OS Project

Title : Case study : Dining Philosopher, Drinking Philosopher, Mobile Philosopher, Readers Writers Problem

Team Name : 4 Nodes

Files related to this project are uploaded on Github and can be found in the following link:

https://github.com/rishabhmurarka7/OS-Project

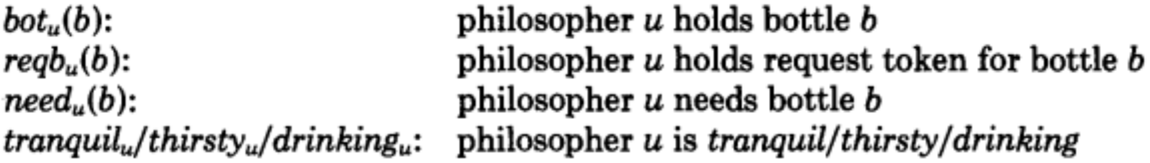
<u>Solution to Drinking Philosophers Problem</u>

We use forks to implement H. Forks are auxiliary resources in the sense that their sole purpose is to implement precedence graph H. In the graph we can determine the precedence between u and v if u holds the fork and the fork is clean then u has precedence else if it is dirty v has precedence.

The real resources in the drinkers problem are bottles. Our philosophers can eat and drink simultaneously, and we emphasize that eating is an artifact of our solution, used only to guarantee fair drinking.

In our solution, the state of a philosopher is a pair (diner's state, drinker's state) where a diner's state is one of thinking, hungry, or eating and a drinker's state is one of tranquil, thirsty, or drinking. Now we define the dining characteristics of our philosophers. We give rules that ensure that all thirsty philosophers drink in finite time.

We introduce a request token, reqb, for every bottle b.
The following Boolean variables are used:



$bot_u(b)$:	philosopher u holds bottle b
$reqb_u(b)$:	philosopher u holds request token for bottle b
$need_u(b)$:	philosopher u needs bottle b
$tranquil_u/thirsty_u/drinking_u$:	philosopher u is *tranquil/thirsty/drinking*

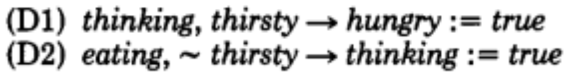
A bottle and the request token for it are held by different philosophers; that is, if u, v share bottle b, then u holds the bottle $(bot_u(b),\ reqb_v(b),\ \sim bot_v(b),\ \sim reqb_u(b))$,and v the token ,

or v holds the bottle and u the token.

From the problem statement we have,

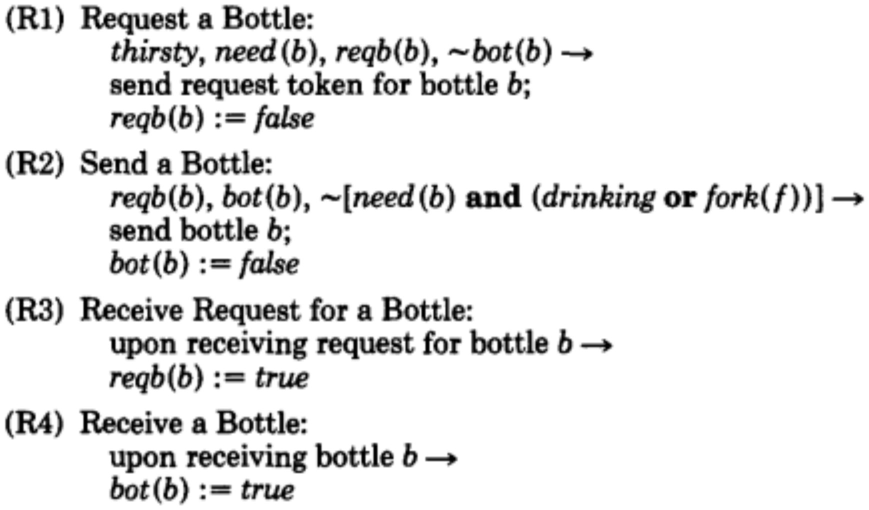
$$tranquil \Rightarrow \forall b[\sim need(b)]$$

State transitions for dining philosopher determined by drinking states are



(D1) *thinking, thirsty* → *hungry* := *true*
(D2) *eating,* ~ *thirsty* → *thinking* := *true*

Rules for bottle and request transmissions {Let f be the fork corresponding to bottle b, i.e., fork f and bottle b are shared by the same two processes}:



(R1) Request a Bottle:
thirsty, *need*(*b*), *reqb*(*b*), ~*bot*(*b*) →
send request token for bottle *b*;
reqb(*b*) := *false*

(R2) Send a Bottle:
reqb(*b*), *bot*(*b*), ~[*need*(*b*) **and** (*drinking* **or** *fork*(*f*))] →
send bottle *b*;
bot(*b*) := *false*

(R3) Receive Request for a Bottle:
upon receiving request for bottle *b* →
reqb(*b*) := *true*

(R4) Receive a Bottle:
upon receiving bottle *b* →
bot(*b*) := *true*