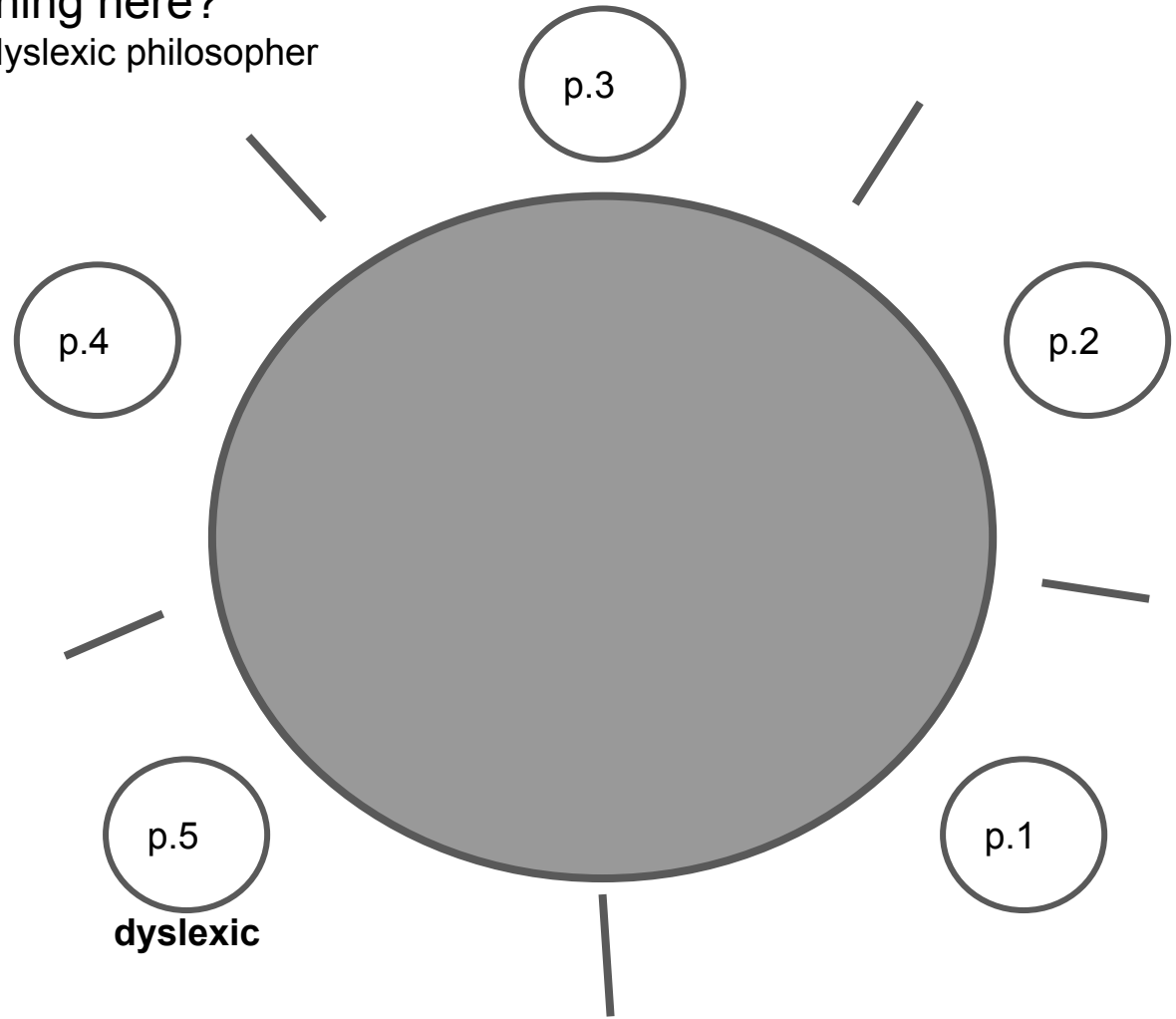
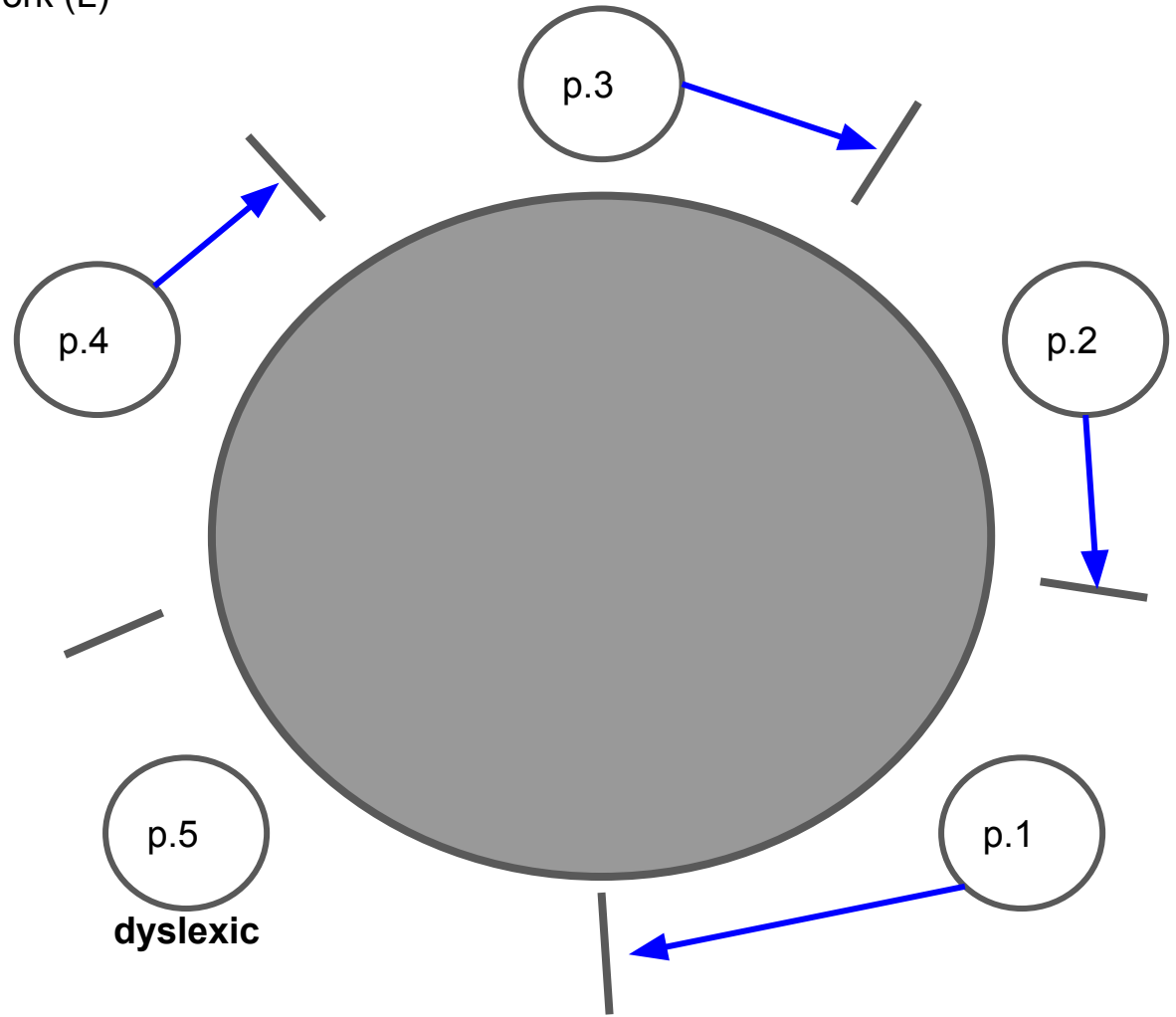
What's happening here?

