Asynchronous Distributed Multiplexed REPLs

DC-RES
Albert Zak
2023-11-03

REPL

DC-RES
Albert Zak
2023-11-03

REPL

DC-RES
Albert Zak
2023-11-03

```
python /...  ⌥⌘1
>>> 1 + 1
2
>>>
```

read

REPL

eval

print

DC-RES
Albert Zak
2023-11-03

```
python /...  ⌥⌘1
>>> 1 + 1
2
>>>
```

read

REPL

eval

print

DC-RES
Albert Zak
2023-11-03

```
python /...  ⌥⌘1
>>> 1 + 1
2
>>>
```

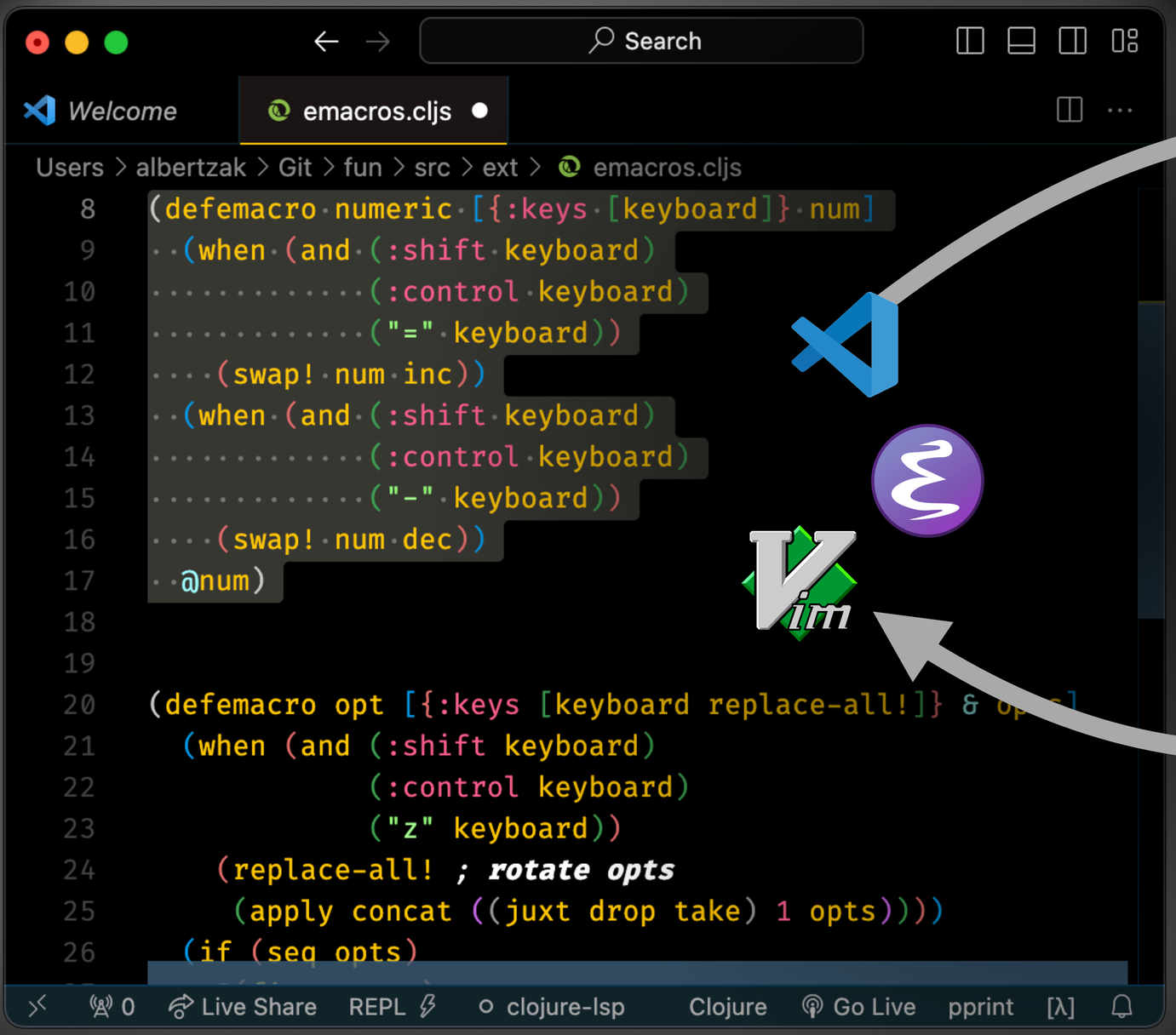
read

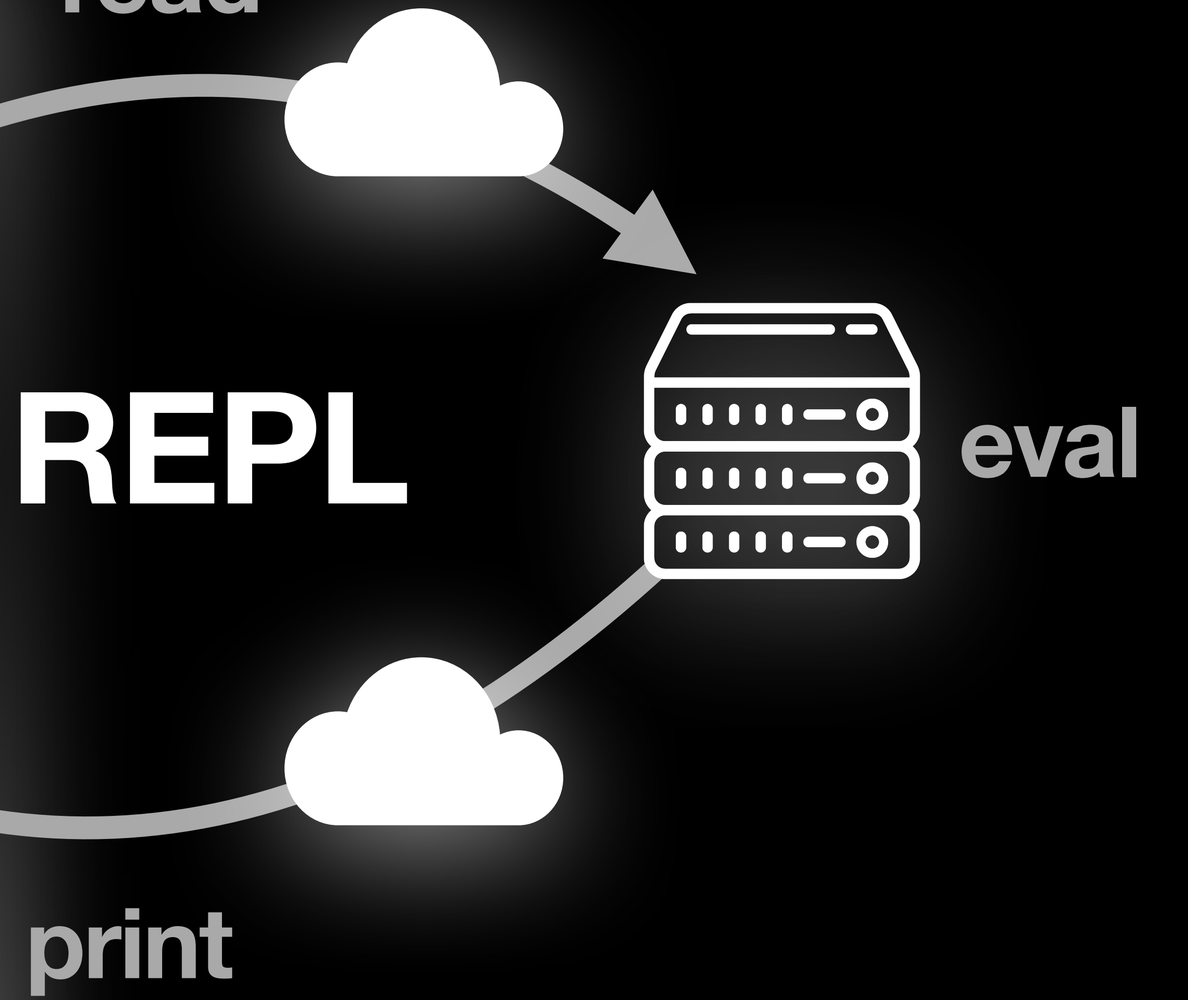
REPL

eval

print

DC-RES
Albert Zak
2023-11-03



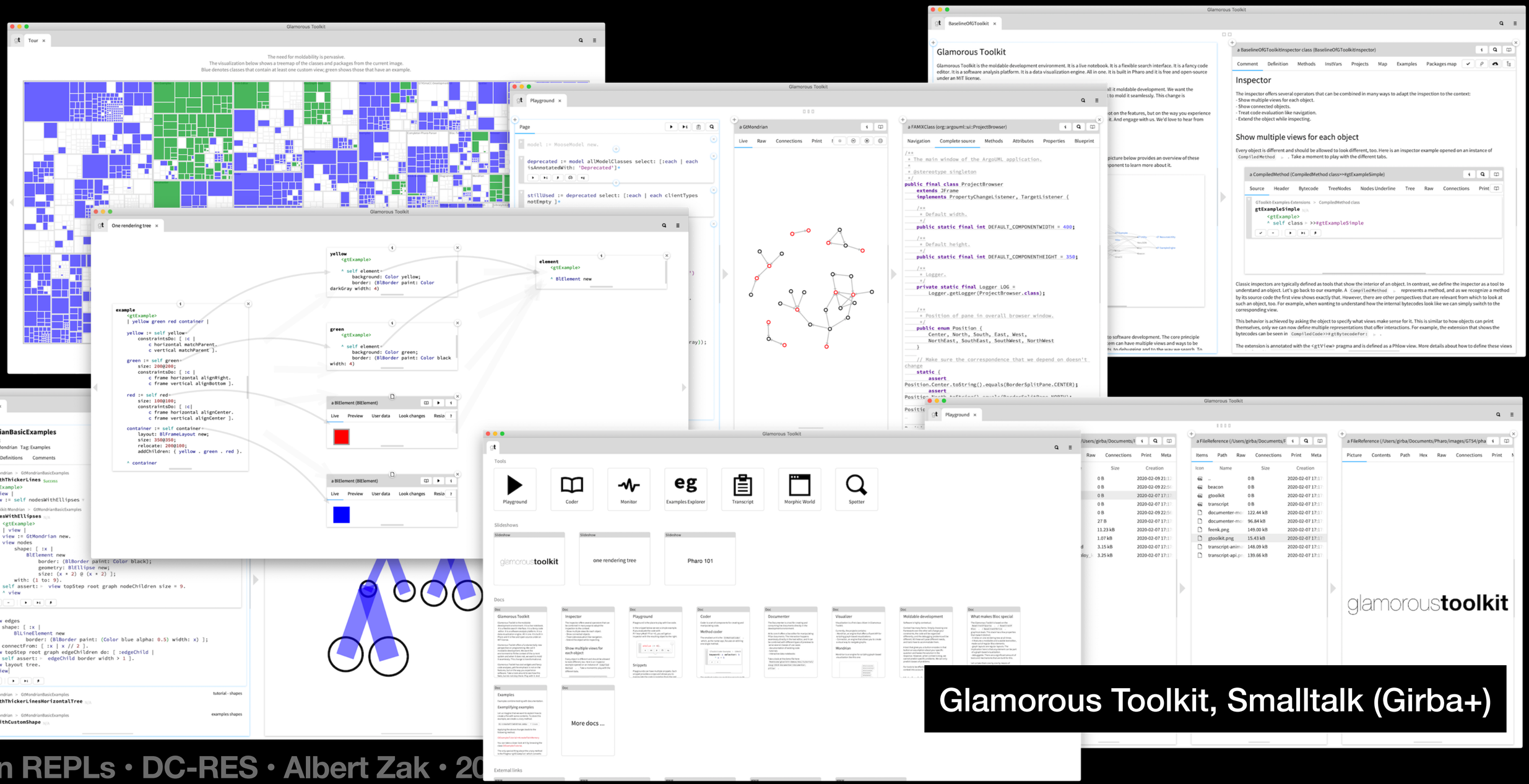


DC-RES
Albert Zak
2023-11-03

state of the art

read-eval-print-loop
sync request/response
1:1
ad-hoc editor integration
exception handling

