

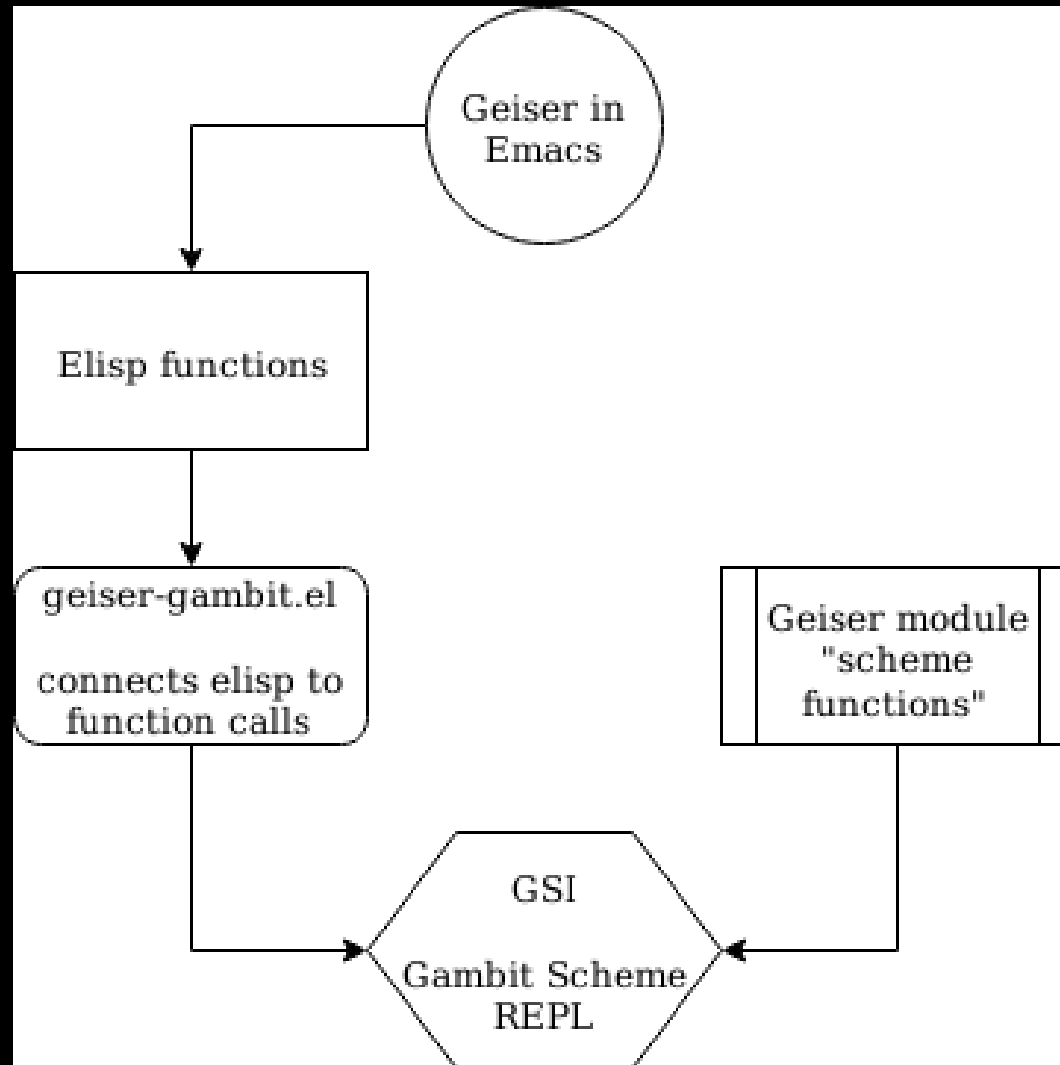


# Geiser with Gambit











Mathieu Perron

# WHAT IS GEISER ?

Geiser is an Emacs  
environment to hack and have  
fun in Scheme.  
( inspired by : Common Lisp's  
Slime, Factor's FUEL, Squeak or  
Emacs itself)



# What can you do with Geiser?

-  Form evaluation
-  Macro expansion
-  File/module loading and/or compilation
-  Identify completion
-  Autodoc: the echo area shows information about the signature of the procedure/macro around point automatically.
-  Access to documentation (including docstrings when the implementation provides it).
-  Listings of callers/callees of procedures.
-  Rudimentary support for debugging (when the REPL provides a debugger) and error navigation.
-  Support for multiple, simultaneous REPLs.
-  Remote connection to REPL