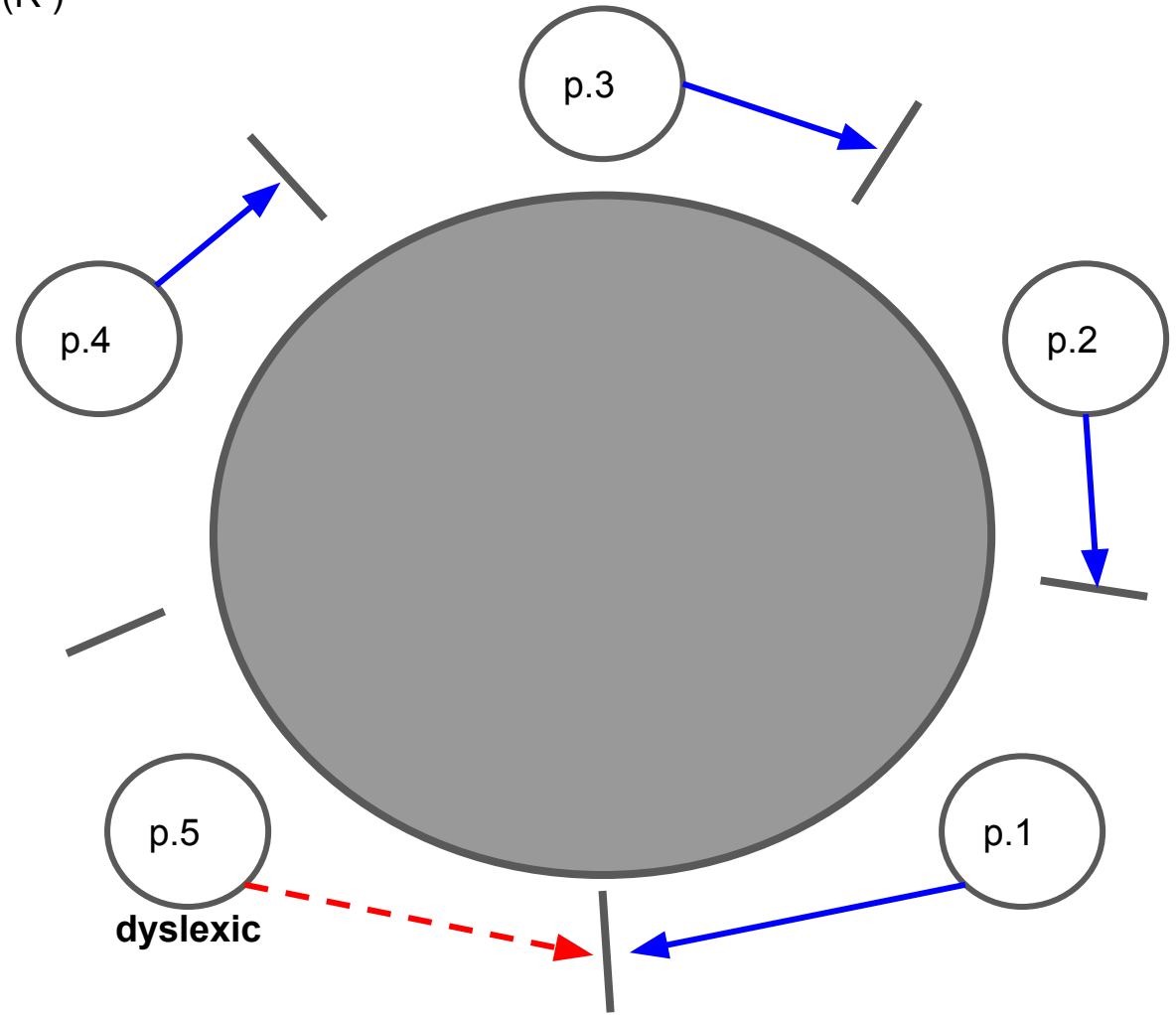
First Setup: One dyslexic philosopher



p.1-p.4 reach 1st fork (L)
→ got it!