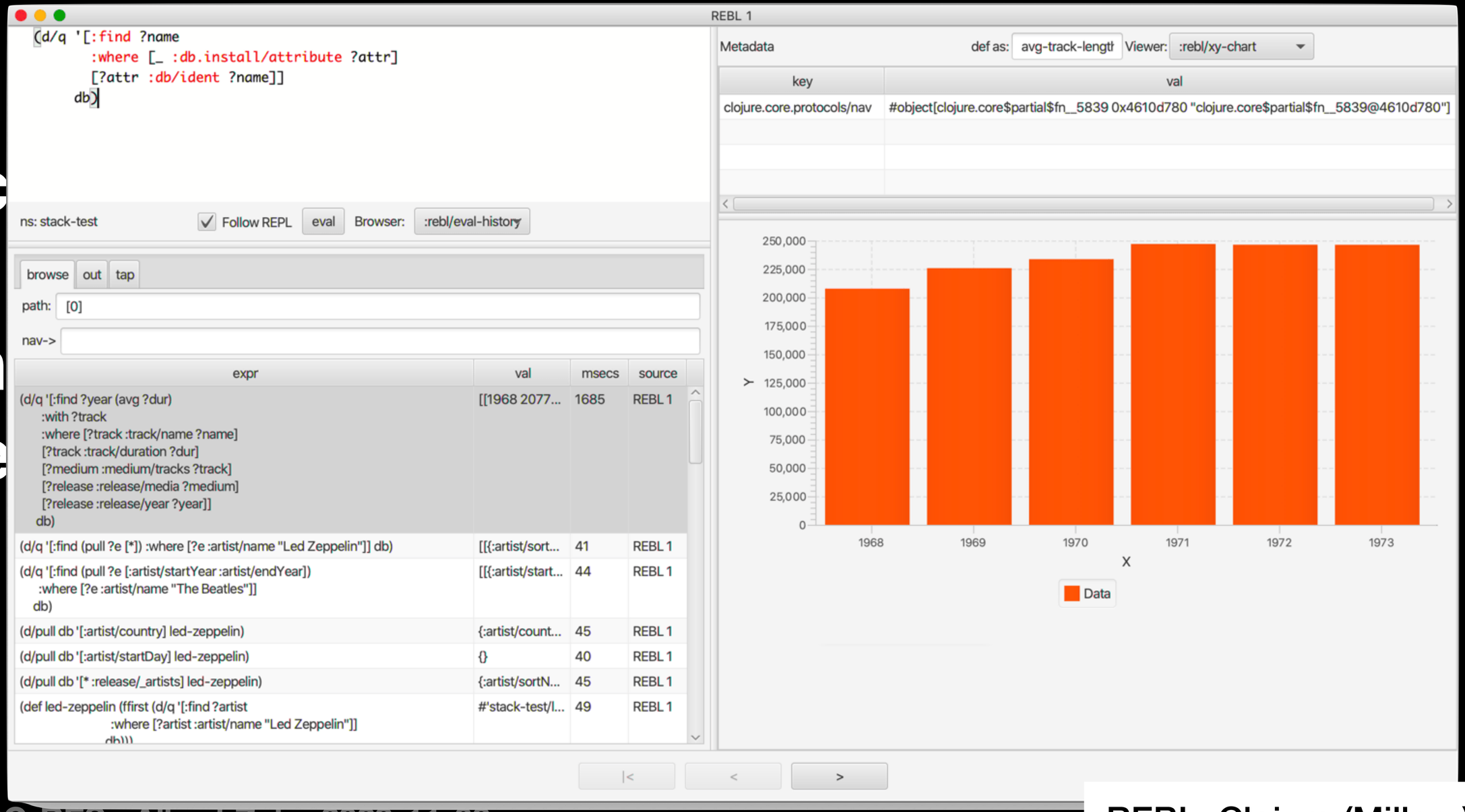
state of the art



On REPLs • DC-RES • Albert Zak • 20

state of the art

read
sync
1:1
ad-h
exce



REBL, Clojure (Miller+)

state of the art

```
clj::fresh-reagent.handler=>
(defn leet->1337 [s]
  (-> s
      (clojure.string/replace #"l", "1")
      (clojure.string/replace #"e", "3") => "133t"
      (clojure.string/replace #"t", "7")))
#'fresh-reagent.handler/leet->1337
clj::fresh-reagent.handler=>
(leet->1337 "leet")
```

Calva, Clojure (Strömberg+)

```
lem --eval "(lem-lisp-mode:start-li
CL-USER> (defun hello (name)
           (print name))
HELLO
CL-USER> (hello)

; No value
CL-USER> (hello)
```

```
 * *lisp-repl* (lisp-repl listener) (8, 0) [CL-USER sbcl:SELF]                All
invalid number of arguments: 0
   [Condition of type SB-INT:SIMPLE-PROGRAM-ERROR]

Restarts:
 0: [REPLACE-FUNCTION] Call a different function with the same arguments
 1: [CALL-FORM] Call a different form
 2: [RETRY] Retry SLIME REPL evaluation request.
 3: [ABORT] Return to sldb level 1.
 4: [REPLACE-FUNCTION] Call a different function with the same arguments
 5: [CALL-FORM] Call a different form
 6: [RETRY] Retry SLIME REPL evaluation request.
 7: [*ABORT] Return to SLIME's top level.
 8: [ABORT] abort thread (#<THREAD "repl-thread" RUNNING {1006710413}>)

Backtrace:
  0: (HELLO) [external]
  1: (SB-INT:SIMPLE-EVAL-IN-LEXENV (HELLO)
  2: (EVAL (HELLO))
 --more--

 % *sldb 1* (sldb) (1, 30)                                                    All
```

SLIME, Common Lisp (Eller+)

state of the art

read-eval-print
sync req/res
1:1
ad-hoc editor
exceptions

missing

pull, as-you-type
async streams
1:n
explicit textual state
fault handling
stable time basis

missing

pull, as-you-type
async streams
1:n
explicit textual state
fault handling
stable time basis

~~side effect safety~~ ~~what-if overlay~~
~~(ocap) security~~ ~~data viz~~ ~~transactions~~

tickets

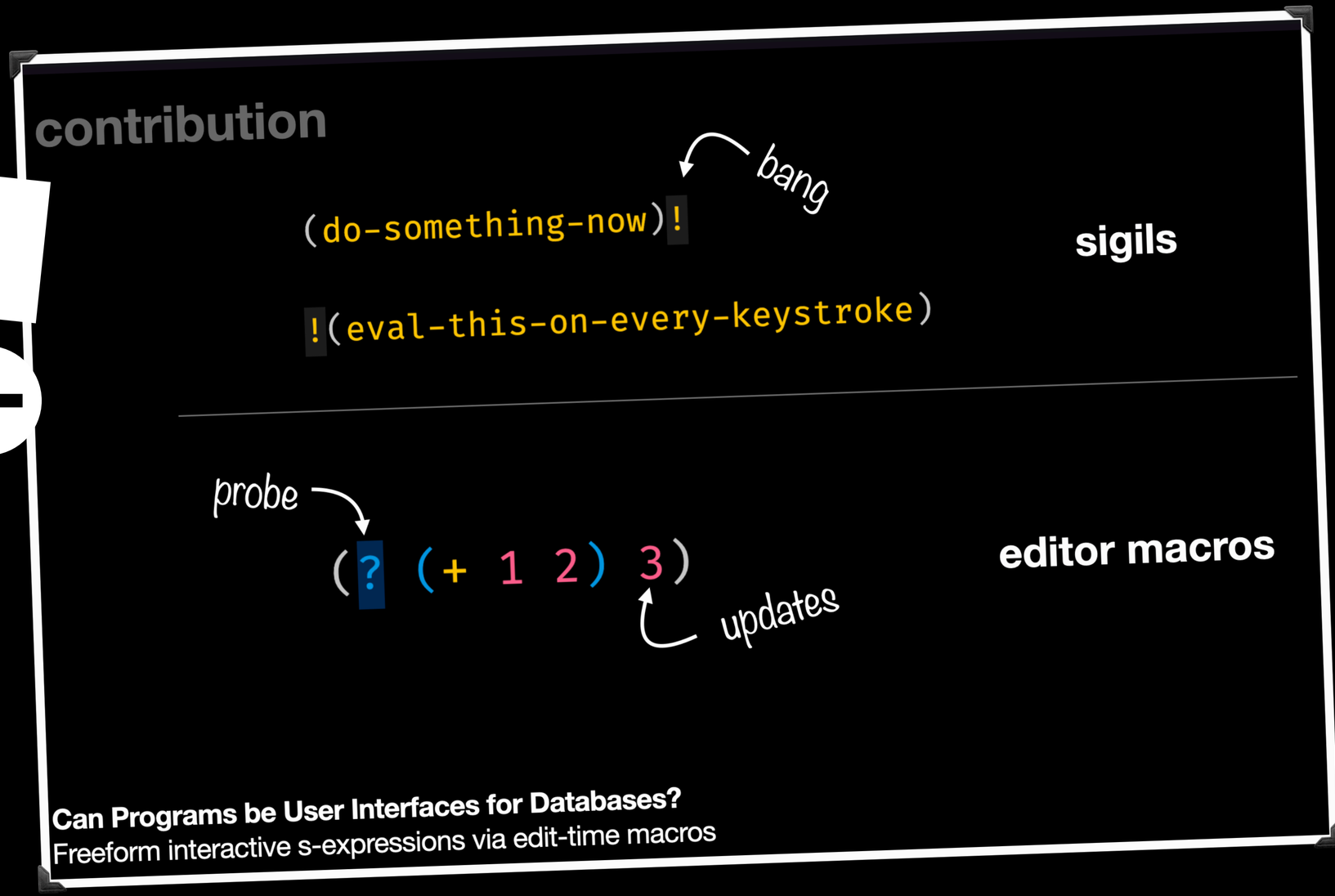
context

pull, as-you-type
async streams
1:n
explicit textual state
fault handling
stable time basis

