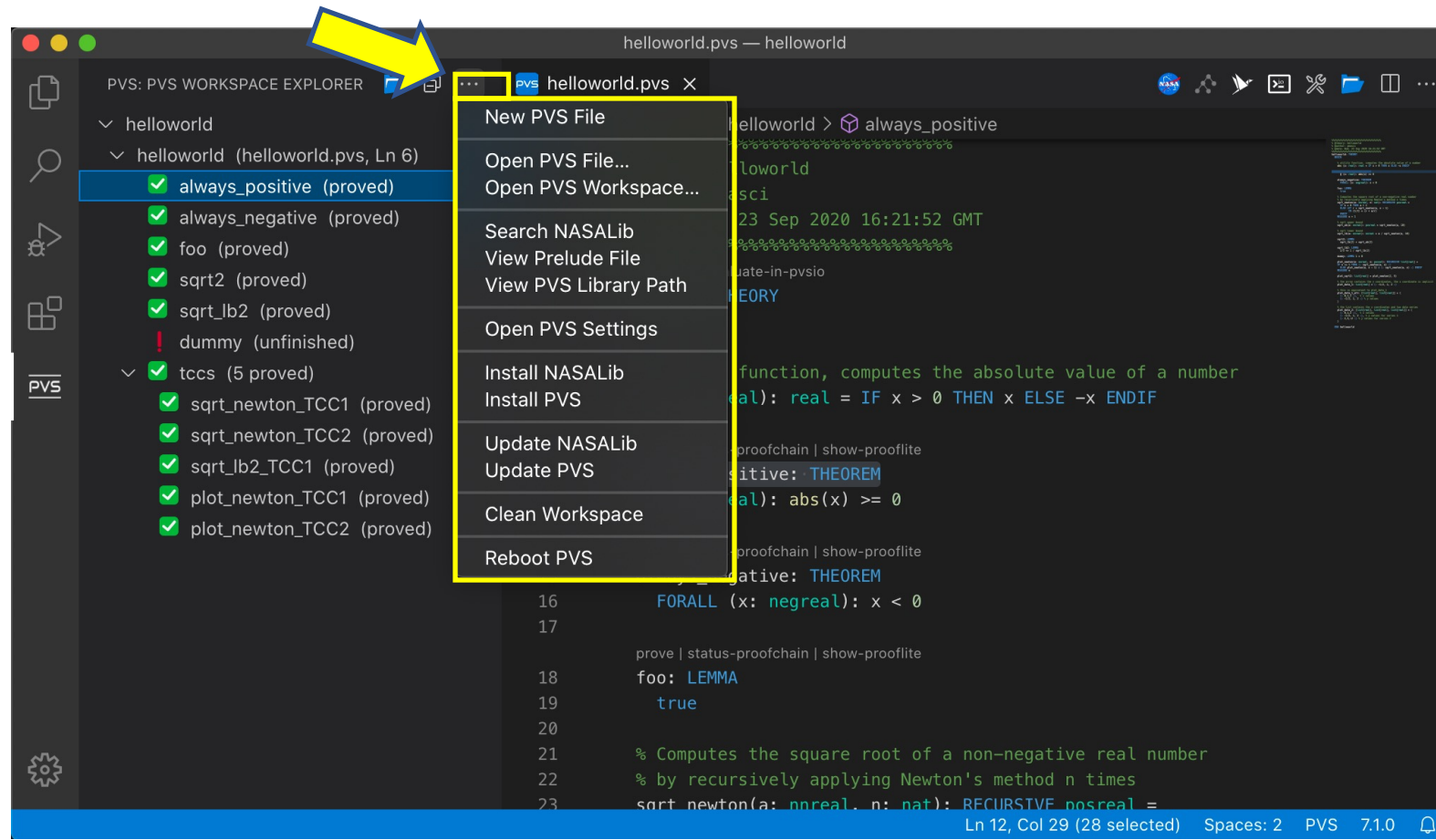

PVS Workspace Explorer (cont'd)