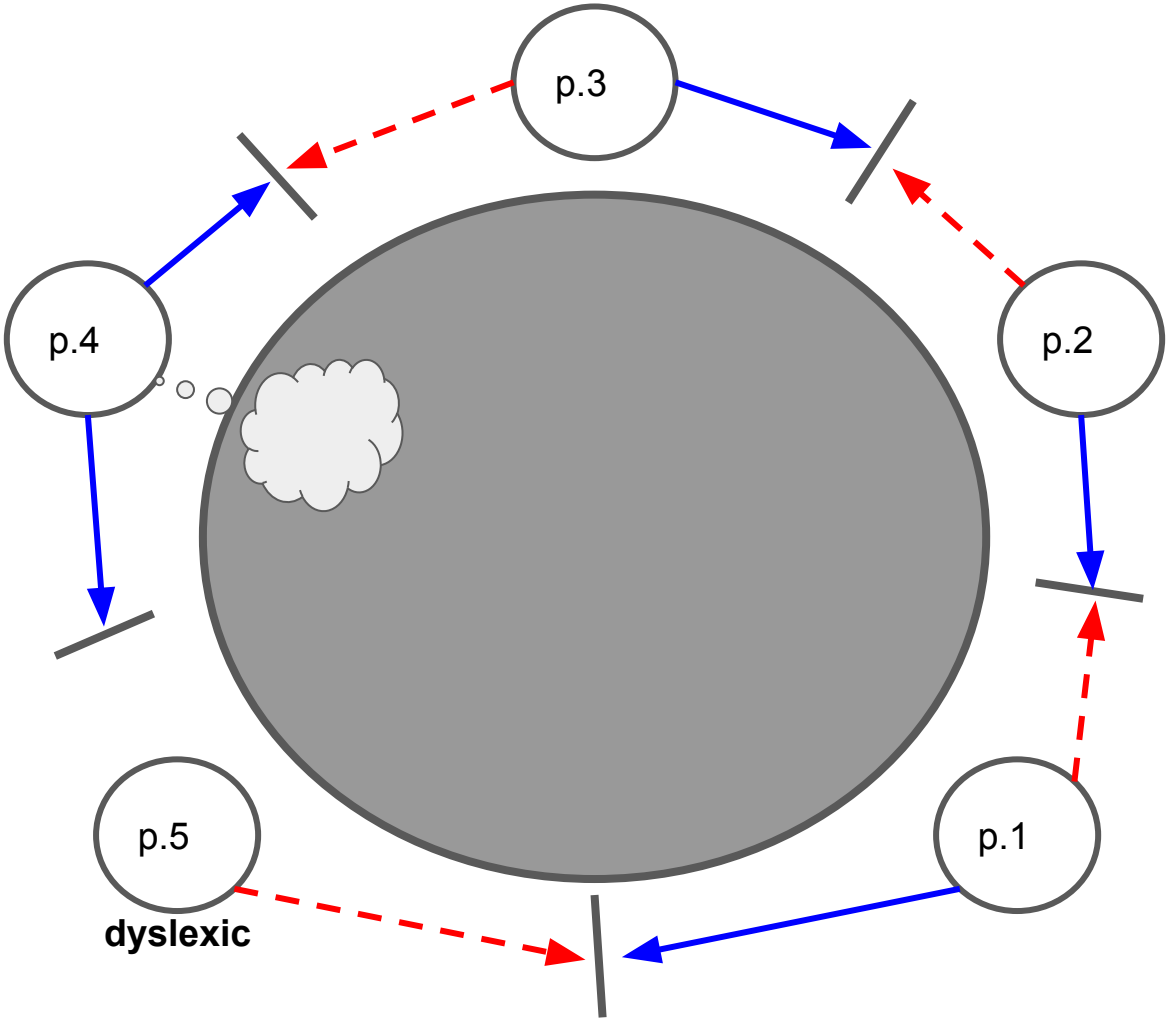


p.5 reach 1st fork (R)
→ denied



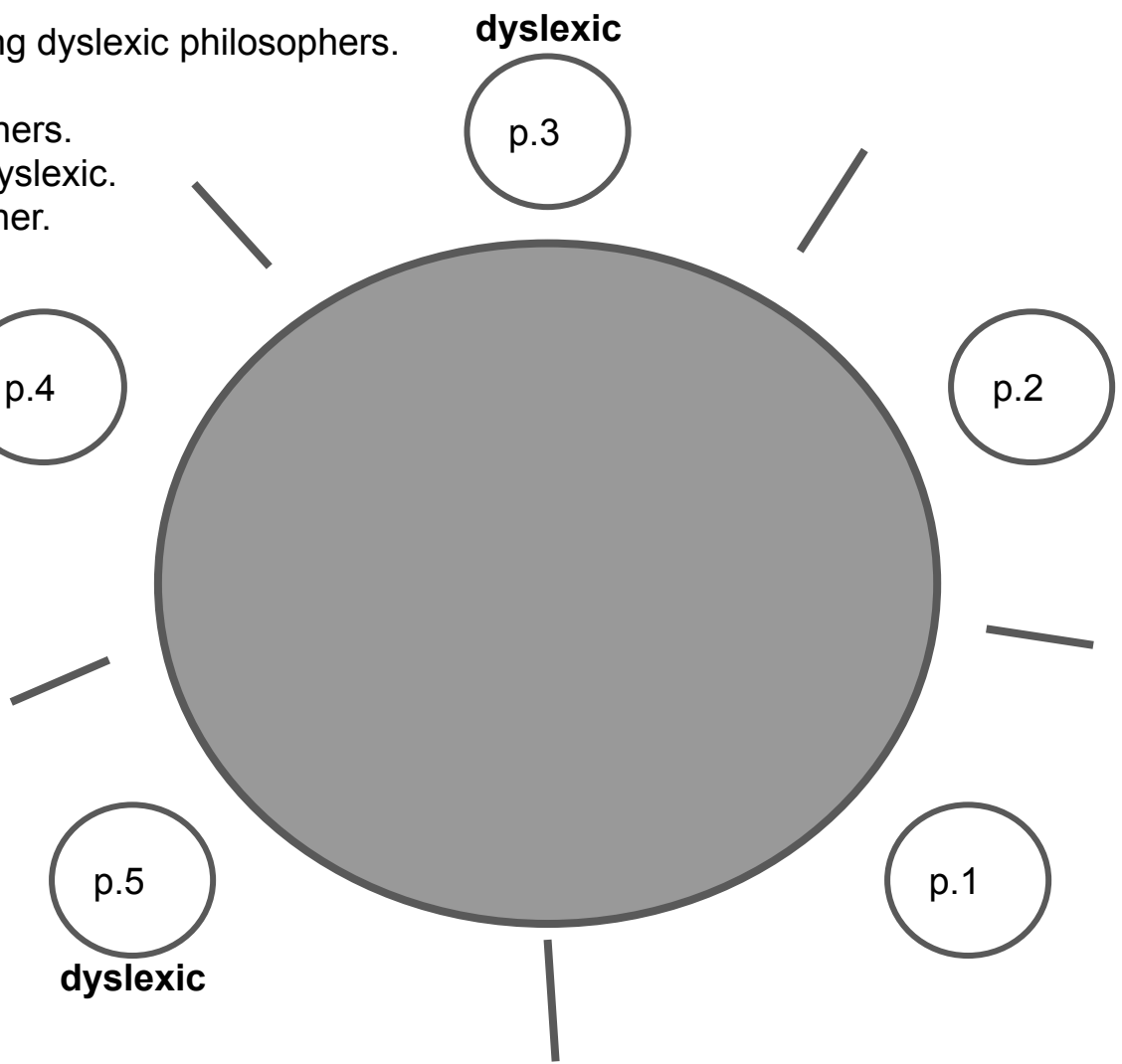
p.4 second-lock ®
& EAT

→ Only 1 can eat.
Surely a more
optimal solution...