tickets

+

context



contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

contribution

pull, as-you-type

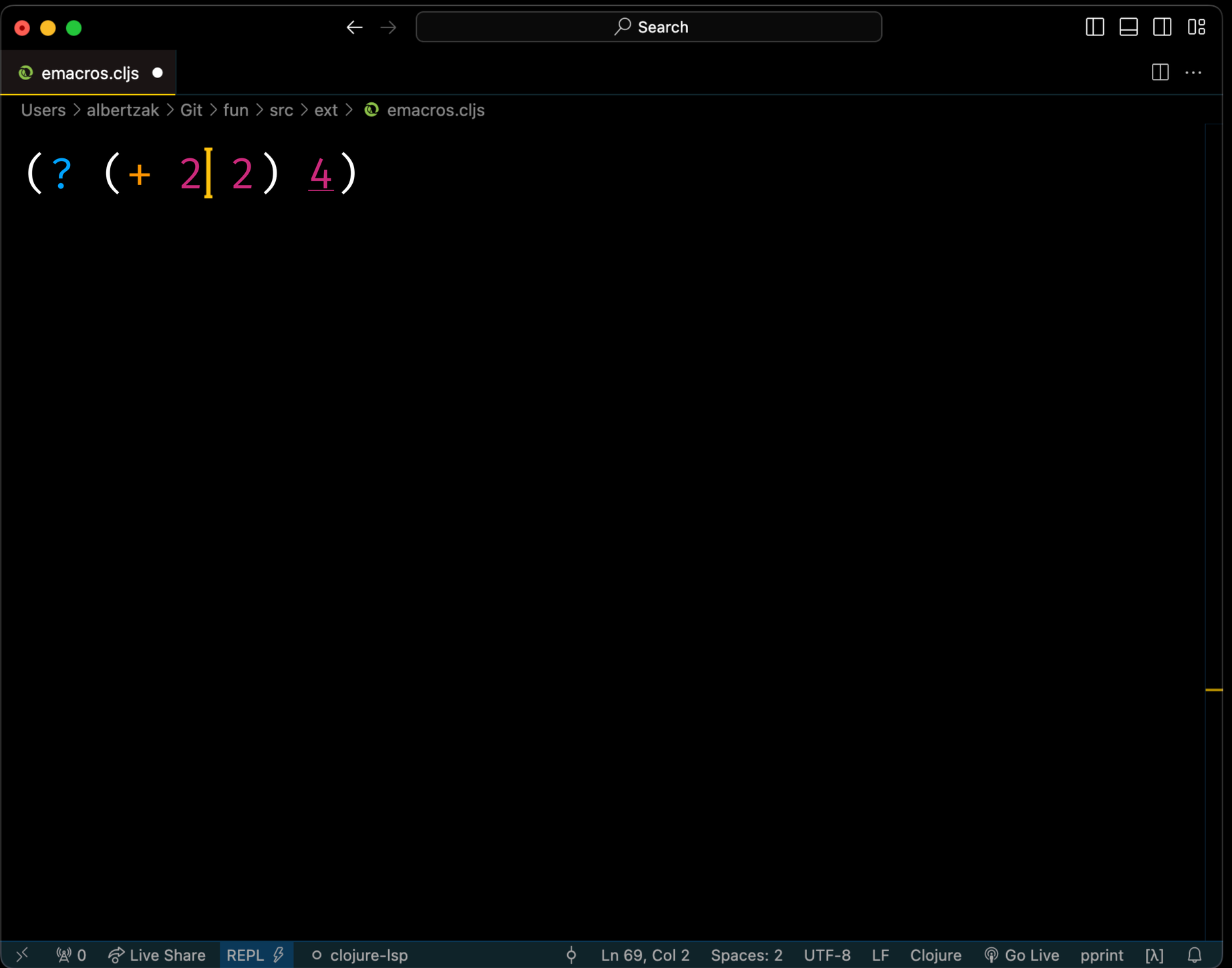
async streams

1:n

explicit state

fault handling

stable time basis



contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

contribution

pull, as-you-type

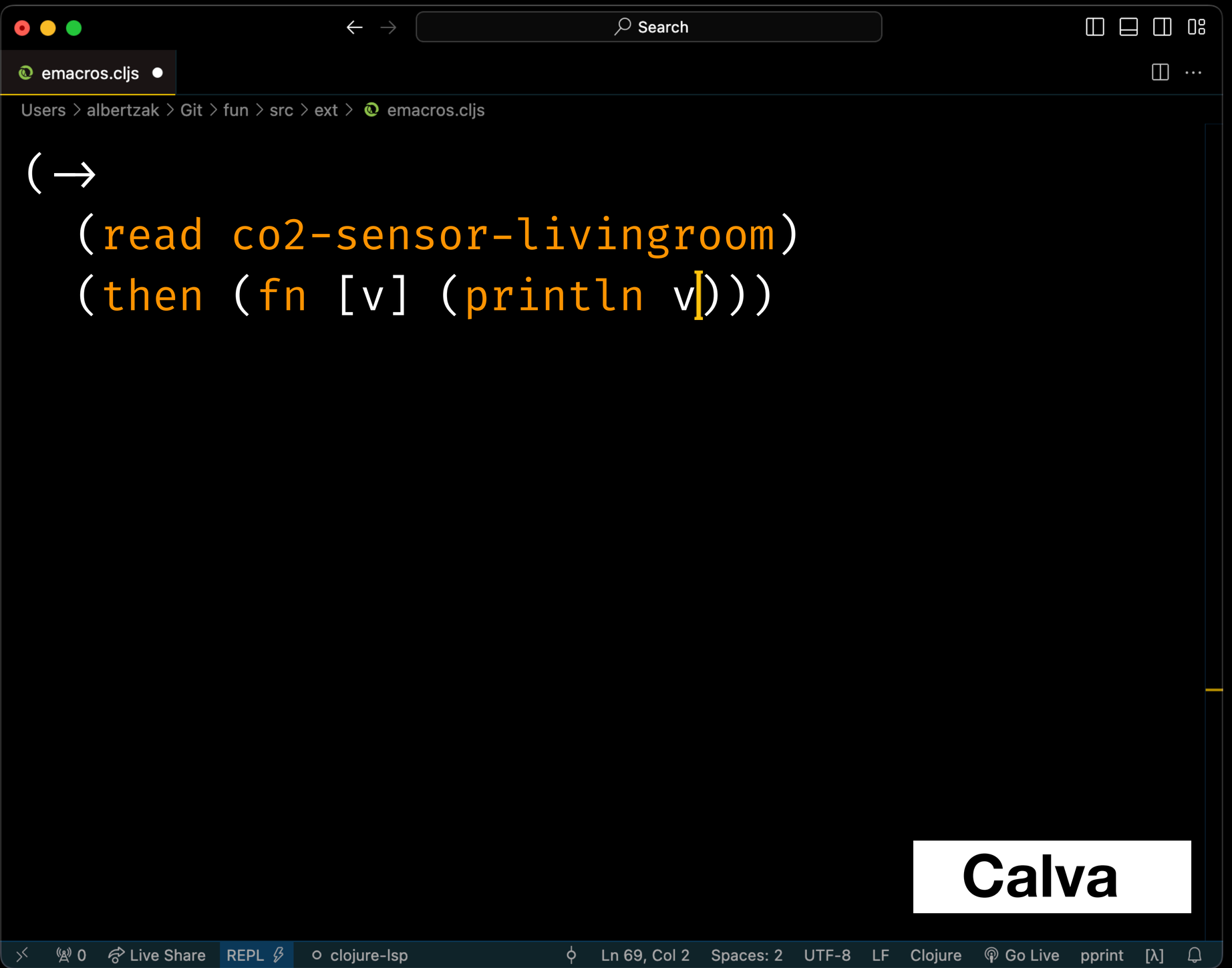
async streams

1:n

explicit state

fault handling

stable time basis



contribution

pull, as-you-type

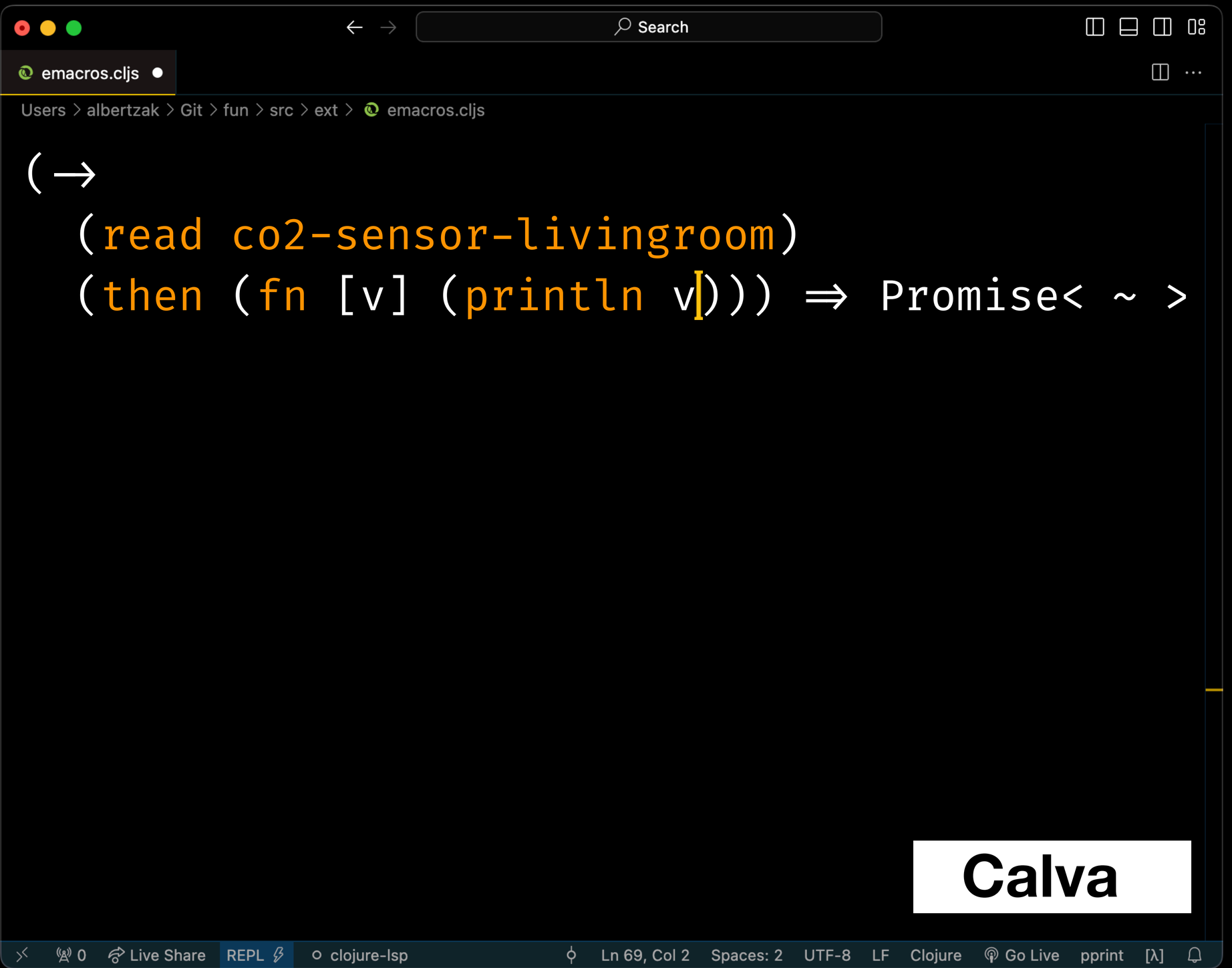
async streams

1:n

explicit state

fault handling

stable time basis



contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

Search

emacros.cljs

Users > albertzak > Git > fun > src > ext > emacros.cljs

```
(→
  (read co2-sensor-livingroom)
  (then (fn [v] (println v))))
```
⇒ Promise< ~ >

fish /Users/albertzak ⌥⌘1

```
⨯> ~ ssh co2-sensor-node
⨯> ~ sudo docker logs -f fdcdab7ca6a7
Password:
512.31
```

