Ob-lesim: Learning Simulator integration for Org-Mode

Stefano Ghirlanda

May 11, 2023

1 Introduction

Ob-lesim supports editing and running Learning Simulator scripts in Emacs org-mode. In short, it bridges between org-babel (org-mode's code execution system) and lesim-mode, an Emacs major mode for Learning Simulator scripts.

2 Installing

1. Install lesim-mode.
2. Put `ob-lesim.el` in your Emacs load path, for example, `~./emacs.d/elisp`.
3. Add this to your Emacs init file, for example, `~./emacs.d/init.el`:

```
(require 'ob-lesim)
(add-to-list 'org-babel-load-languages '(lesim . t))
```

3 Using

Ob-lesim works mostly behind the scenes. Using Learning Simulator code blocks in org-mode should feel familiar:

- Embed Learning Simulator code in org-mode buffers with:

```
#+begin_src lesim
...
#+end_src
```

- Ob-lesim defines these default header arguments:
  - `:results value`, which is used to insert error messages from the Learning Simulator as `#+RESULTS:#`.
  - `:exports none`, since you rarely want error messages or scripts when exporting.
  - `:noweb yes`, see 4.
  - `:eval no-export` because Learning Simulator scripts can take quite some time to run.

  You can override these defaults globally for an entire org file or locally for each lesim code block as usual in org-mode.

- Other header arguments have no effect unless handled directly by org-mode. For example, you can use `:tangle file.les` to write a script to `file.les`.
- Scripts can be run as code blocks in the org-mode buffer with the usual `C-c C-c` key binding.
- Scripts can also be run from the edit buffer, which has major mode `lesim-mode`, with the regular lesim-mode key binding (by default, `C-c C-r`).
- Lesim-mode features are partly available in the org-mode buffer (for example, `TAB` moves by field in phase blocks), and fully available in org-mode edit buffers.

4 Sharing code across scripts

Ob-lesim supports noweb references to share code across lesim code blocks. For example, this is how you can make sure that two different scripts use the same parameter values:

```
This is a lesim parameter block:
#+name: common-parameters:
#+begin_src lesim
  # lesim parameter block here
#+end_src

This is the script 1:
#+begin_src lesim
  <<common-parameters>>

  # script 1 here here
#+end_src

Ths is script 2:
#+begin_src lesim
  <<common-parameters>>

  # script 2 here here
#+end_src
```

You can also share phase blocks and other script snippets if useful. Thanks to awesome engineering, noweb references are automatically resolved when tangling, so that you end up with complete scripts.

When editing a lesim block, any noweb references are expanded so that you can see the whole script and syntax checking by lesim-mode works. When you close the edit buffer, noweb references are collasped again.

5 Bugs and planned features

Bugs and planned features are tracked as issues on Github.

6 Internals

These are mostly notes to myself. Ob-lesim has two ways of running scripts:

1. Evaluate a code block in an org-mode buffer. This works by saving `full-body` to temporary file (noweb references are expanded by org-mode), then using lesim-error and lesim-run on the file.

2. Evaluate an org-src edit buffer with `C-c C-r` (`lesim-mode-run-key`). This works by saving the buffer to a temporary file, then using lesim-error and lesim-run.