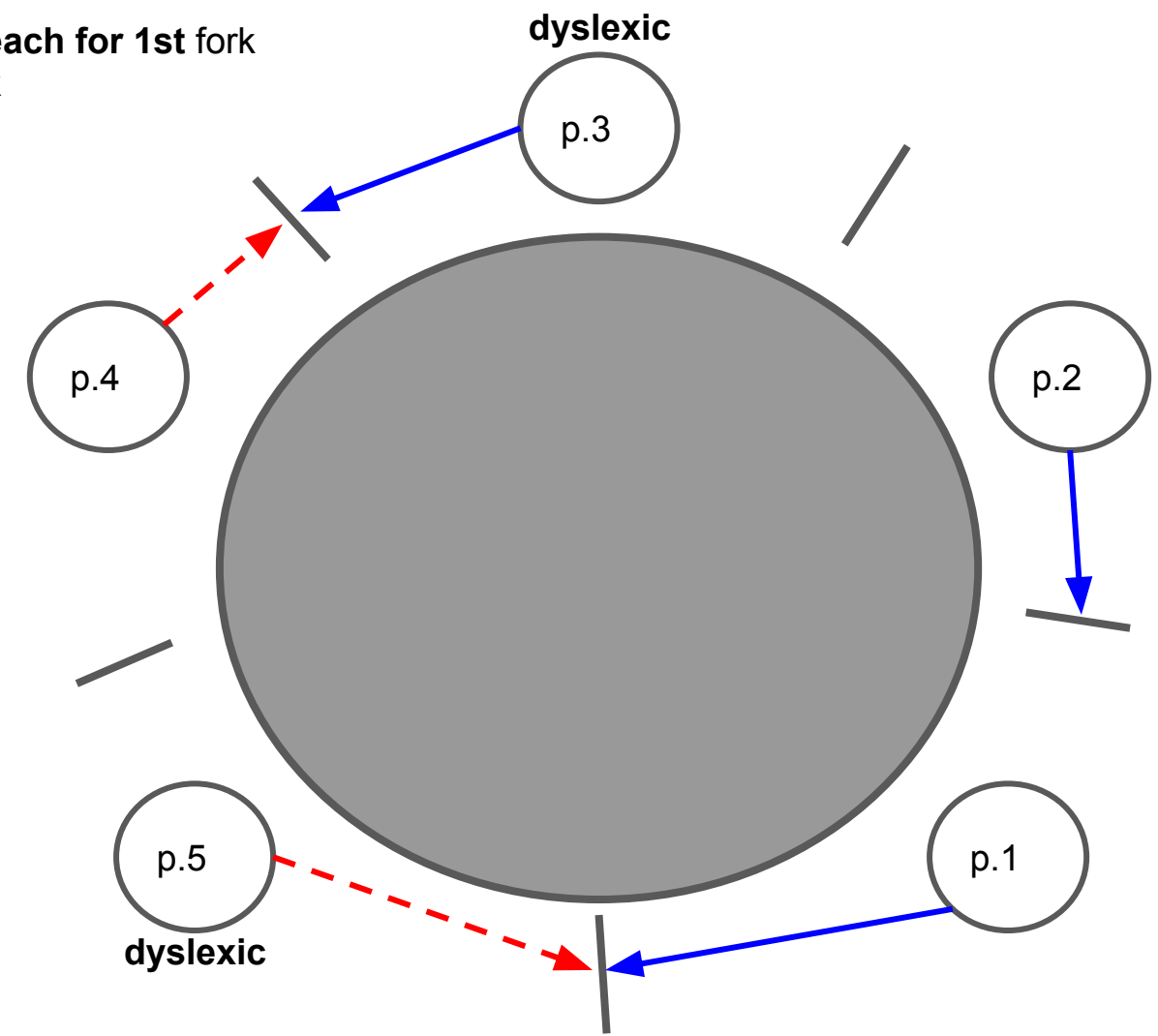


New Setup: Alternating dyslexic philosophers.