Calva

0 Live Share REPL clojure-lsp Ln 69, Col 2 Spaces: 2 UTF-8 LF Clojure Go Live pprint [λ]

contribution

pull, as-you-type

async streams

1:n

explicit state

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

tickets

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

tickets

contribution

pull, as-you-type

async streams

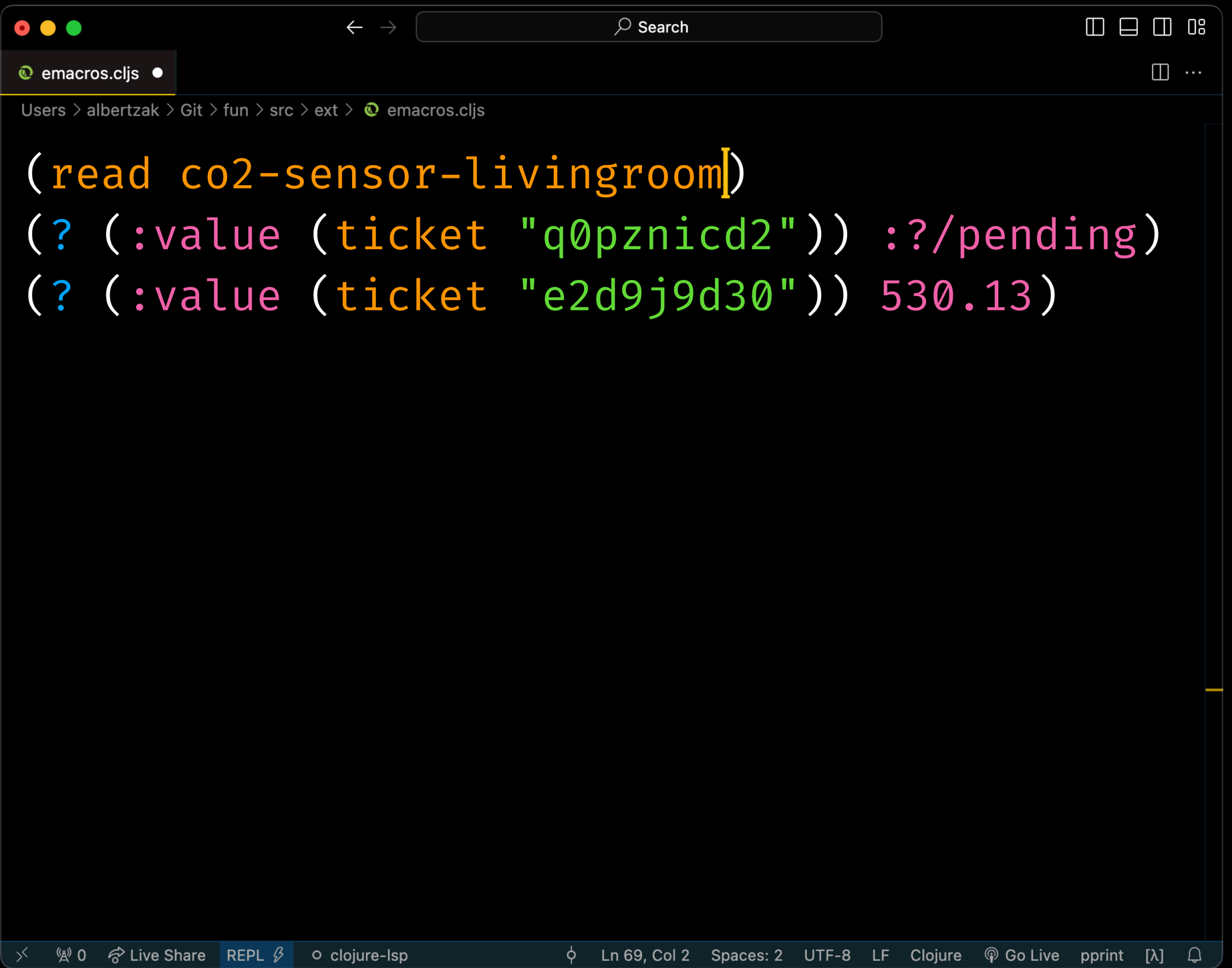
1:n

explicit state

fault handling

stable time basis

tickets



contribution

pull, as-you-type

async streams

