① ⇒ API (get, put,) - (load, range) - delete

② ⇒ ≈ design of access method ⇒ data structure design
⇒ algorithms to support API

③ ⇒ is the design ~~are they~~ concurrent? ↝ is No a valid answer? ⇒ I think no

Yes, because it may be a manual design that we test

or concurrency may not be required

④ is there a way to rank concurrent solutions ⇒ how much concurent is a design?

↳ what is the metric?

= concurrent actions ~~per~~ allowed

reads writes

Concurrent conflicting actions allowed

= we count independent / lockable pieces of the design

⑤ Do we get design variations because of concurrency or because of design?

⑥ Say we have design X and we want to make it "more concurrent"
Do we change the algorith or the layout?

⑦ Is our goal to give the best concurrent & correct search/put/*
algorith given a layout?

with that we can ⇒ Interactively give concurrency metric as design evolves
⇒ Fix bad algo designs for a layout.
bad = not bad but not optimal.

architecture : rule based system ⇒ capture DSL

how : is it OK to capture local design options and syntesize?
↳ or it has to be a global design choice?
↳ ideally we want to avoid that huge space

For example: Tree based structure ⇐ Can we argue separatelly about links & nodes?

↳ For example:

⇒ We will have rules anyway so we can do transformations

Metric of success:

correctness → From Dennis' paper

Concurrency rank? ↳ enumerate options? ⇐

Side-effects

how expensive is it?
search design → apply rules. *

competition

(is it possible to have a non concurrent design that is faster to run sequentially?)

(is it possible to have other concurrent designs that we do not capture due to the framework)