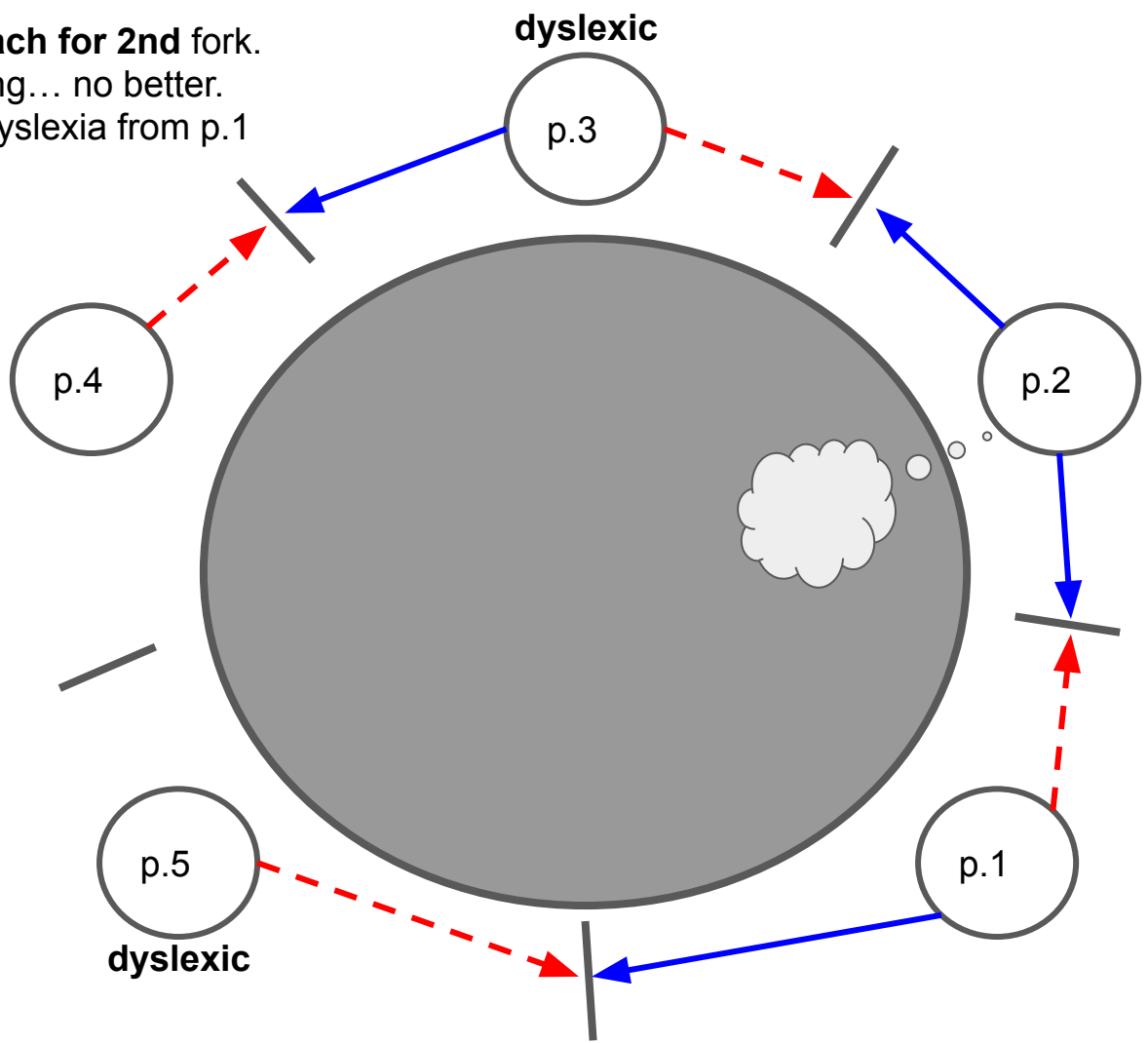
Let N = # of philosophers.
Let N//2 of them be dyslexic.
Space them every other.



