cpk_i	cpv_i	cps_i
ptr_k_i	ptr_v_i	s_i

retire[] is logical delete. retire(ptc_k_A) means ptr_k_A is not being held by any cpk_i. when ptr_k_A is no longer hazardous or being protected, it will be deleted. another thread can still deference ptr_k_A to get k_A but $ptr\_k_A \notin$ any $cpk_i \implies k_A \notin dB$ so returning v_A would be a correctness issue.

$s_i == F$ ✓

$maybe\_ptr\_k_i := cpk_i.load[]$

$maybe\_ptr\_k_i \neq \phi$ ✓

$reload := cpk_i.load[]$

$reload == maybe\_ptr\_k_i$ ✓

$ptr\_k_i := maybe\_ptr\_k_i$

protect()

$ptr\_k_i \neq \phi$ ✓

ptr_k_i could have been swapped here for ptr_k_i' but that's fine

* $ptr\_k_i == k_X$ ✓

ptr_k_i swapped with $ptr\_k_i' \rightarrow k_X'$: $ptr\_k_i \neq \phi$

$ptr\_k_i \neq \phi$

$cas(F \rightleftarrows X)$ //no deletes of k_X are possible now

$ptr\_k_i \neq \phi$

retire($ptr\_k_i \cap ptr\_v_i$)

cpk_i	cpv_i	cps_i
ϕ	ϕ	D

swap results in this branch

last delete results in this branch

$cas[F \rightleftarrows X]$ //no deletes of k_X are possible now

cpk_i	cpv_i	cps_i
ϕ	ϕ	D

$ptr\_k_i \neq \phi$ ✓

k_X' deleted instead of k_X!

$ptr\_k_i \rightarrow \phi$ ✓

$ptr\_k_i \neq$ ~~$cpk_i.load[]$~~ ϕ ✓

$cps_i := F$

cpk_i	cpv_i	cps_i
ϕ	ϕ	F

currupted state!