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

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

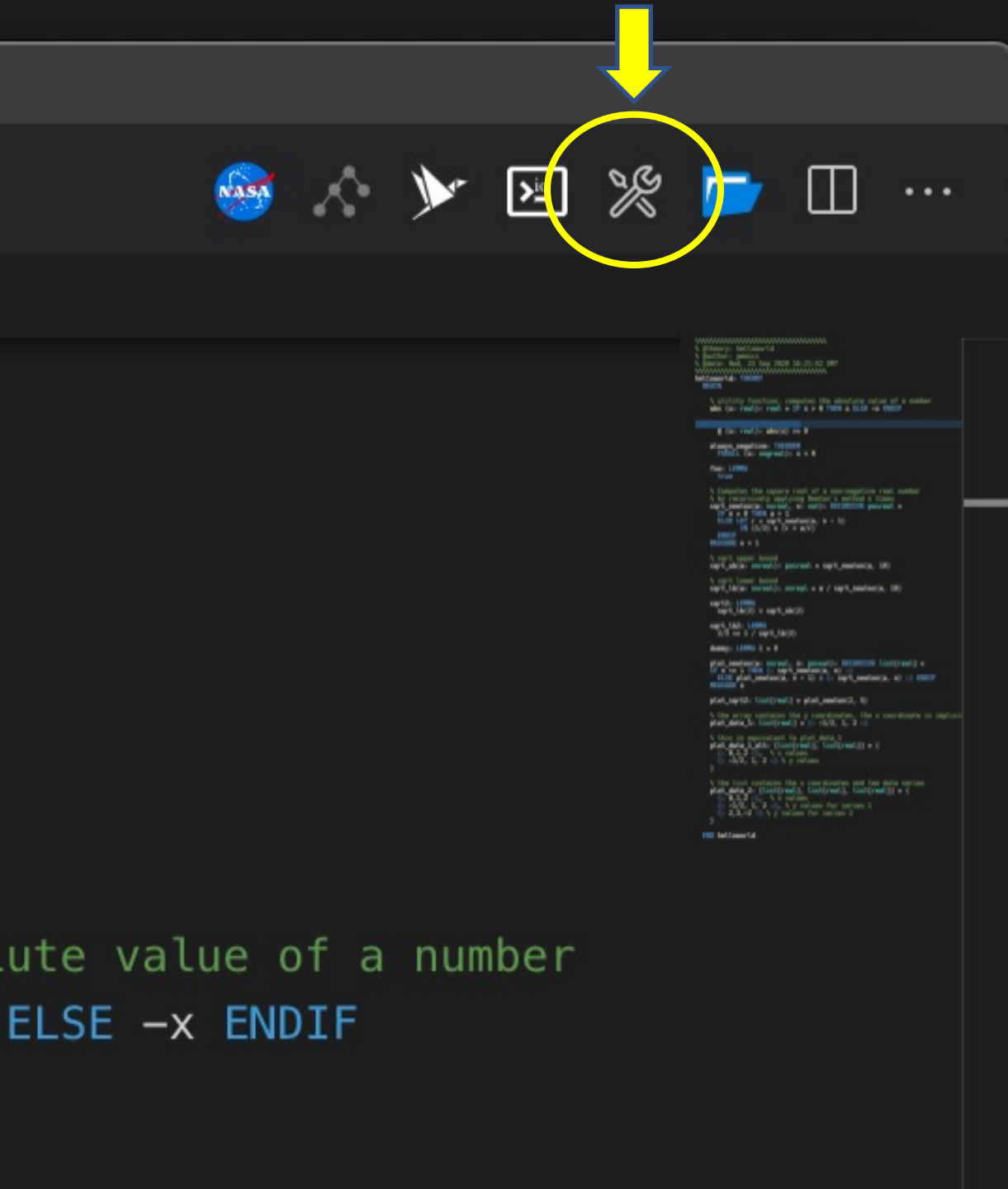
Inline actionable commands are available, to typecheck theories and prove theorems

```

helloworld.pvs x
helloworld > PVS helloworld.pvs > ...
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % @theory: helloworld
3  % @author: pmasci
4  % @date: Wed, 23 Sep 2020 16:21:52 GMT
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
typecheck-file | evaluate-in-pvsio
6  helloworld: THEORY
7  BEGIN
8
9  % utility function, computes the absolute value of a number
10 abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
11
12 prove | show-prooflite
13 always_positi
14   FORALL (x:
15     negreal: NONEMPTY_TYPE = {x: nonpos_real | x < 0}
16     nnreal: TYPE = nonneg_real
17     prelude (Ln 2019, Col 2)
18     negreal: TYPE+ = {x: nonpos_real | x < 0} CONTAINING
19     FORALL (x: negreal): x < 0
20
21 prove | show-prooflite
22 foo: LEMMA
23   true
24
25 % Computes the square root of a non-negative real number
26 % 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)