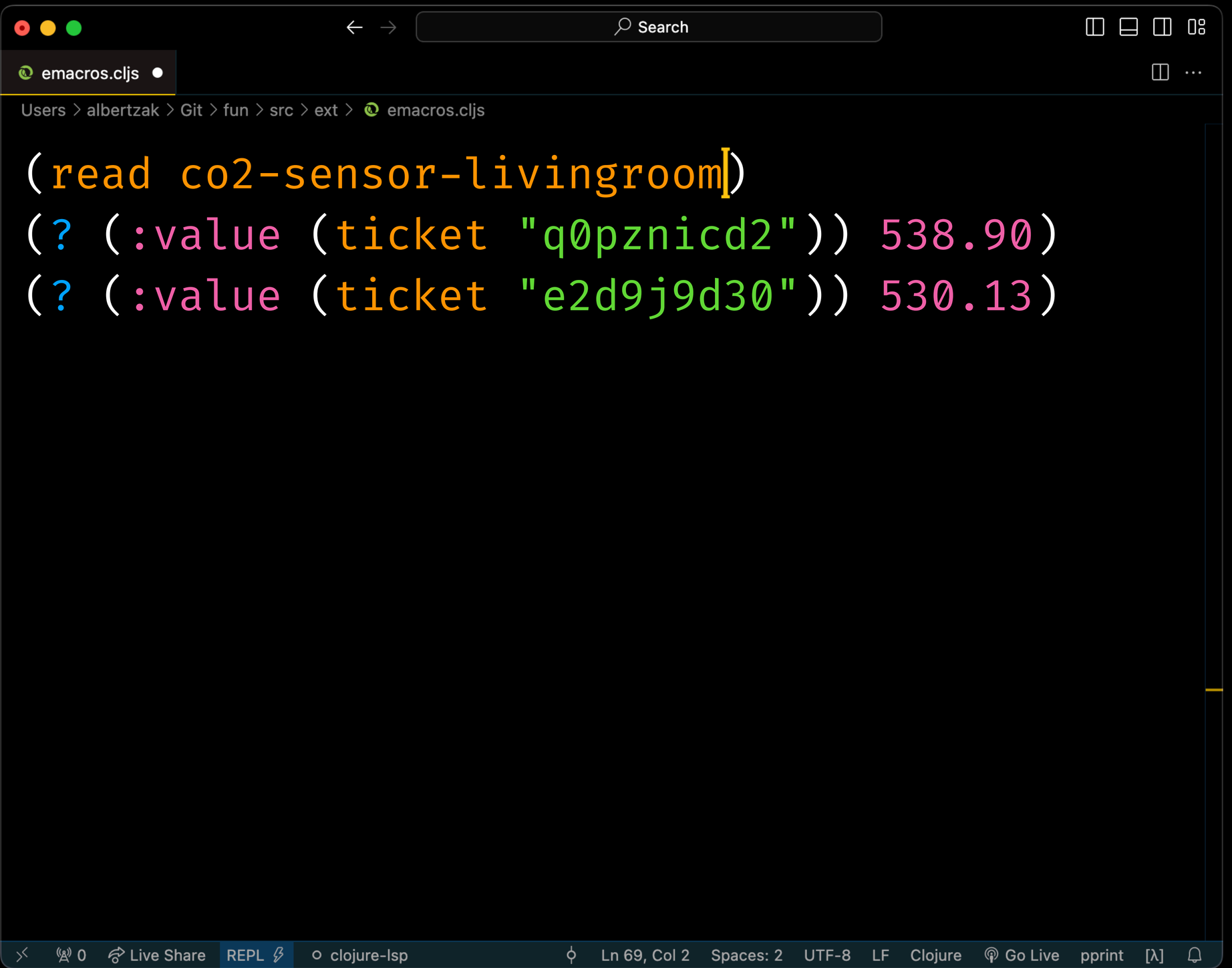
1:n

explicit state

fault handling

stable time basis

tickets



contribution

pull, as-you-type

async streams

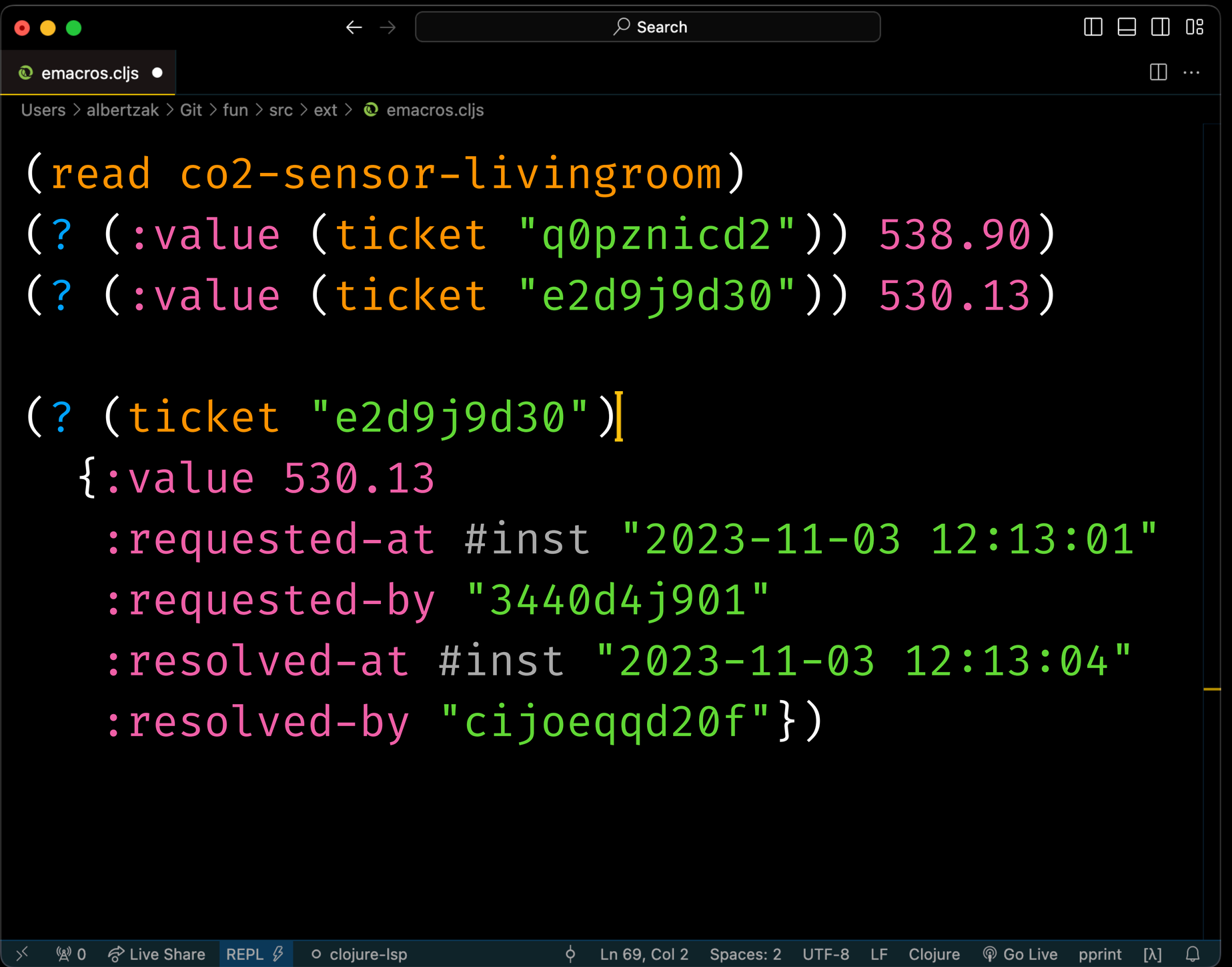
1:n

explicit state

fault handling

stable time basis

tickets



contribution

pull, as-you-type

async streams

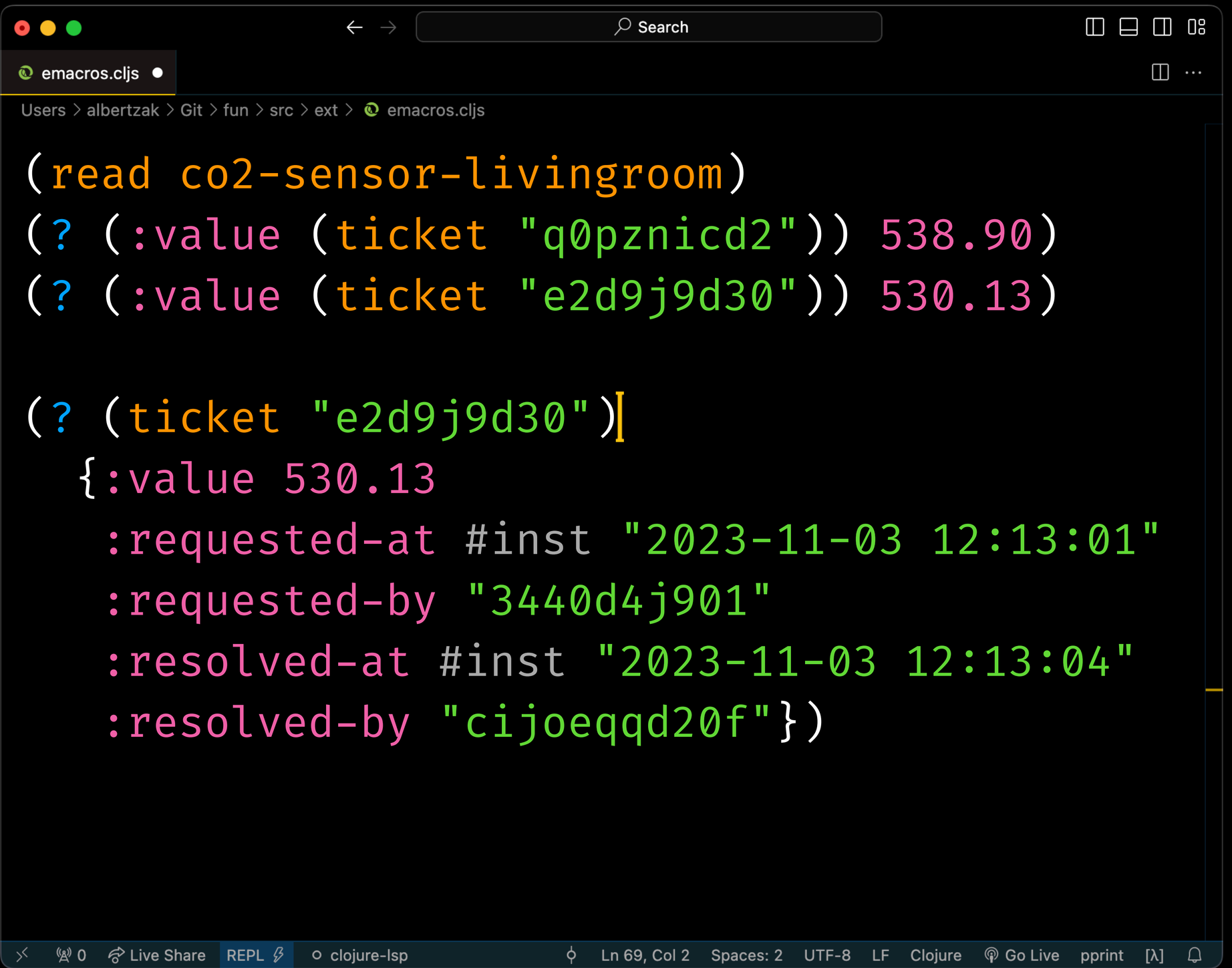
1:n

explicit state

fault handling

stable time basis

tickets



contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

tickets

contribution

pull, as-you-type

