(Hybrid) Active Logic Documents and the Ciao Prolog Playground

with an Application to Verification Tutorials

[1,2]Daniela Ferreiro, [1,2]José F. Morales, [3]Salvador Abreu and [1,2]Manuel V. Hermenegildo

[1]Universidad Politécnica de Madrid (UPM) [2]IMDEA Software Institute, Madrid, Spain [3]Nova LINCS / University of Évora, Portugal

MOTIVATION

Traditional teaching materials for Prolog (books, slides, static web pages)

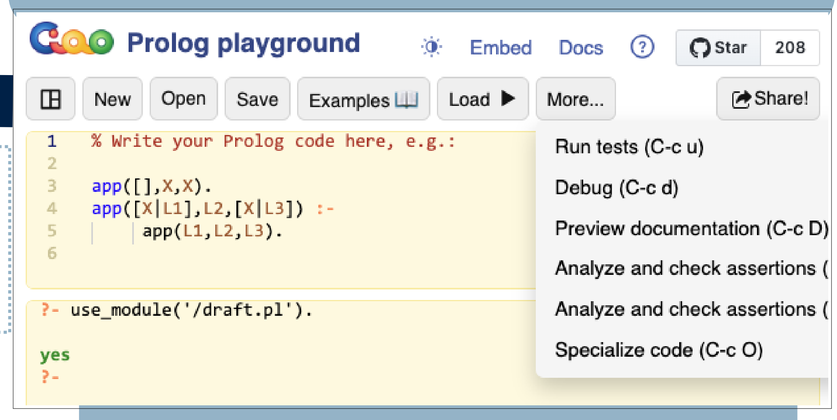
- **Little or no interactivity**
- **Modern web technology enables much richer experiences**

ciao-wasm

Runs the Prolog engine locally!

- ✓ Full, native engine (i.e., not interpreter)
- ✓ Security/privacy
- ✓ Portability, no installation, low/no maintenance...