```

Typechecking



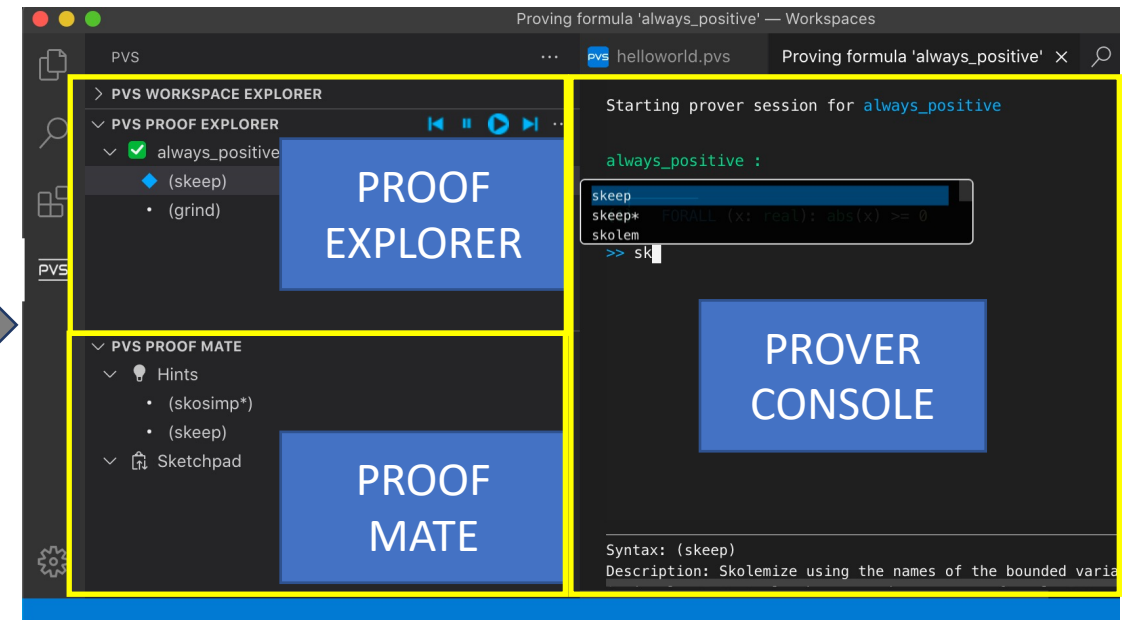
To typecheck the file open in the editor, click the **build icon** in the editor toolbar

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

```
abs (x: real): real  
1 1 X  
prove | show-prooflite  
always_positive: THEOREM  
FORALL (x: real): abs(x)  
prove | show-prooflite  
always negative: THEOREM
```