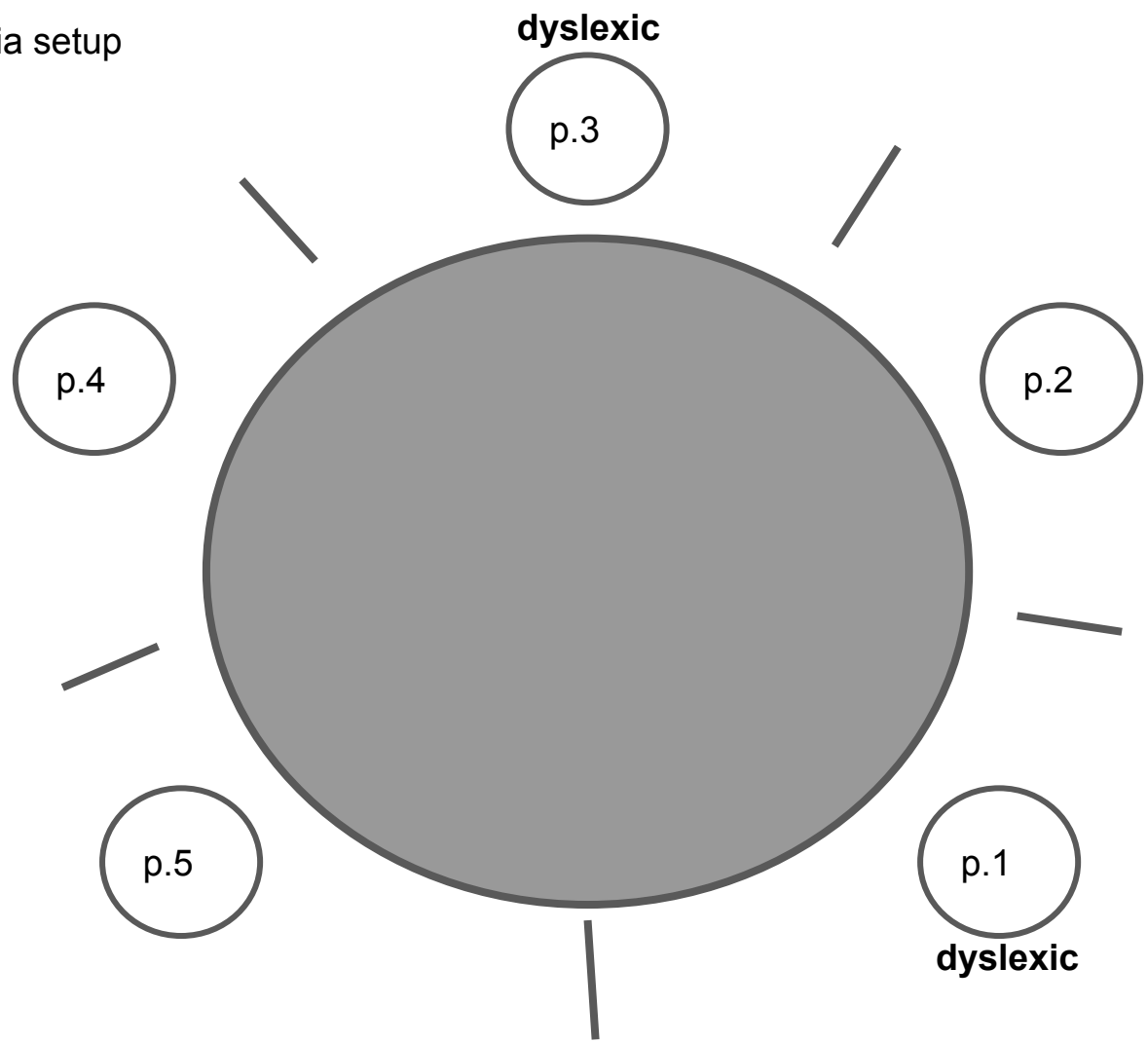
P.n increasing, **reach for 1st** fork
→ 3 get their fork
→ 2 are denied