The Ciao Prolog Playground

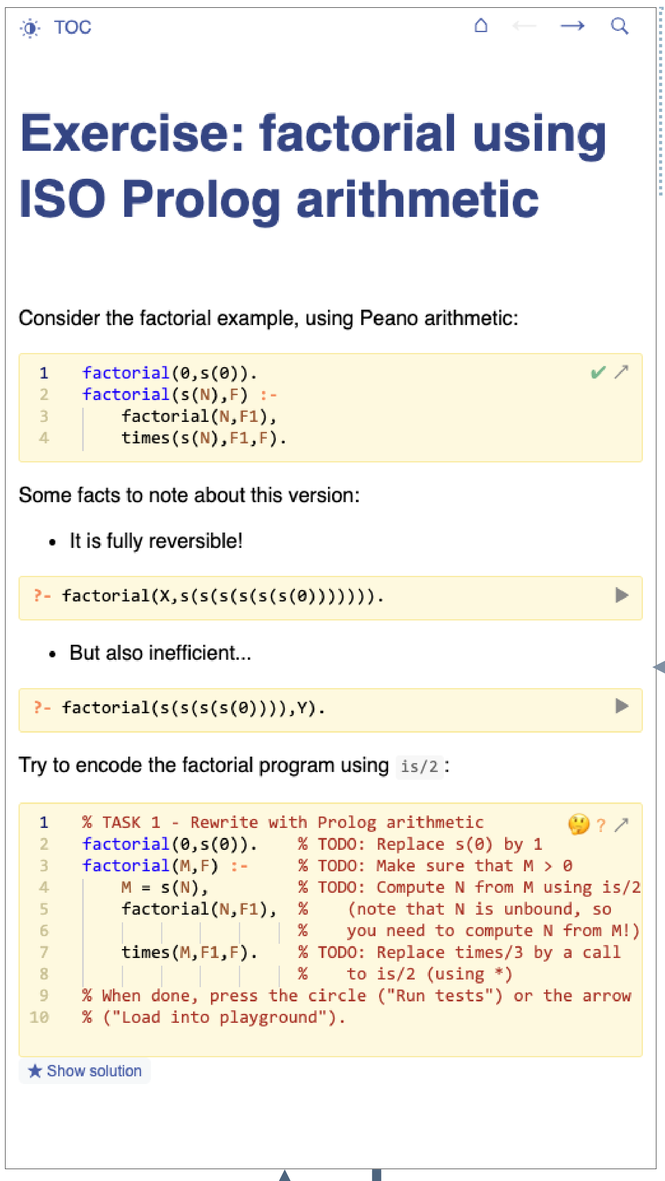


- **Editing** and **execution directly within the browser**: browser- based IDE

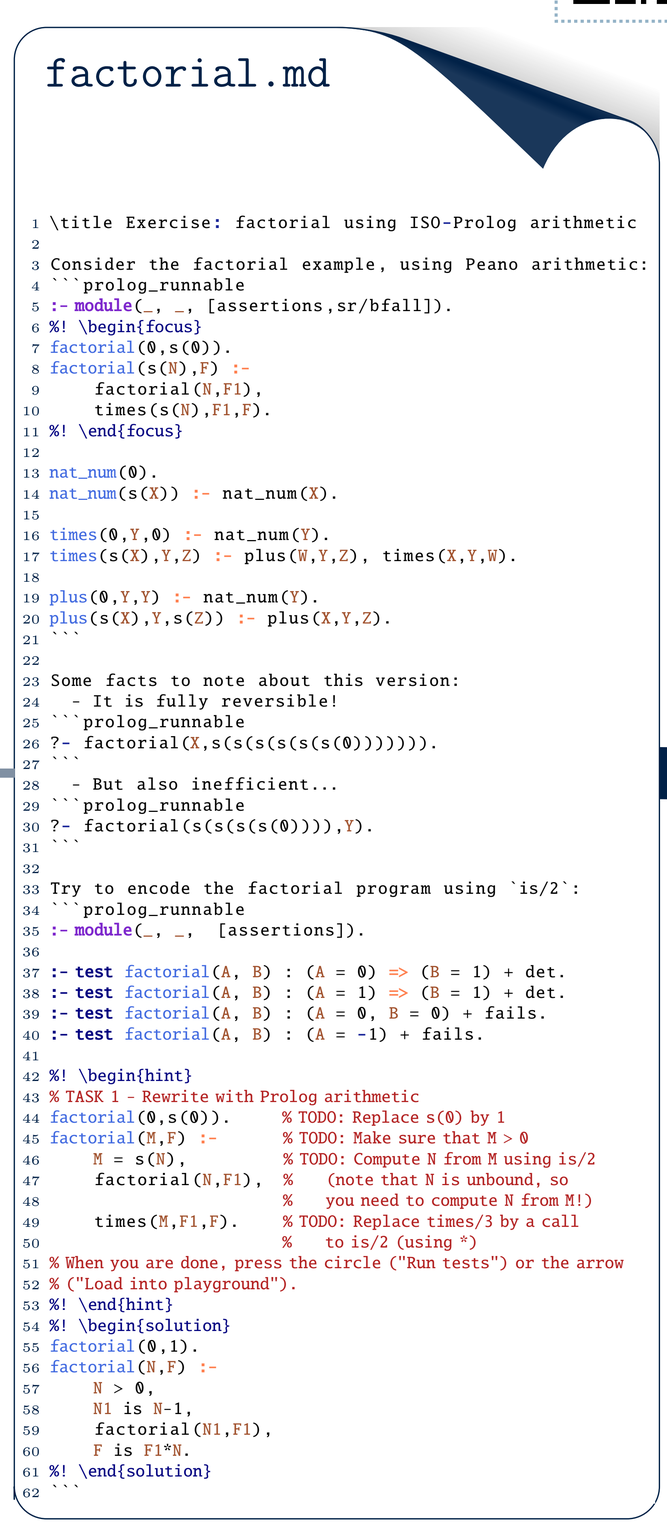
Share! **Easy click-to-play links (embedabble in slides, tutorials, papers, ...)**

- **NEW! Integrated with tooling: documenter** (LPdoc), **verifier** / **tester** / **optimizer** (CiaoPP), ...
- Can be customized for different use cases and applications (e.g., s(CASP) site)

ALD in browser



LPdoc



Active Logic Documents (ALDs)

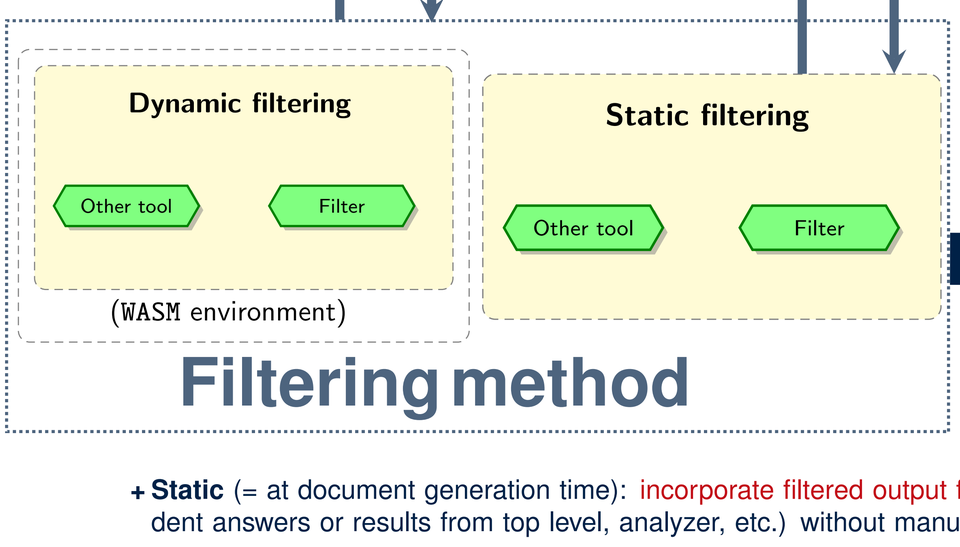
- Interaction with Prolog directly within the documents
- Playground cells ***embedded*** (no server!)
- Incorporates **self-evaluation**
- Develop easily, locally with **any editor**
- Deploy by simply copying a file or directory

A full example!

The Ciao Playground + ALD + Filtering method =

Hybrid Active Logic Document

Filtering method



+ **Static** (= at document generation time): incorporate filtered output from other tools (e.g., expected student answers or results from top level, analyzer, etc.) without manual copy-pasting.

+ **Dynamic** (= when document "runs"): real-time filtering of output from different tools and comparison with user interactions (e.g., student answers).

No server/installation

Entire process runs directly in the user's web browser, eliminating installations and dependencies.

Self-evaluation

The generated documents provide feedback to student based on expected solutions (tests), types, modes, etc.

Customization

Integration with tooling (tests, documentation, verification, ...) provides wide range of customization options.

GET STARTED!

1 Write an ALD

Create the source file(s); include code blocks, narrative text, queries, solutions, tests, etc.

(Just as shown in the above example)

2 Process

```
$ lpdoc file.md
```

3 View

```
$ lpdoc --view file.md
```

4 Share!

```
$ cp -rp file_html ~/public_html
```