

MOTIVATION

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

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

ALD in browser

Exercise: factorial using ISO Prolog arithmetic

Consider the factorial example, using Peano arithmetic:

```
1 factorial(0,s(0)).
2 factorial(s(N),F) :-
3   factorial(N,F1),
4   times(s(N),F1,F).
```

Some facts to note about this version:

- It is fully reversible!

```
?- factorial(X,s(s(s(s(s(s(0))))))).
```

- But also inefficient...

```
?- factorial(s(s(s(s(0))),Y).
```

Try to encode the factorial program using `is/2`:

```
1 % TASK 1 - Rewrite with Prolog arithmetic
2 factorial(0,s(0)). % TODO: Replace s(0) by 1
3 factorial(M,F) :- % TODO: Make sure that M > 0
4   M = s(N), % TODO: Compute N from M using is/2
5   factorial(N,F1), % (note that N is unbound, so
6   % you need to compute N from M!)
7   times(M,F1,F). % TODO: Replace times/3 by a call
8   % to is/2 (using *)
9 % When done, press the circle ("Run tests") or the arrow
10 % ("Load into playground").
```

★ Show solution

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

Prolog playground

```
1 % Write your Prolog code here, e.g.:
2
3 app([],X,X).
4 app([X|L1],L2,[X|L3]) :-
5   app(L1,L2,L3).
6
```

Run tests (C-c u)
Debug (C-c d)
Preview documentation (C-c D)
Analyze and check assertions (C-c a)
Analyze and check assertions (C-c o)
Specialize code (C-c O)

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

- ▶ **Easy click-to-play links** (embeddable 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)

factorial.md

```
1 \title Exercise: factorial using ISO-Prolog arithmetic
2
3 Consider the factorial example, using Peano arithmetic:
4 ```prolog_runnable
5 :- module(., _, [assertions,sr/bfall]).
6 %! \begin{focus}
7 factorial(0,s(0)).
8 factorial(s(N),F) :-
9   factorial(N,F1),
10  times(s(N),F1,F).
11 %! \end{focus}
12
13 nat_num(0).
14 nat_num(s(X)) :- nat_num(X).
15
16 times(0,Y,0) :- nat_num(Y).
17 times(s(X),Y,Z) :- plus(W,Y,Z), times(X,Y,W).
18
19 plus(0,Y,Y) :- nat_num(Y).
20 plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
21 ...
22
23 Some facts to note about this version:
24 - It is fully reversible!
25 ```prolog_runnable
26 ?- factorial(X,s(s(s(s(s(s(0))))))).
27
28 - But also inefficient...
29 ```prolog_runnable
30 ?- factorial(s(s(s(s(0))),Y).
31 ...
32
33 Try to encode the factorial program using `is/2`:
34 ```prolog_runnable
35 :- module(., _, [assertions]).
36
37 :- test factorial(A, B) : (A = 0) => (B = 1) + det.
38 :- test factorial(A, B) : (A = 1) => (B = 1) + det.
39 :- test factorial(A, B) : (A = 0, B = 0) + fails.
40 :- test factorial(A, B) : (A = -1) + fails.
41
42 %! \begin{hint}
43 % TASK 1 - Rewrite with Prolog arithmetic
44 factorial(0,s(0)). % TODO: Replace s(0) by 1
45 factorial(M,F) :- % TODO: Make sure that M > 0
46   M = s(N), % TODO: Compute N from M using is/2
47   factorial(N,F1), % (note that N is unbound, so
48   % you need to compute N from M!)
49   times(M,F1,F). % TODO: Replace times/3 by a call
50   % to is/2 (using *)
51 % When you are done, press the circle ("Run tests") or the arrow
52 % ("Load into playground").
53 %! \end{hint}
54 %! \begin{solution}
55 factorial(0,1).
56 factorial(N,F) :-
57   N > 0,
58   N1 is N-1,
59   factorial(N1,F1),
60   F is F1*N.
61 %! \end{solution}
62 ...
```

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!

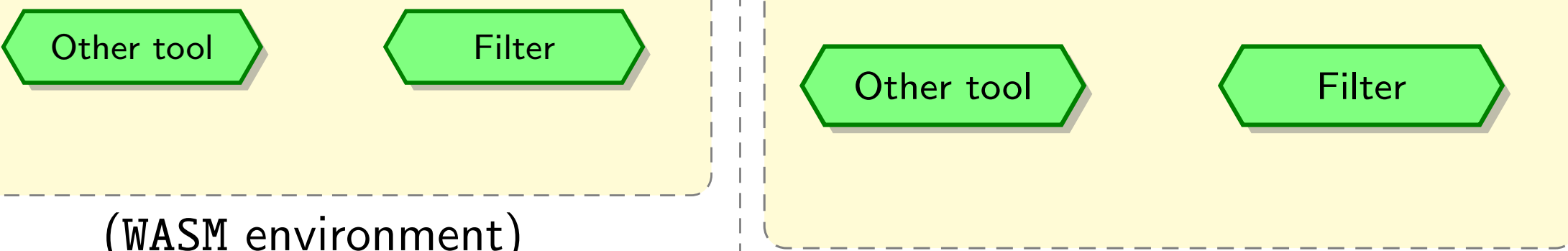


The Ciao Playground + ALD + Filtering method =

Hybrid Active Logic Document

Dynamic filtering

Static filtering



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
```