Univerzální bounded wait-free protokol [Herlihy]

$Op: Stav \times Arg \to Stav \times Res$

Stav struktury

Cell:
- seq (int) – pořadové číslo
- arg – argument operace, kterou vznikl akt. stav
- new – akt. stav (objekt pro konsensus)
- next – ukazatel na další buňku (obj. pro konsensus)
- prev – předch. buňka (atom. ukazatel)

globální:
- announce[P] ← právě přidv. buňka
- head[P] ← ukazatel na konec seznamu

na poč. kotva

"<" na buňkách porovnává seq

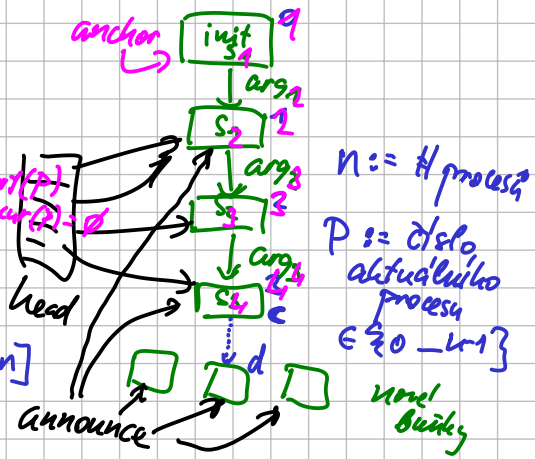
min(cell1, cell2)

max(head) := max c (c ∈ head)

Operate(arg):

1. mine ← Cell(seq=0, arg=arg, new=∅, next=∅, prev=∅)
2. announce[P] ← mine • start(P) • concur(P) = ∅
3. head[P] ← max(head)
4. while mine.seq = 0:
5. c ← Head[P]
6. help ← announce[c.seq % n]
7. if help.seq = 0: prefer ← help
   else: prefer ← mine
8. d ← c.next.decide(prefer)
9. d.new.decide(Op(c.new, d.arg))
10. d.prev ← c
11. d.seq ← c.seq + 1
12. head[P] ← d
13. head[P] ← mine
14. Return mine.new

13. přidání do všech concur(*)



start(P) ← max(head).seq v okamžiku announcování procesem P

concur(P) := množina buněk, které přibyly do hlav od okamžiku announce v P

---

max(head).seq = start(P) + |concur(P)|

(L1) Pokud |concur(P)| > n, pak announce[P] ∈ head (během výpočtu procesu P)

Dk: ↳ concur(P) obsahuje buňky se seq ≡ p-1 (mod n) ← buňka q přidaná procesem Q

seq ≡ p (mod n) ← buňka r přidaná procesem R

- q ∈ concur(P) ⇒ Q připojil buňku q poté, co P ohlédl
- R připojil r za q ⇒ v kroku 5 procesu R je head[R] nastaveno na q
- předtím Q nastavil head[Q] ← q ⇒ buňka q je připojena procesem Q

↳ v kroku 5 procesu R (čtení head[R]) je už nastaveno announce[P]

- pak je buď ann[P] už připojená nebo r = ann[P]

|concur(P)| := max(head).seq − start(P)

(L2) $\max(head)_{.seq} \geq start(P)$

(L4) po kroku 3: $head[P]_{.seq} \geq start(P)$

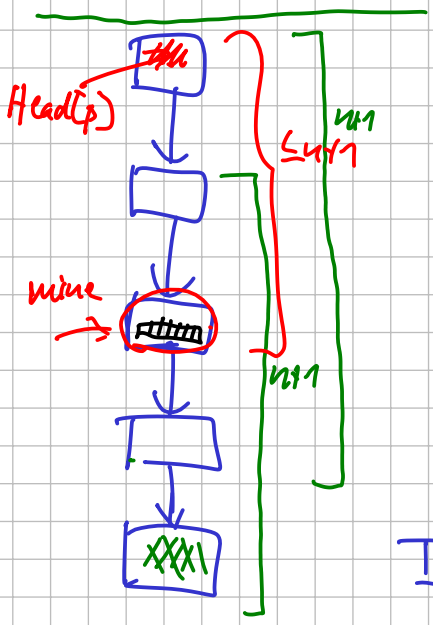
(L5) $|concur(P)| \geq head(P).seq - start(P) \geq 0$ (L4)

↳ v každém průchodu cyklem while se dolní odhad na |concur(P)| zvýší aspoň o 1.

⇒ po nejvýše n+1 průchodech platí předp. od (L1)

⇒ cyklus skončí!

Správa paměti



Naivní "invariant": vždy se pohybujeme po posledních n+1 buňkách seznamu.

Buňka je při vzniku zamčena, potřebuje (n+1)-krát odemknout, než je uvolněna.
→ šetch & add
→ pole n+1 atom. bitů

Alokace: † proces má pool velikosti ≥ n²
vždy při alokaci projde pool a najde (n+1)-krát odemč. buňku, tu recykluje
} alokace trvá O(n²) s počítadly, O(n³) bez nich

n−1 procesů, každý blokuje max. n+1 buněk } blokováno max. (n+1)(n−1) = n²−1 buněk

Tvrzení: Buňka, kterou zkoumám, je ve vzdálenosti max. n+1 buď od konce seznamu nebo od mine, je-li mine zalokována.

⇐ je dosud zamčena → není recyklována.