# Functionalities helper

- Handy until you've memorized Geiser commands ( learning device )
- `Ctrl-h m` :: another option

Geiser		In/Out	Signals	Help
Complete symbol		M-TAB		
Complete module name		M-`		
Edit symbol		M-.		
Switch to module...		C-c RET		
Import module...		C-c TAB		
Previous matching input		M-p		
Next matching input		M-n		
Previous input		C-c M-p		
Next input		C-c M-n		
<input type="checkbox"/>	Autodoc mode	C-c C-d C-a		
	Symbol documentation	C-c C-d C-d		
	Module documentation	C-c C-d m		
Kill Scheme interpreter		C-c C-q		
Restart		C-c C-z		
Revive REPL		C-c C-k		
REPL options				
Run		▶		
Switch to		▶		
Customize		▶		

```
;; This buffer is for text that is not saved, and for Lisp evaluation.  
;; To create a file, visit it with C-x C-f and enter text in its buffer.
```

# Form evaluation

- You can open a scheme file and evaluate every sexpr and the entire file directly from Emacs !

C-M-x	<code>geiser-eval-definition</code>	Eval definition around point
C-c C-c	<code>geiser-eval-definition</code>	Eval definition around point
C-c M-e	<code>geiser-eval-definition-and-go</code>	Eval definition around point and switch to REPL
C-c M-c	<code>geiser-eval-definition-and-go</code>	Eval definition around point and switch to REPL
C-x C-e	<code>geiser-eval-last-sexp</code>	Eval sexp before point
C-c C-r	<code>geiser-eval-region</code>	Eval region
C-c M-r	<code>geiser-eval-region-and-go</code>	Eval region and switch to REPL
C-c C-b	<code>geiser-eval-buffer</code>	Eval buffer
C-c M-b	<code>geiser-eval-buffer-and-go</code>	Eval buffer and switch to REPL

C-C W-P `geiser-eval-buffer-and-go` Eval buffer and switch to REPL

C-C C-P `geiser-eval-region` Eval region

C-C M-P `geiser-eval-region-and-go` Eval region and switch to REPL

**A lot of nice details**

**Code  
indentation in  
REPL**

**Company-mode  
compatibility**

**Macro-  
expansion**

**Remote REPL  
connection**

# Fully customizable

Indentation in terminal vs indentation using Geiser

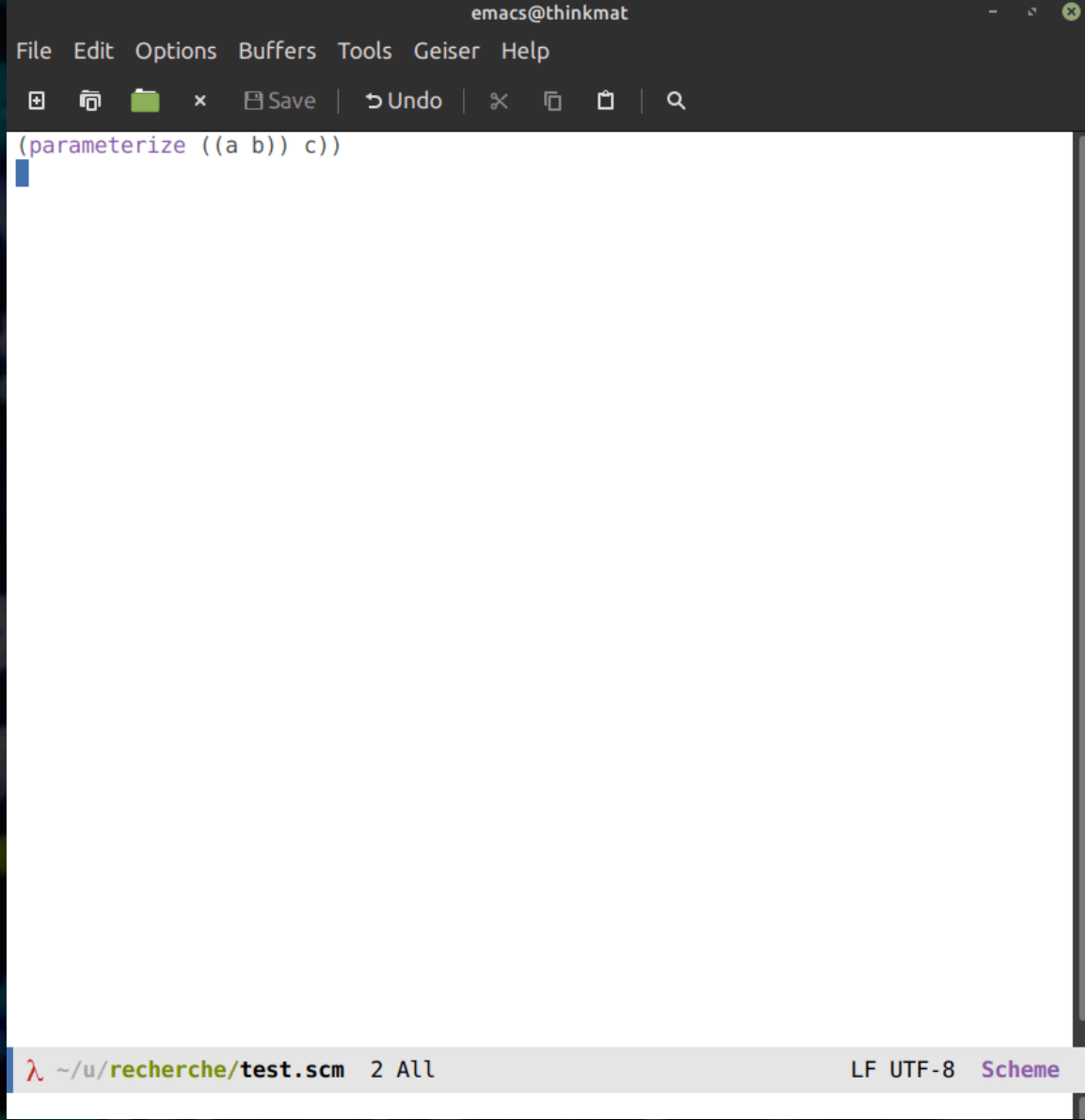
```
Gambit v4.9.3
> █
```

```
File Edit Options Buffers Tools Geiser In/Out Signa
[Icons] Save Undo [Icons]
> (d

>_ ÷ * Gambit REPL * 1 All LF UTF-8 REPL
```



