Soluzione basata sui semafori

Introduciamo 3 stati: THINKING, HUNGRY, EATING

- Thinking: pensa ovviamente
- hungry: ha fame, non ha le forchette
- eating: ha le forchette

Vediamo che un filosofo faccia questo percorso

Da Hungry a eating passerà quando avrà il "permesso"

Usiamo un vettore costante state[N] che permette di esprimere lo stato di ogni filosofo

Usiamo un semaforo mutex per la mutua esclusione su state e dei semafori, uno per filosofo, conservati nel vettore s[N].

Questi semafori permettono al filosofo di addormentarsi se non riesce a prendere le forchette e di essere risvegliati poi dai vicini che avranno finito di mangiare

Funzioni (codice su slide)

- left(i): da vicino left di i
- right(i)
- test(i): se i è H e i vicini non sono E allora i passa a E (e gli facciamo up sul semaforo)
- takeforks(i): lo stato passa a H e facciamo la test. Le due operazioni sono racchiuse tra down e up su mutex.
  Alla fine facciamo down sul semaforo

- get_forks(i): lo stato passa a T e lasciamo test sui due vicini. Le V.C. sono racchiuse tra down e up sul mutex
- giveBackForks(i): banalmente il filosofo finisce, poi prende le forks (e prova), mangia e posa le forchette.

Codice su slide

Vogliamo rendere la test bloccante in un qualche modo, ma la test sta tra down e up mutex. Blocchiamo tutto.

Facciamo un trucchetto: se non ci sono le condizioni nella test perché i pari a E non vicine fatto la up(S[i]) e dunque la down (fuori dalla sezione critica) in ~~take~~ forks diventa bloccante. Lo sblocco sarà da parte di un altro filosofo facendo la get_forks (che fa le test sui vicini)