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

ORER



(proved)

pvs helloworld.pvs

Proving f

Starting prover session for

`always_positive :`

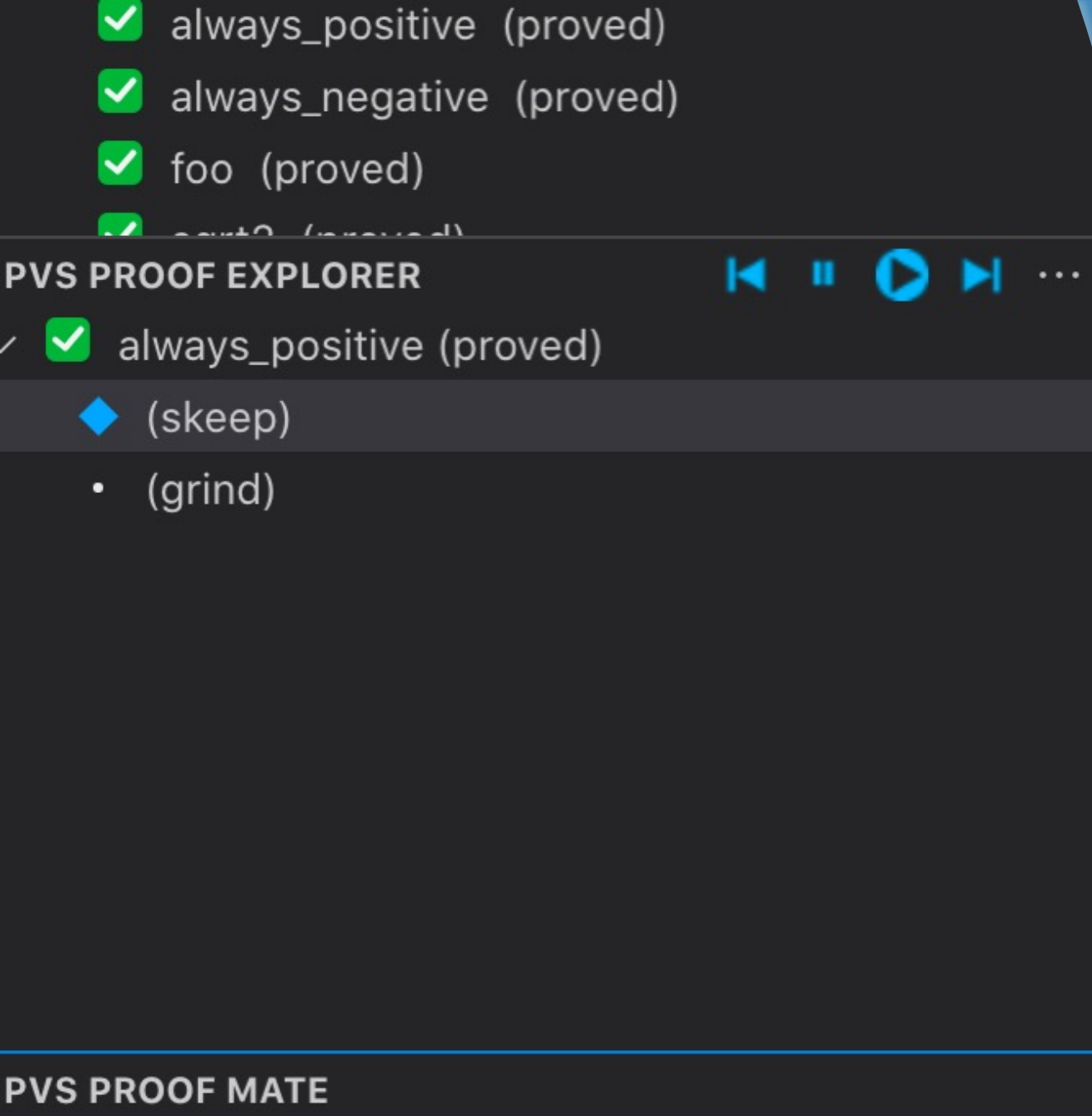
```
skeep  
skeep* FORALL (x: real): abs(x)  
skolem  
>> sk
```

Syntax: (skeep)

Description: Skolemize using t

- 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!)

