S Query_handler

M { Unique
List_dir

name

config

sexp_of_config

config_of_sexp

t ← internal_state

create : config → t

eval : t → Sexp.t → Sexp.t Or_error.t

↑ query

S Query_handler_instance

Query_handler

this : Query_handler.t

Q: Query_handler config

↓ ↓

fun: build_instance

↓

I : Query_handler_instance

handlers : Query_handler_instance list

↓

fun: build_dispatch_table

↓

table : name → I

table name_and_equry : Sexp.t

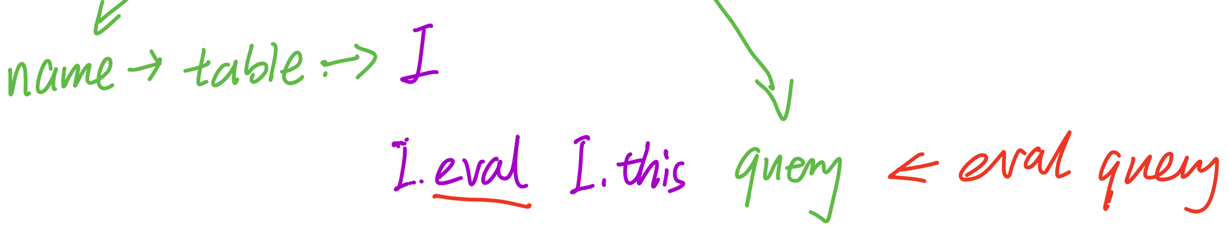
↓ ↓

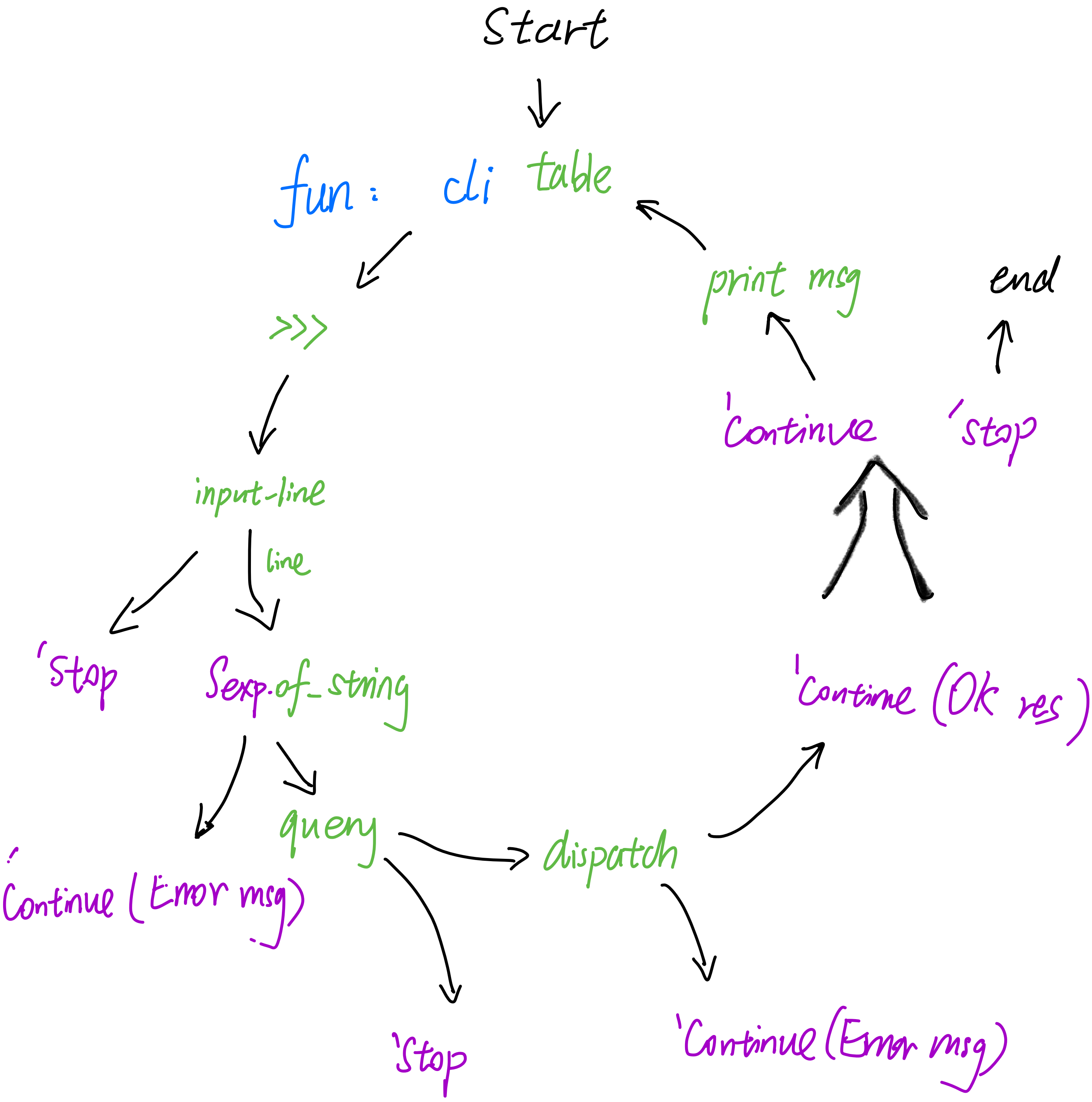
fun: dispatch

query format:
(query_name, query)