async streams

1:n

explicit state

fault handling

stable time basis

tickets

context

contribution

pull, as-you-type

async streams

1:n

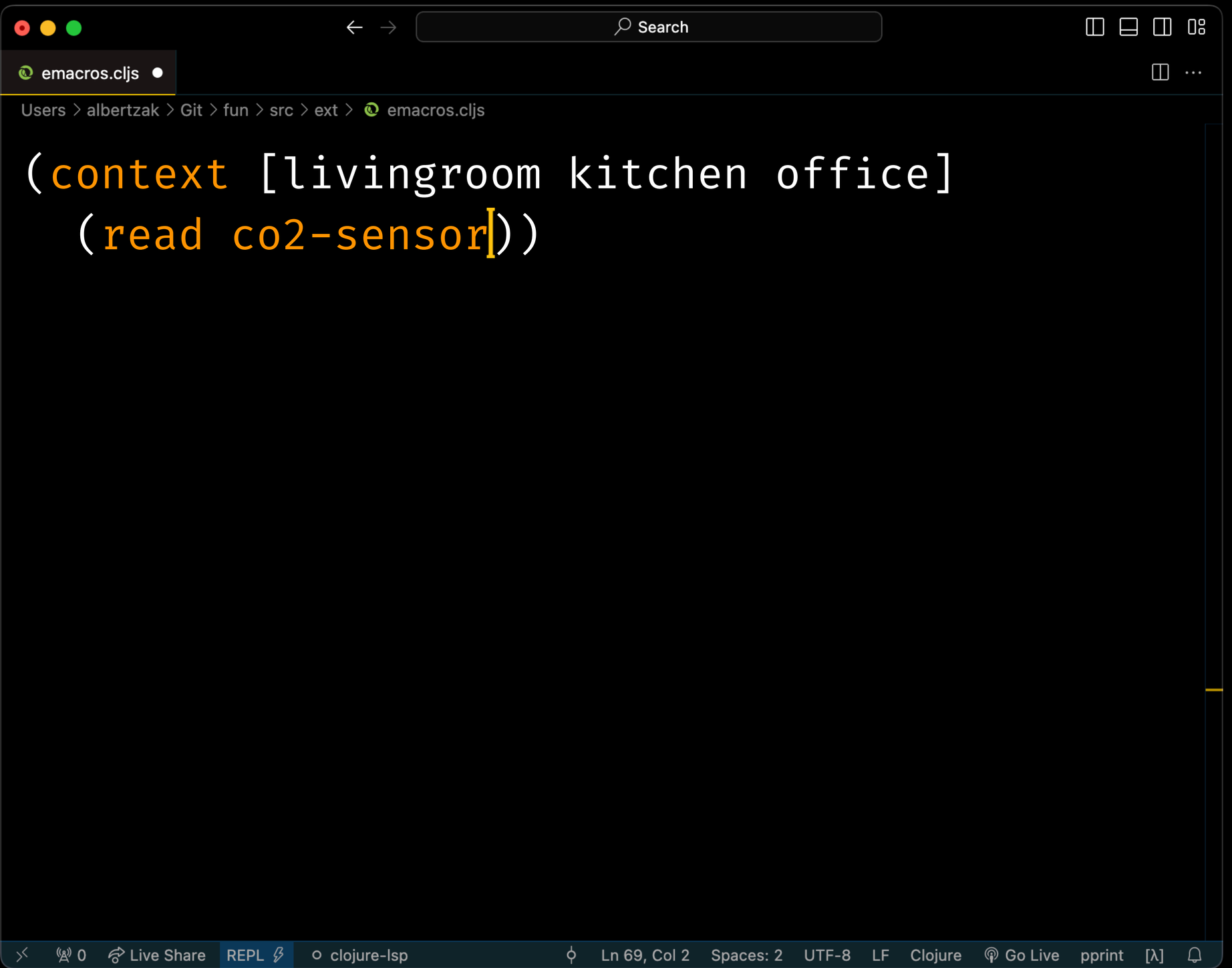
explicit state

fault handling

stable time basis

tickets

context

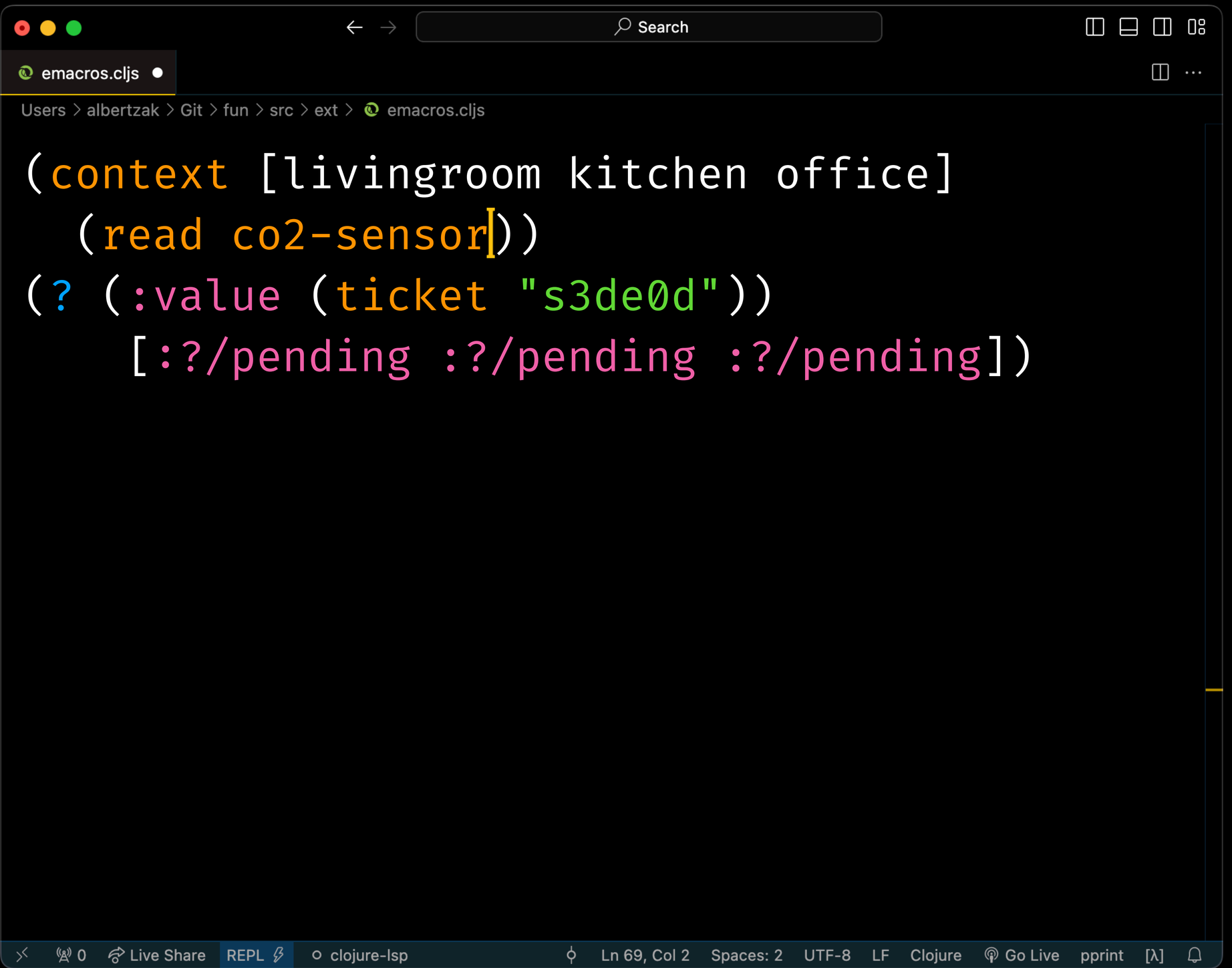


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

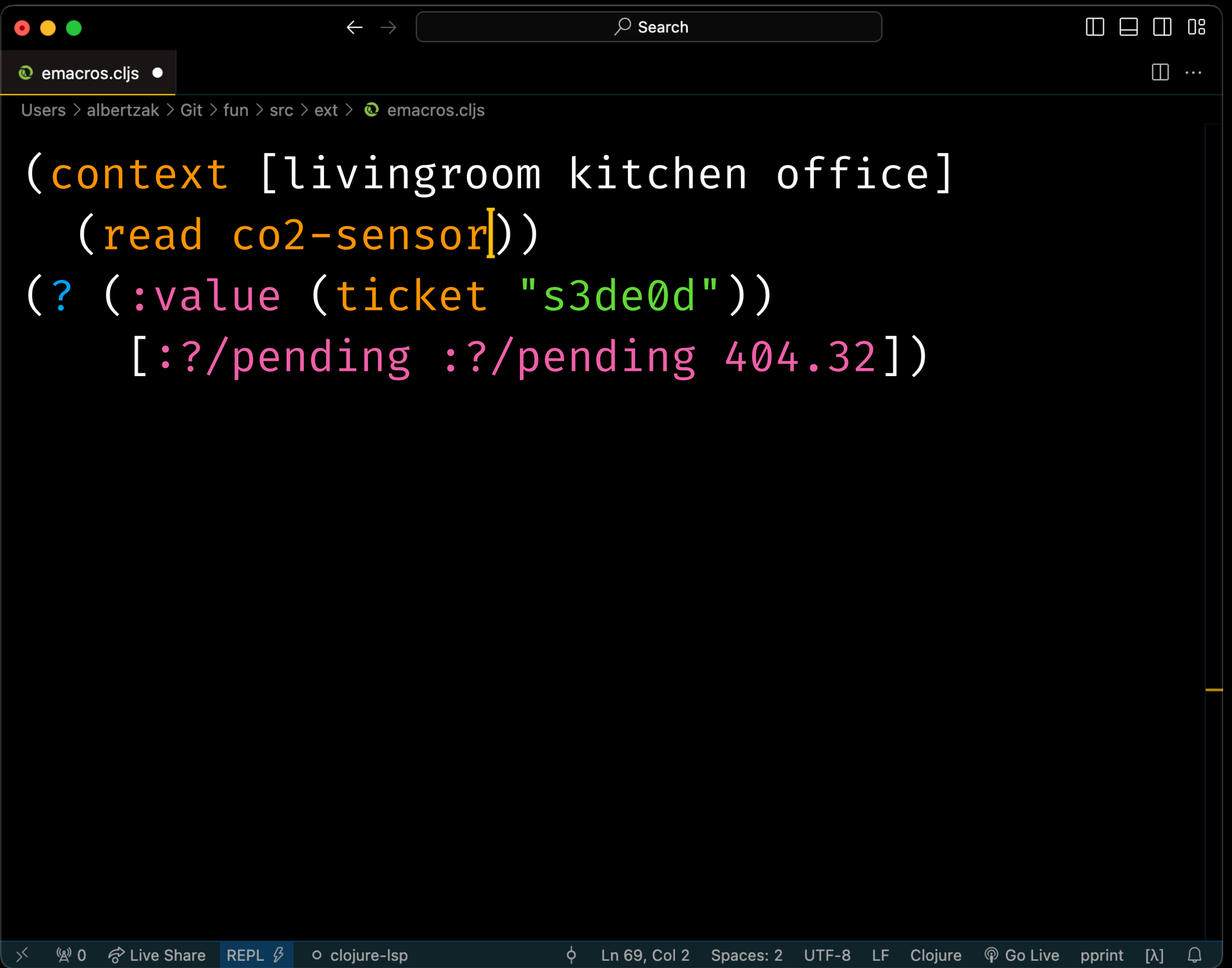


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

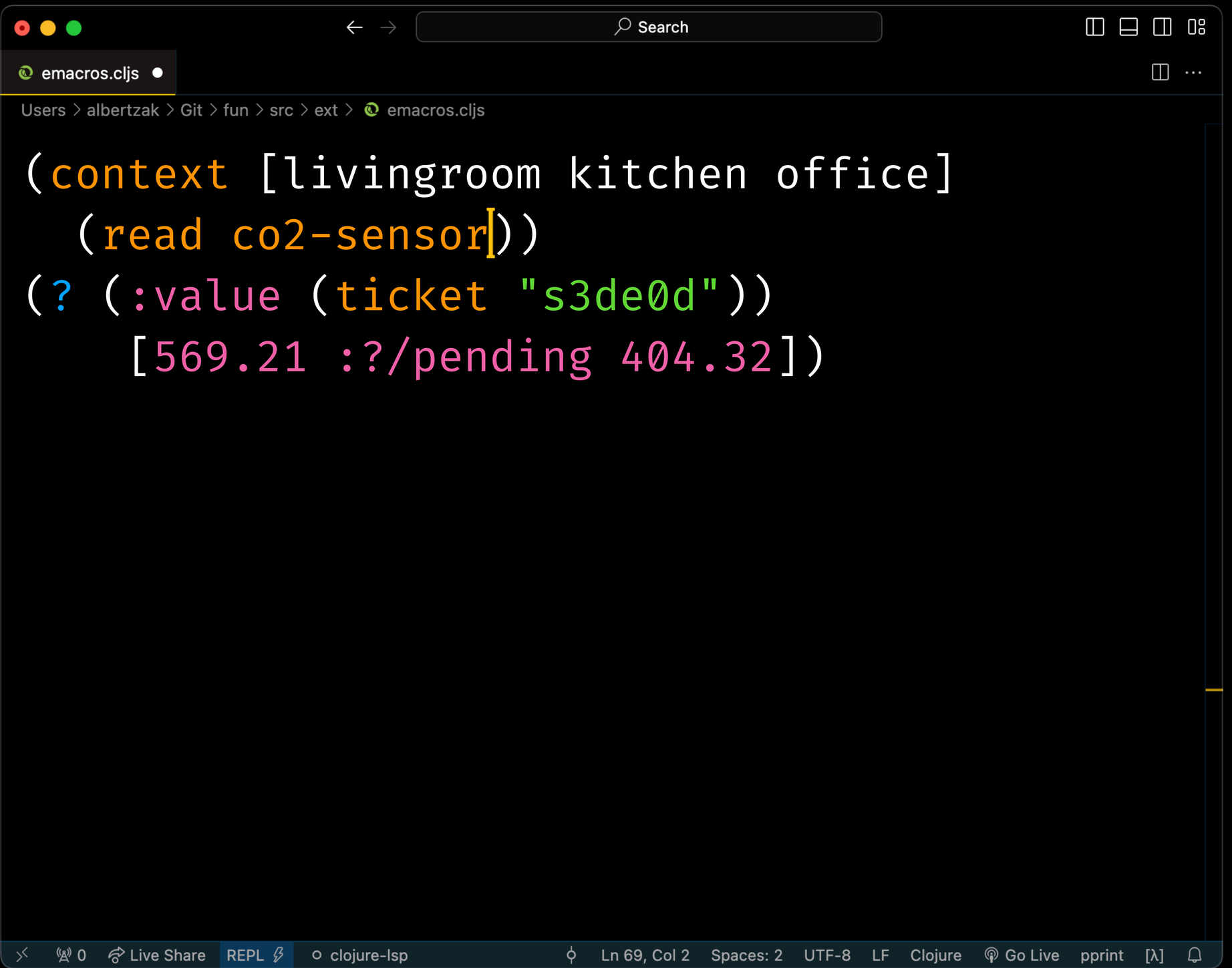