✓ PVS PROOF MATE

✓ Hints

- (split)
- (ground)

✓ Sketchpad

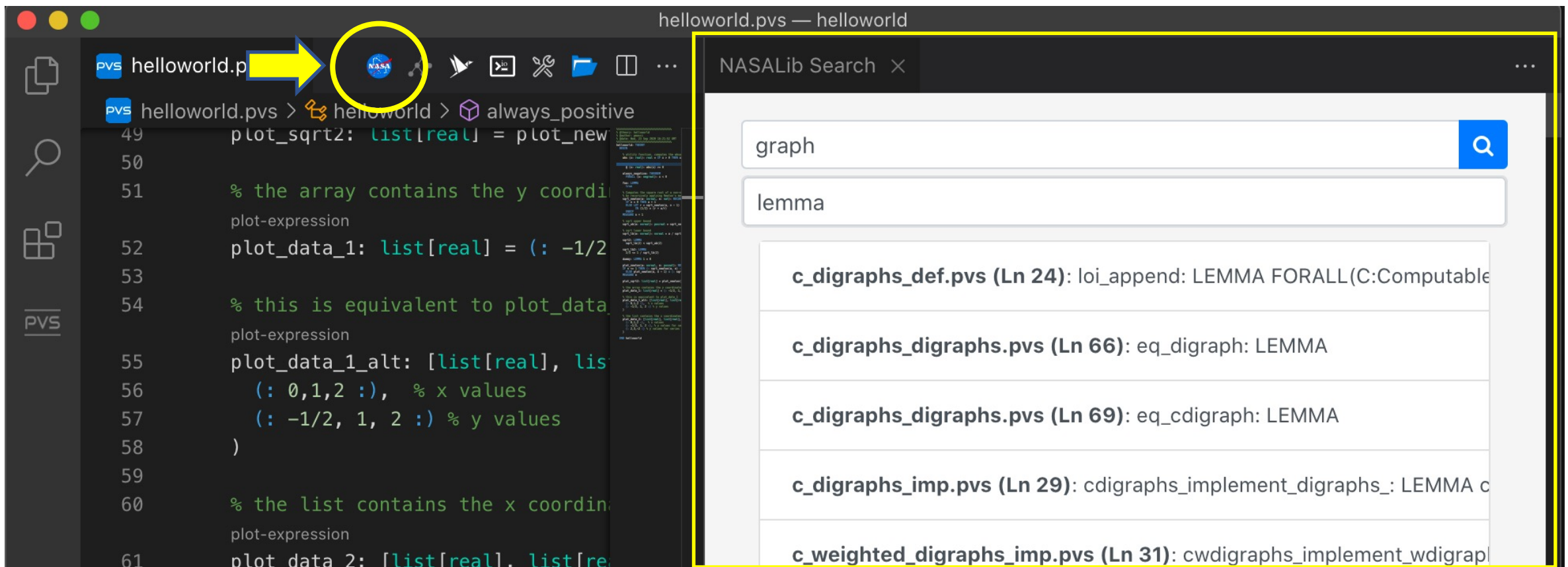
✓ 5/21/2021, 11:56:43 PM

- (skeep)
- (grind)

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.



The screenshot shows the VSCode-PVS editor interface. The toolbar at the top contains several icons, with the NASA logo (a blue circle with a white meatball) circled in yellow. A yellow arrow points from the left towards the NASA logo. The editor window displays a PVS script named 'helloworld.pvs' with the following code:

```
helloworld.pvs > helloworld > always_positive
49 plot_sqrt2: list[real] = plot_new
50
51 % the array contains the y coordi
plot-expression
52 plot_data_1: list[real] = (: -1/2
53
54 % this is equivalent to plot_data
plot-expression
55 plot_data_1_alt: [list[real], lis
56 (: 0,1,2 :), % x values
57 (: -1/2, 1, 2 :) % y values
58 )
59
60 % the list contains the x coordin
plot-expression
61 plot data 2: [list[real]. list[re
```

The NASALib Search panel is open on the right, showing a search input field with the text 'graph' and a search button. Below the input field, the search results are displayed as a list of items:

- lemma
- c_digraphs_def.pvs (Ln 24): loi_append: LEMMA FORALL(C:Computable
- c_digraphs_digraphs.pvs (Ln 66): eq_digraph: LEMMA
- c_digraphs_digraphs.pvs (Ln 69): eq_cdigraph: LEMMA
- c_digraphs_imp.pvs (Ln 29): cdigraphs_implement_digraphs_: LEMMA c
- c_weighted_digraphs_imp.pvs (Ln 31): cwdigraphs_implement_wdigrap