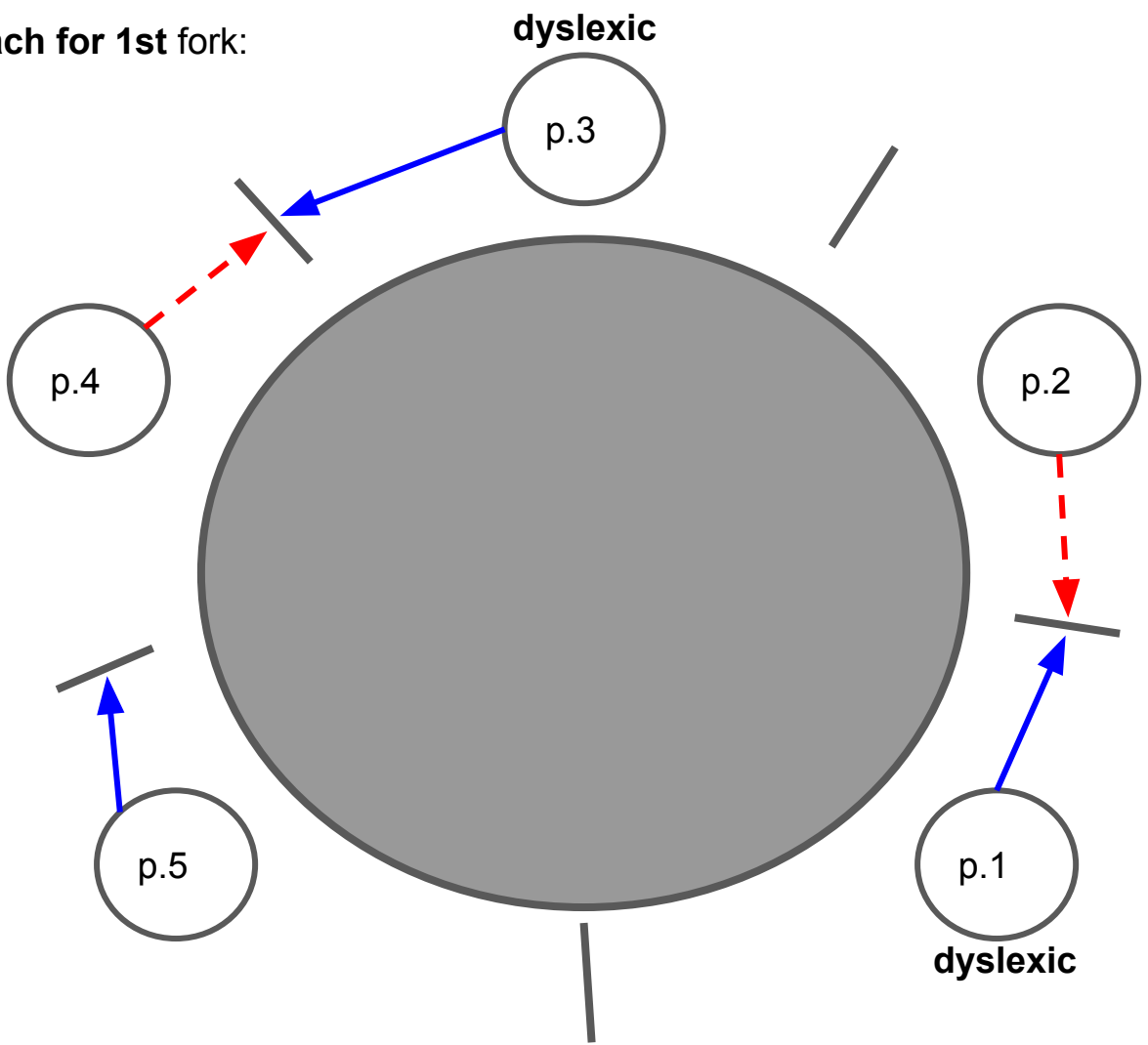
P.n increasing, **reach for 2nd** fork.
→ Still only 1 eating… no better.
→ Try assigning dyslexia from p.1