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

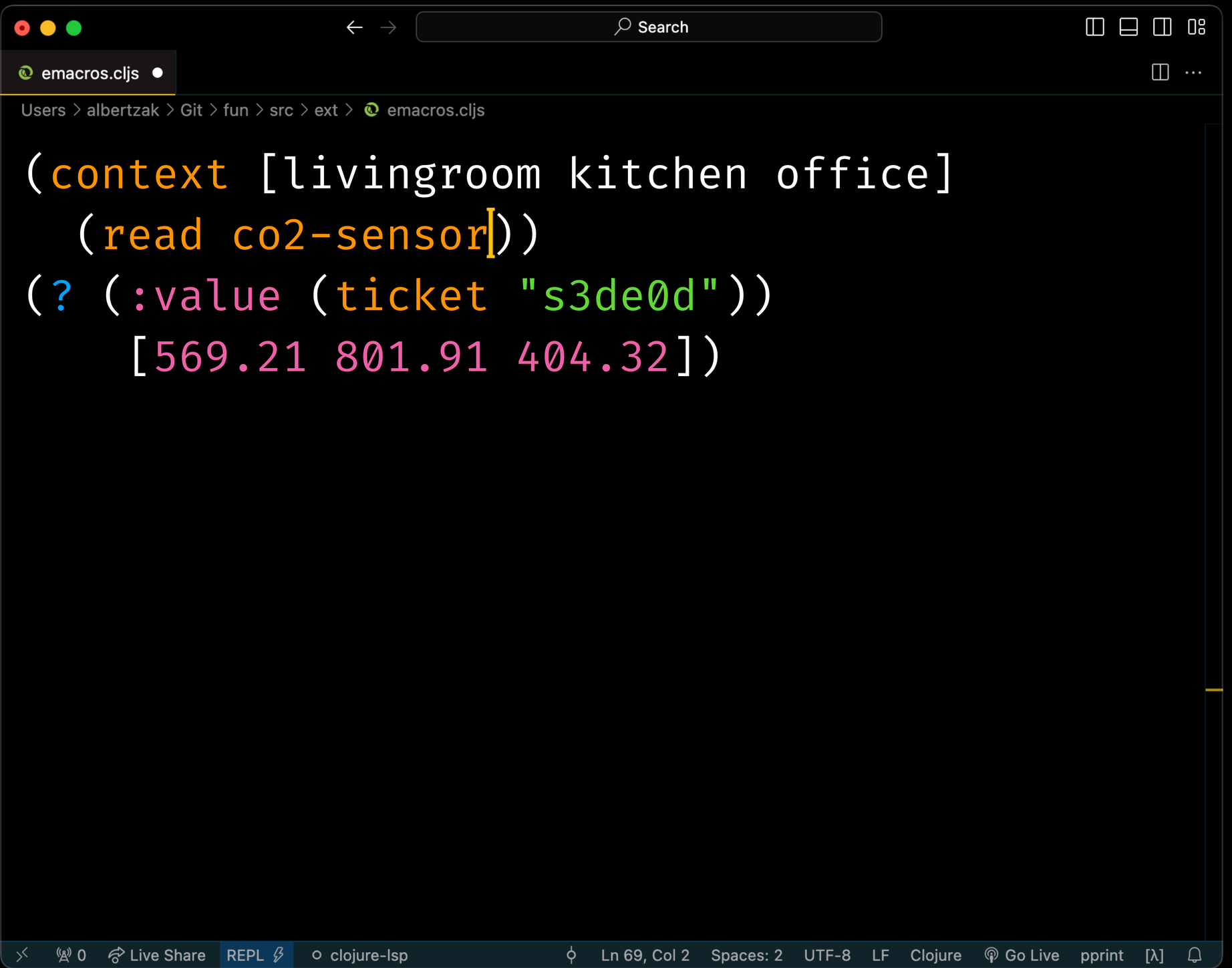


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

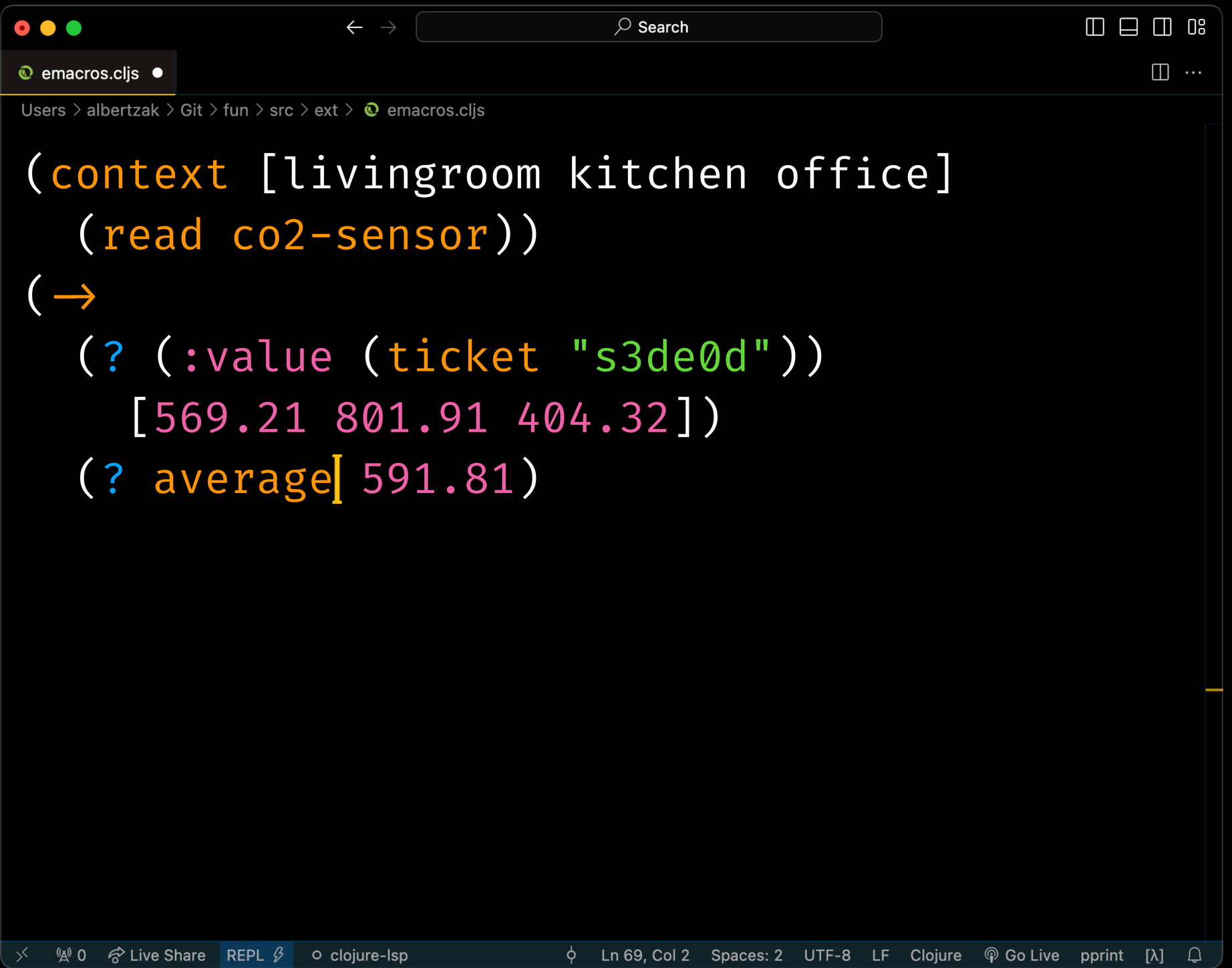


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

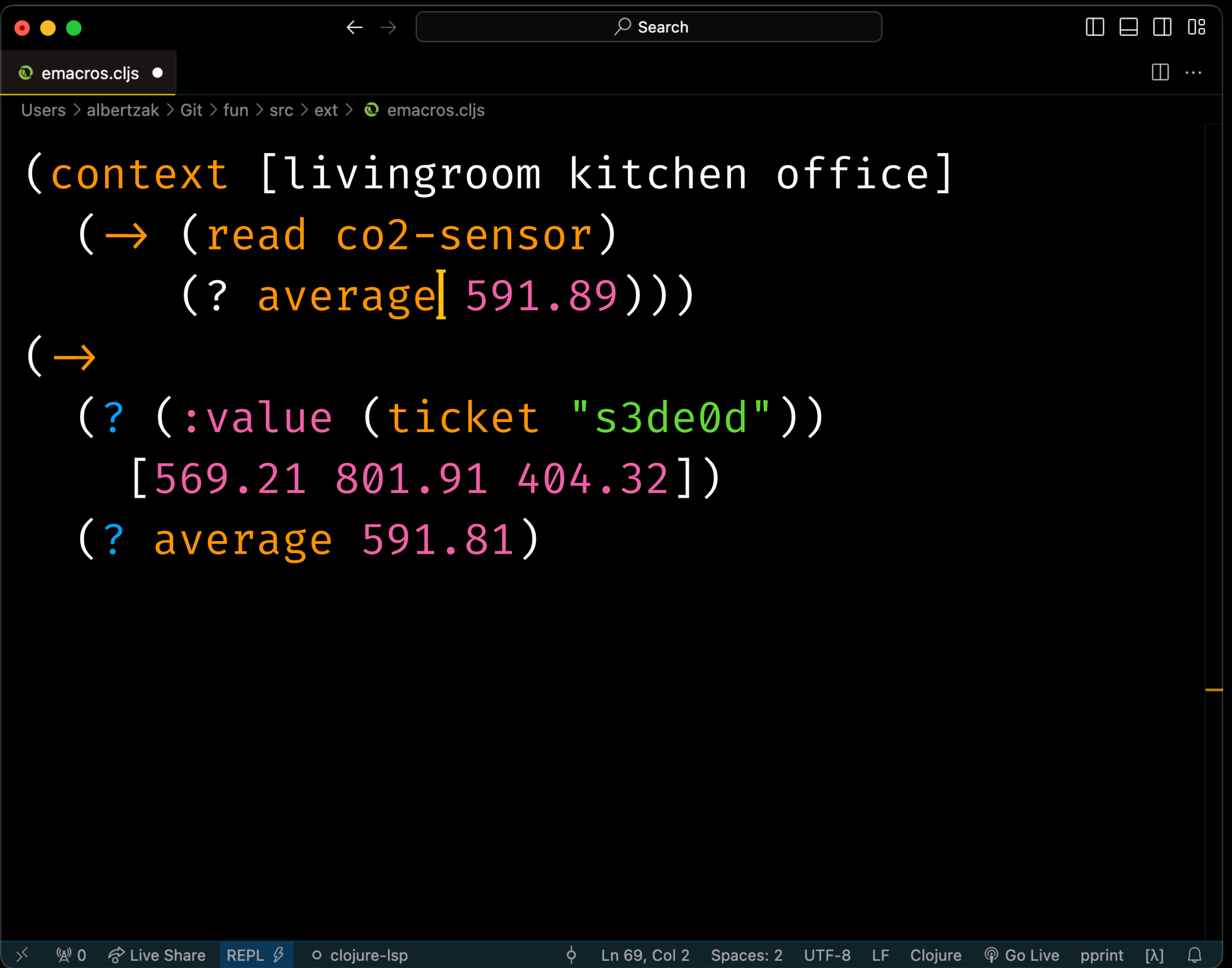


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

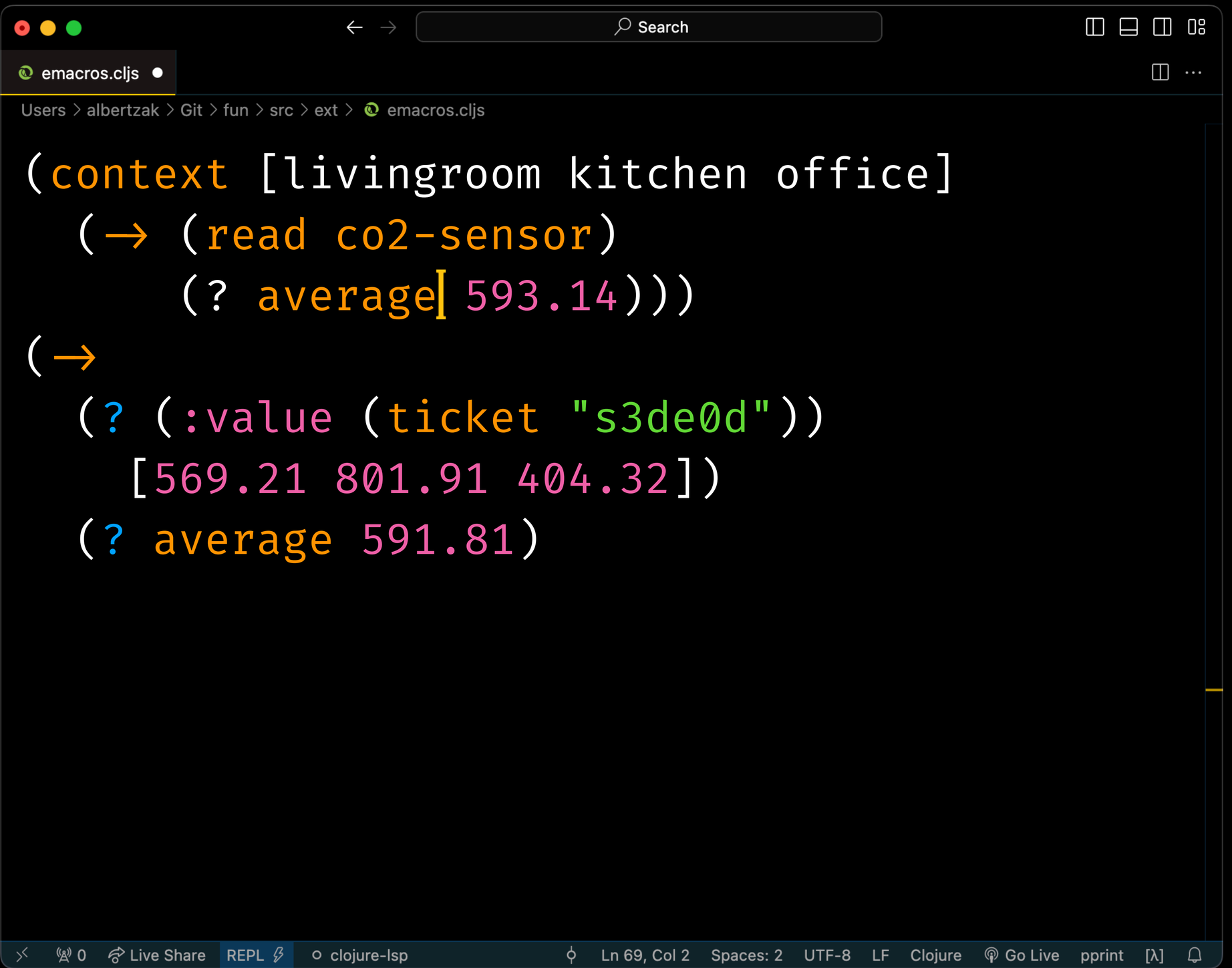


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

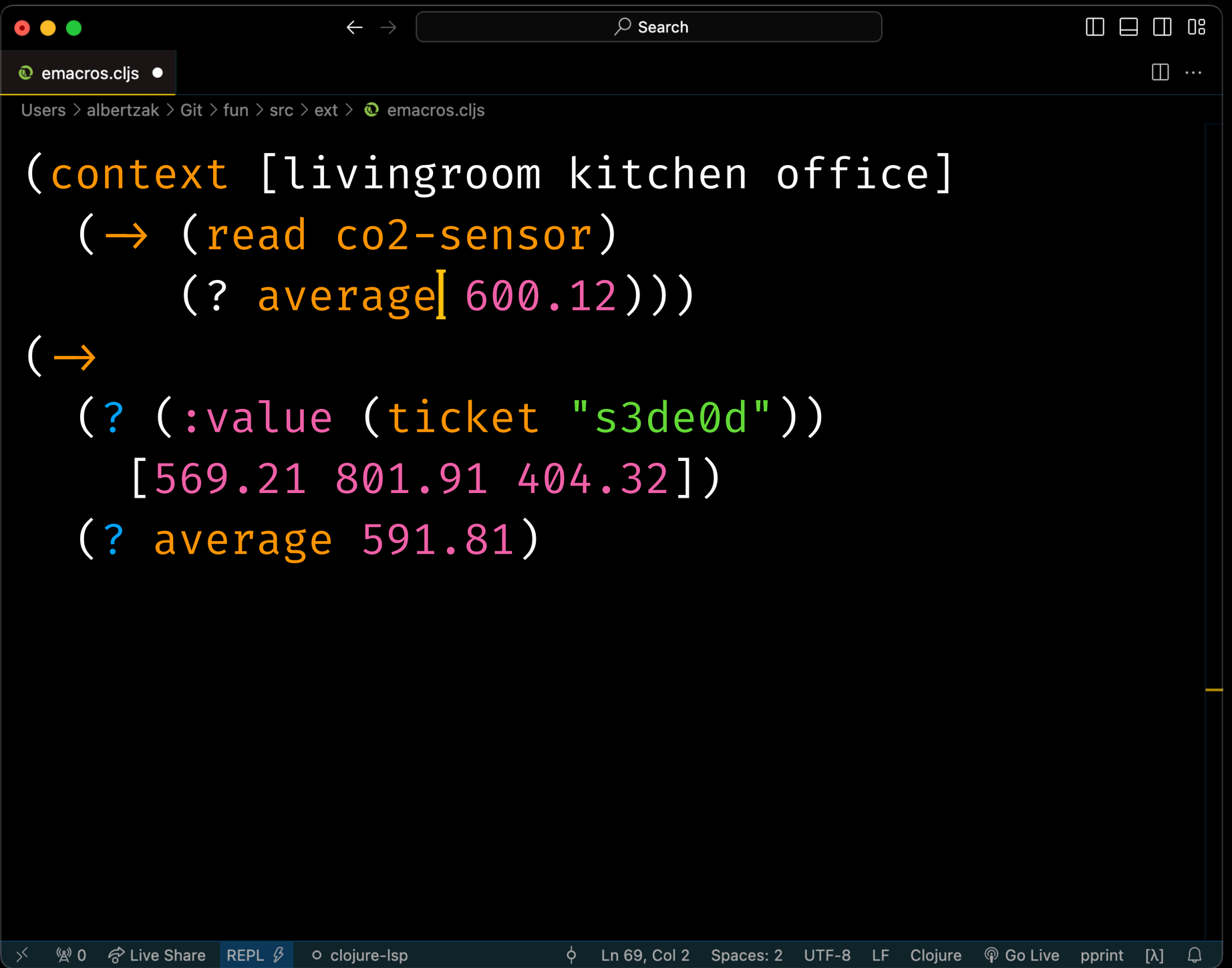


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