↓

name_and_equry





unique_instance

list_dir_instance

M Loader

name

config

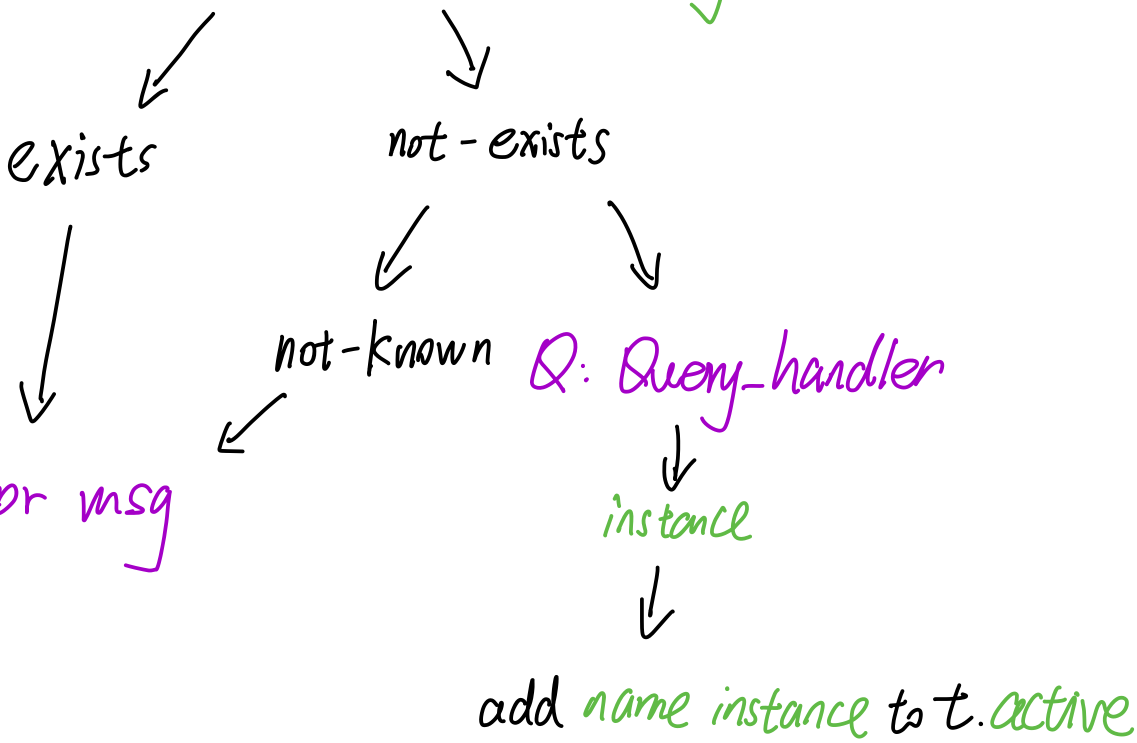
sexp_of_config

config_of_sexp

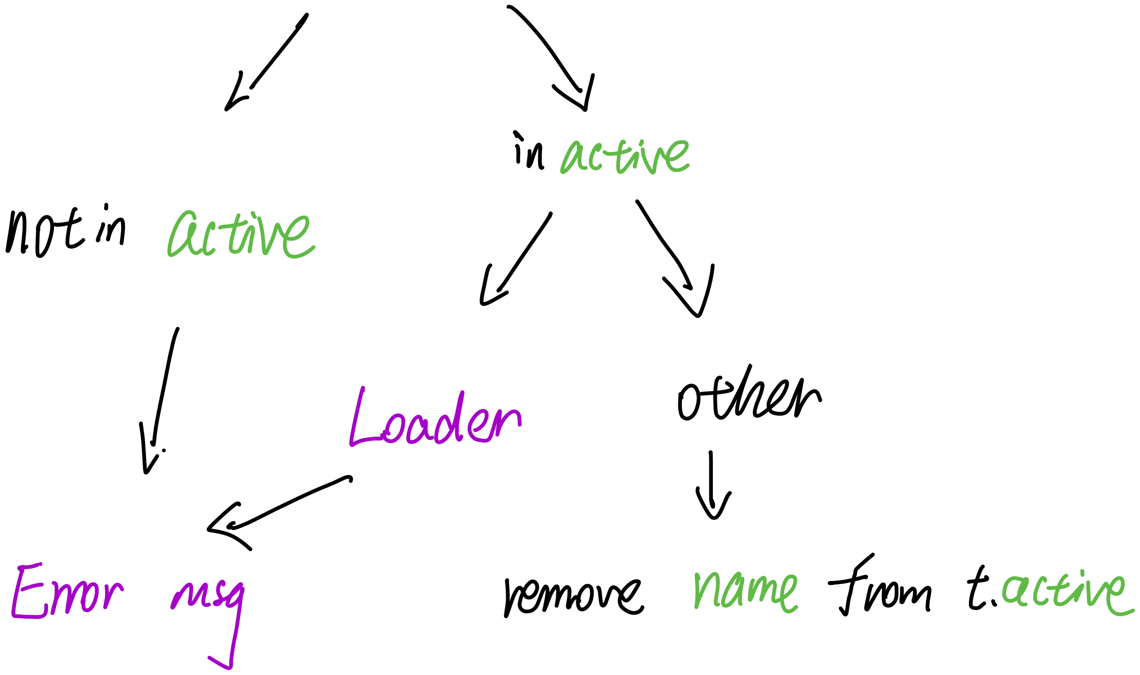
t < known
active

fun: create < know: module list
active: []

fun: load t name config



fun unload name



type request =