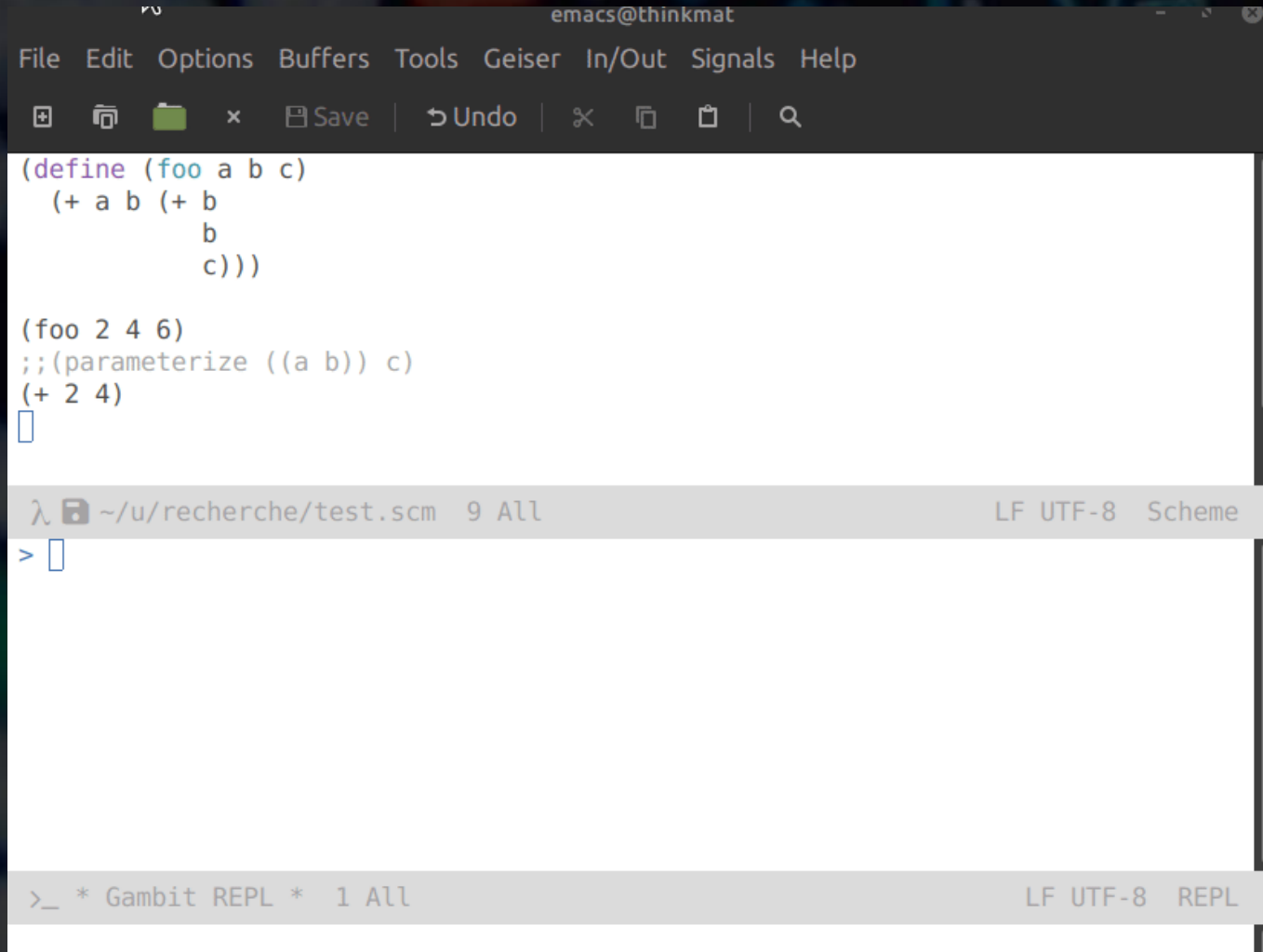
# Macro-expand example



The image shows a screenshot of an Emacs editor window. The title bar at the top reads "emacs@thinkmat". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Geiser", and "Help". The toolbar contains icons for new file, open file, save, close, save all, undo, redo, copy, paste, and search. The main text area contains the Scheme code snippet: `(parameterize ((a b)) c)`. The cursor is positioned at the end of the first line. The status bar at the bottom shows the file path `λ ~/u/recherche/test.scm`, the line and column number `2 All`, the encoding `LF UTF-8`, and the dialect `Scheme`.

```
emacs@thinkmat
File Edit Options Buffers Tools Geiser Help
+ [new] [open] [save] [close] | Save | Undo | Redo | Copy | Paste | Search
(parameterize ((a b)) c)
λ ~/u/recherche/test.scm 2 All LF UTF-8 Scheme
```

# Buffer evaluation



The image shows a screenshot of the Emacs editor window titled "emacs@thinkmat". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Geiser", "In/Out", "Signals", and "Help". The toolbar contains icons for new file, open file, save, undo, redo, copy, paste, and search. The main editing area contains the following Scheme code:

```
(define (foo a b c)
  (+ a b (+ b
            b
            c)))

(foo 2 4 6)
;;(parameterize ((a b)) c)
(+ 2 4)

```

Below the code is a status bar showing the current buffer as "λ ~/u/recherche/test.scm 9 All" with encoding "LF UTF-8" and mode "Scheme". Below this is a REPL window with the prompt "> " and a cursor.