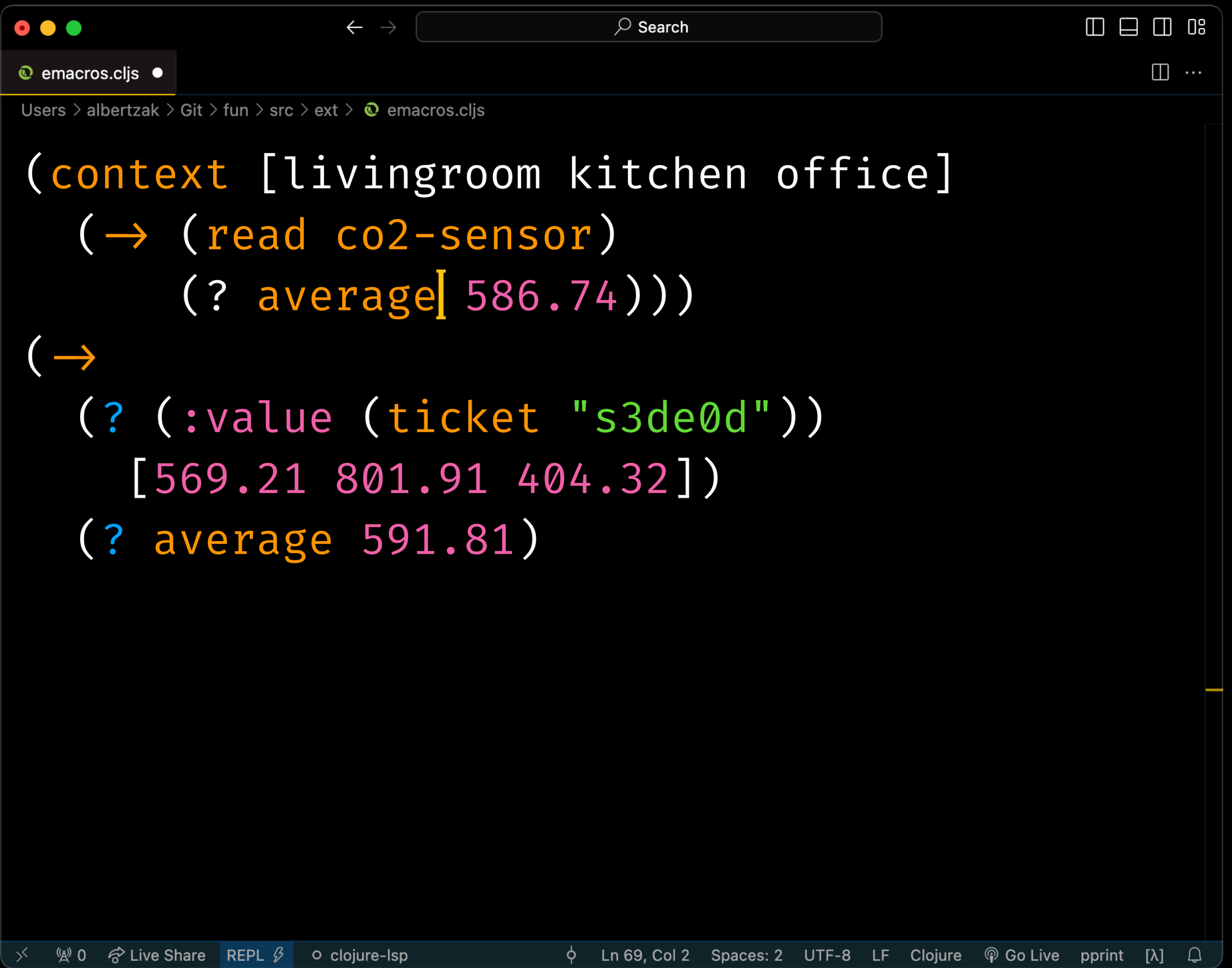


contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context



contribution

pull, as-you-type
async streams
1:n
explicit state
fault handling
stable time basis

tickets

context

future work

sigil-probe/eval arbitrary subexpressions

"under construction" sigil (+theming)

security: object capability emacros

nondestructive preview overlays: history, what-if

(thank-you)!

On REPLs

read

eval

print

DC-RES
Albert Zak
2023-11-03