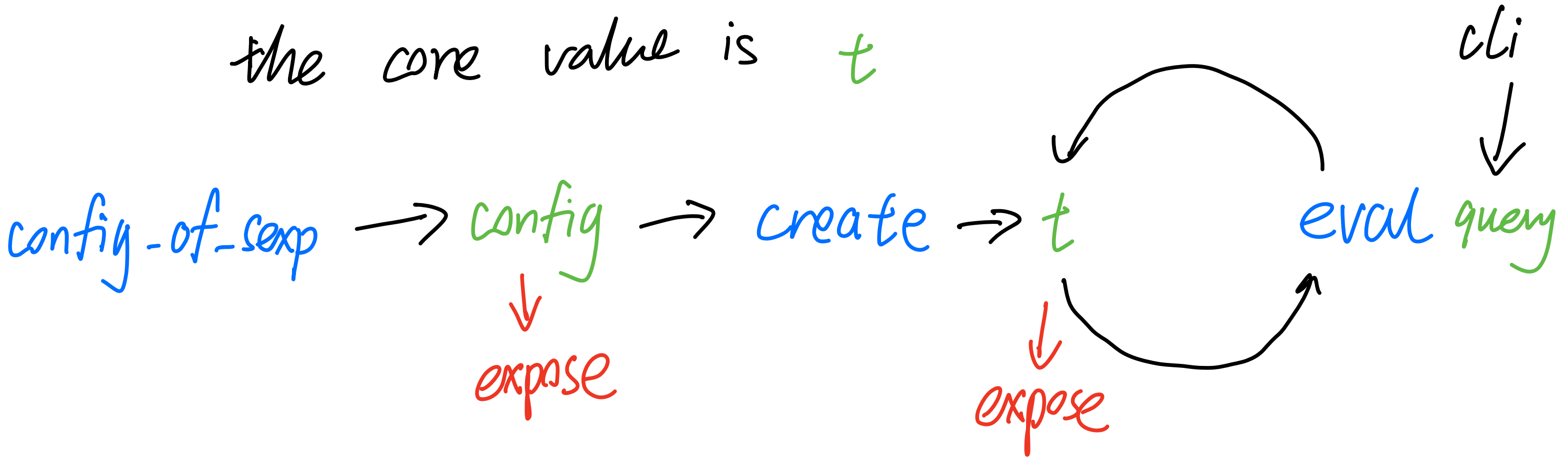
```
  | Load    name  config
  | Unload  name
  | Known_services
  | Active_services

                   Sexp.t
fun   eval   t   query
     Load    ->    load
     Unload  ->    unload
     Known_services ->   known -> sexp.t
   Unknown_services ->   active -> sexp.t
```