At the bottom of the image, there is another status bar for the REPL window showing ">\_ \* Gambit REPL \* 1 All" with encoding "LF UTF-8" and mode "REPL".

Autodoc

# Remote connect

To connect to a distant REPL

# FIRST OPEN A REPL USING THE GEISER MODULE

```
mathieu@thinkmat: ~  
File Edit View Search Terminal Help  
mathieu@thinkmat:~$ gsi -:dar$ gambit/geiser  
-█
```

File Edit Options Buffers Tools Lisp-Interaction Help

⊞ ⊞ ⊞ × Save | Undo | ✂ ⊞ ⊞ | 🔍

;; This buffer is for text that is not saved, and for Lisp evaluation.  
;; To create a file, visit it with C-x C-f and enter text in its buffer.



🔗 \*scratch\* 4 All

LF UTF-8 Lisp Interaction

mathieu@thinkmat: ~

File Edit View Search Terminal Help

mathieu@thinkmat:~\$ gsi -:dar\$ gambit/geiser

->

# Coming soon !



Current thread identification in geiser REPL for multiple connections.



Debugging system for multithreading



Remote connection request to Geiser from the REPL



You can contribute too ! Suggestions of fonctionnalities are always welcome too.

Loading...



# How to configure Geiser with Gambit for full functionalities ?

Configure  
using `-enable-  
rtlib-debug`

Be sure to add  
gsi to your  
terminal path

Package-install  
geiser (from  
Melpa)



Missing features:

- External documentation
- Better debugging system
- Modules handling directly from geiser
- Compilation

Still very usable.

LETS DO A DEMO!



**QUESTIONS?**