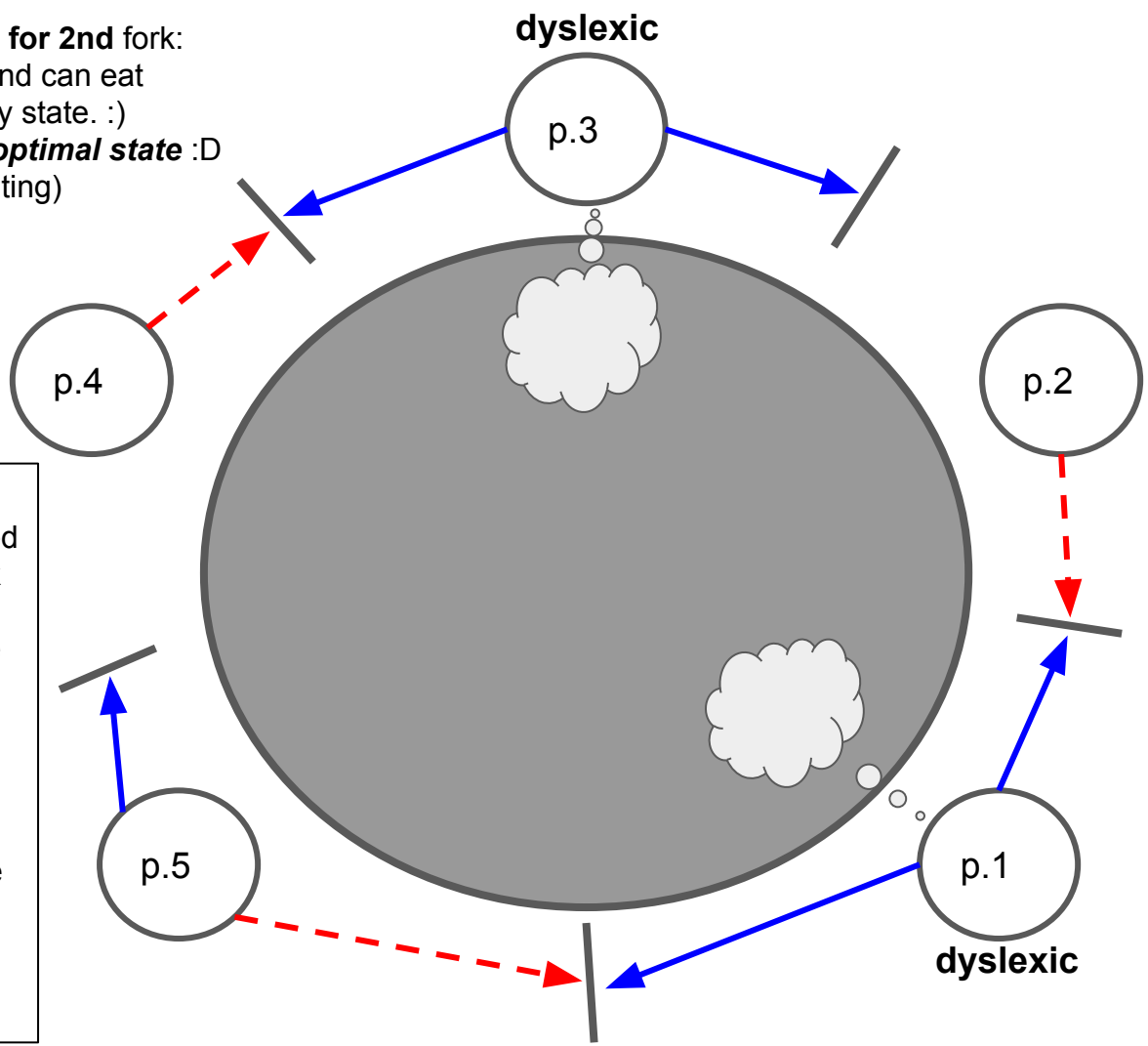
Alternating dyslexia setup
(Try #2): p.1 start



p.n increasing, **reach for 1st** fork:
→ 3 get a fork.
→ 2 are denied.



p.n increasing, **reach for 2nd** fork:
→ now, 2 get theirs and can eat
→ Improved efficiency state. :)
→ In fact, we hit the ***optimal state*** :D
(N//2 philosophers eating)



Discussion: In this example, we assumed the philosophers pick up forks in order. However, in real-time threading, order will be random. The important thing is to space the dyslexic philosophers every other, which adds the (N//2 eating) state to the pool of possible states.