A module define some values and

some methods on these values

the core value is t



Query_handler and Query-handler_instance like

Class and Object

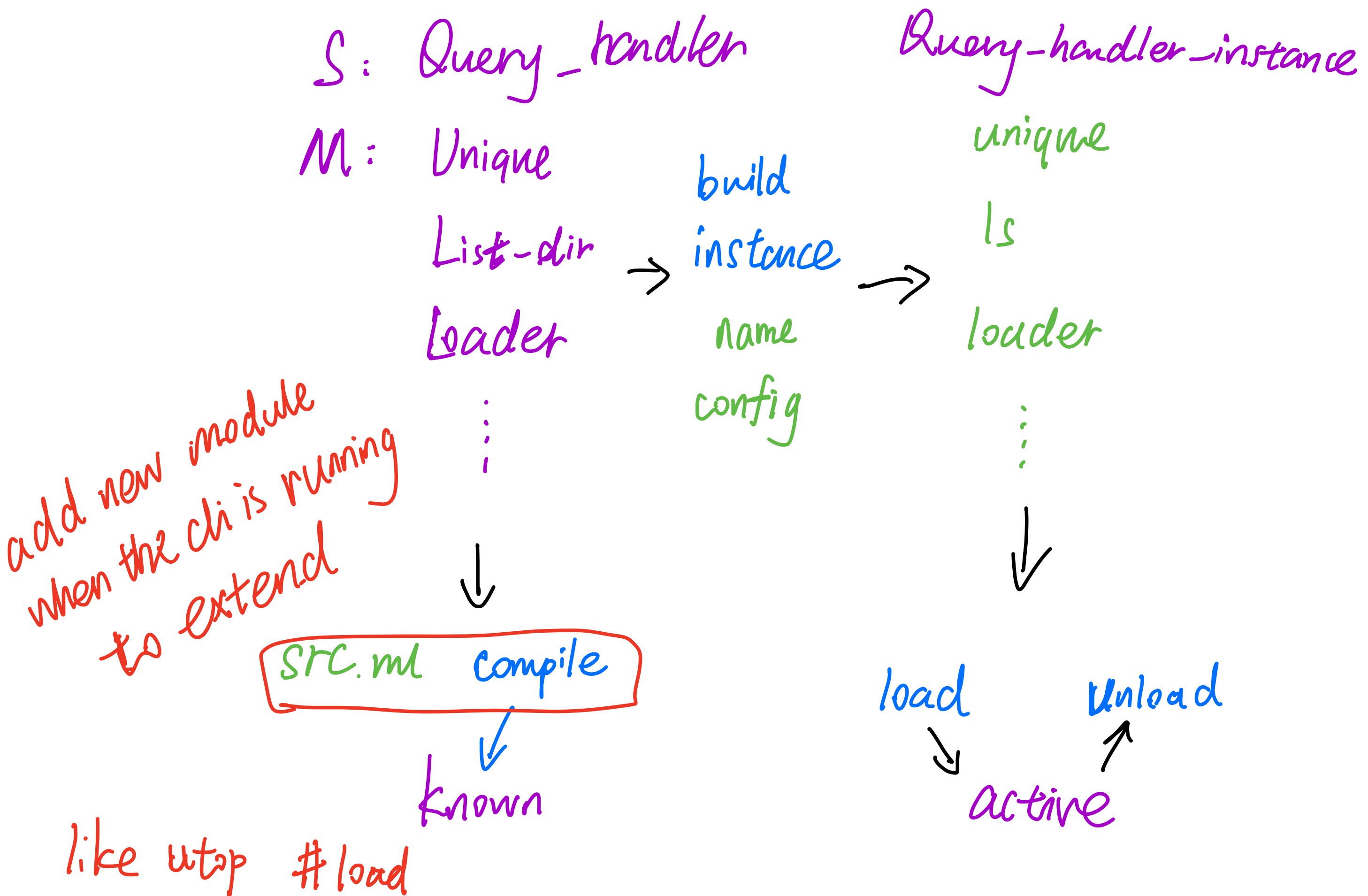
fun run

loader ← create all_known_Query_handle_module list

loader-instance ← < Loader
loader

Add loader_instance to loader.active

cli loader.active



S: Query_handler

Query-handler_instance

M: Unique

List-dir

Loader

⋮

build

instance

name

config

unique

ls

loader

⋮

src.ml compile

known

load

active

unload

add new module when the cli is running to extend

like utop #load