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)

The screenshot shows a PVS workspace with a code editor and a plot window. The code editor displays the following PVS code:

```
helloworld.pvs > helloworld.pvs > helloworld
prove | show-proofLite
sqrt2: LEMMA
  sqrt_lb(2) < sqrt_ub(2)

prove | show-proofLite
sqrt_lb2: LEMMA
  2/3 <= 1 / sqrt_lb(2)

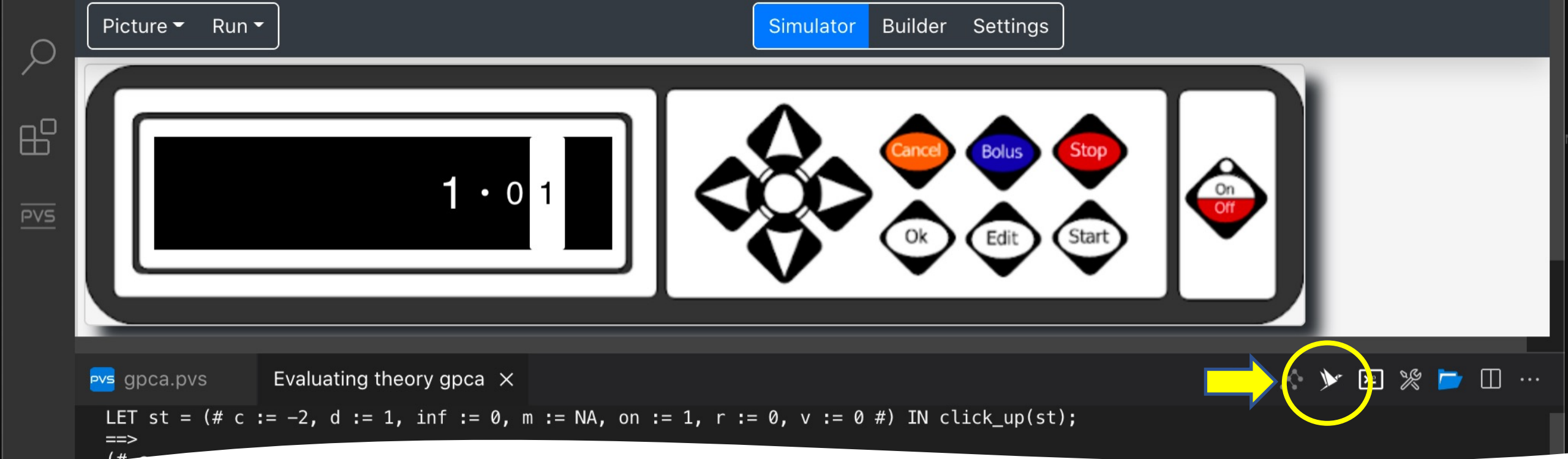
prove | show-proofLite
dummy: LEMMA 1 = 0

plot_newton(a: nnreal, n: posnat): RECURSIVE list[real] :=
  IF n <= 1 THEN (: sqrt_newton(a, n) :)
  ELSE plot_newton(a, n - 1) o (: sqrt_newton(a, n) :)
MEASURE n

plot_sqrt2: list[real] = plot_newton(2, 5)
```

The plot window, titled "Plot Expression", shows the command `plot_newton(2, 5)` and three radio buttons for the plot style: **Linear** (selected), **SemiLog**, and **LogLog**. The plot displays a line graph with three data points connected by a blue line. The Y-axis is labeled "Click to enter Y axis title" and has tick marks at 1.4, 1.5, 1.6, 1.7, and 1.8. The X-axis is labeled "Click to enter X axis title" and has tick marks at 0, 1, and 2. The data points are approximately at (0, 1.85), (1, 1.46), and (2, 1.42).

X	Y
0	1.85
1	1.46
2	1.42



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