Estrategia de apropiamiento y espera para solucionar el problema de los filósofos comensales.

Introducción

A continuación, estudiaremos un par de soluciones al problema de los filósofos comensales, en las cuales, los filósofos se apropian de los tenedores siempre que tengan oportunidad, incluso aún cuando no pueden empezar a comer (se quedan en estado hambriento). Para esto, vamos a considerar que hay dos tipos de filósofos:

- Tipo L. Filósofos que se preocupan por primero tomar el tenedor izquierdo y posteriormente el de la derecha (podemos pensar que son zurdos)
- Tipo R. Filósofos que tratan de tomar primero el tenedor a su derecha y luego el que está la izquierda.

El algoritmo LR

Vamos a considerar que los filósofos son numerados consecutivamente desde el 1 al n, de tal manera que:

- El filósofo i es tipo R si y sólo si i es par.
- El filósofo i es tipo L si y sólo si i es impar.

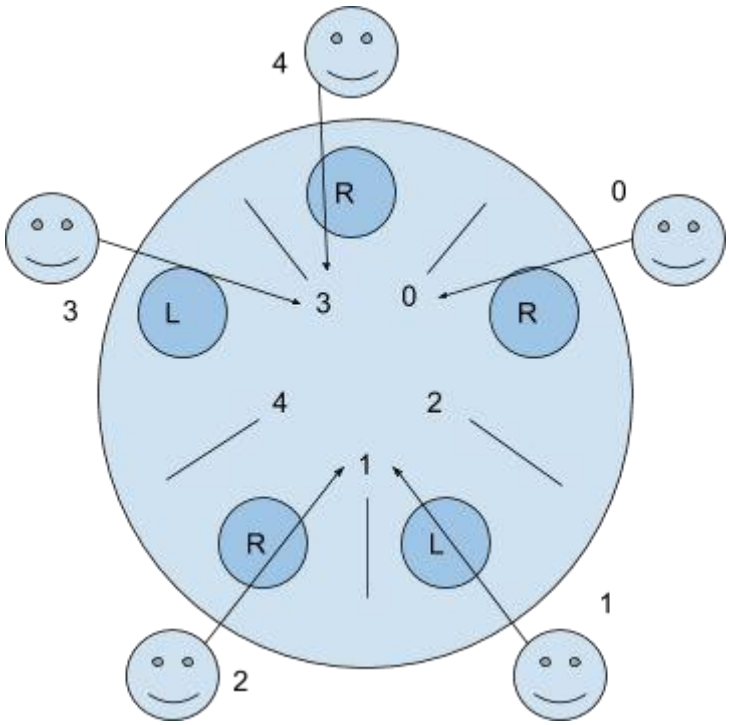
Considerando las características de ambos tipos de filósofos, podemos asegurar que el algoritmo es libre de hambruna.

Construcción de la solución

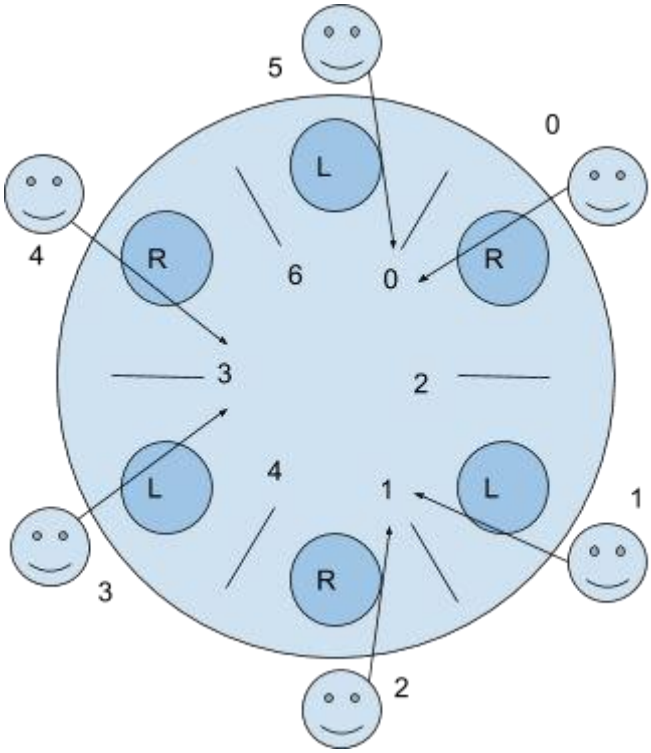
Considere uno de los tenedores involucrados en el problema. Consideremos que cualquier otro tenedor a una distancia i (en sentido horario) se enumerará como i -1 si i es par y como i +1 si i es impar; tomando siempre del tenedor 0 como referencia. De esta manera, sólo faltaría acomodar a los filósofos de manera que contiendan por los tenedores en orden ascendente de numeración

Note que habrá competencia 2 a 2 sobre los tenedores, y alguno de los dos filósofos será el ganador, mientras que el otro deberá esperar a que el ganador suelte el tenedor, pero por el acomodo entre zurdos y diestros, el segundo tenedor siempre estará disponible, no importa la competencia.

Así, podemos garantizar que esta solución satisface *libertad de abrazos mortales* y además *libertad de hambruna*.



Ejemplo de la solución con un número impar de filósofos. Note que siempre queda libre el segundo tenedor para el primer ganador y además se invierte el último filósofo: por la regla R par, deberá tratar de apoderarse del tenedor con identificador mayor.



Ejemplo con de la solución con un número par de filósofos. Puesto que el tiempo de duración del estado comer, se garantiza libertad de hambruna.

Notemos que en el peor caso de contendencia, los filósofos se harán esperar entre sí, por bloques contiguos de 4 en 4. Esto se debe a lo siguiente: representamos estos bloques contiguos de 4 filósofos en nuestro problema aplicando módulo 4 en su identificador i. En el peor caso, ocurre lo siguiente:

- i = 0 mod 4. i es par, por lo que es un filósofo tipo R. Supongamos que pasa a estado de hambre y logra apoderarse del tenedor a su derecha y posteriormente logra también ganar el tenedor a su izquierda y pasa a estado comiendo.
- i = 1 mod 4. i es impar, por lo que es un filósofo tipo L. Supongamos que también pasa a estado hambriento y toma el tenedor a su izquierda. Deberá esperar a que el filósofo i = 0 mod 4 termine de comer para poder usar el tenedor a su derecha, por lo que permanece en estado de hambriento.
- i = 3 mod 4. i es impar, por lo que es un filósofo tipo L. Supongamos pasa a estado hambriento y logra apoderarse del tenedor a su izquierda. Pero el filósofo i = 0 mod 4 está comiendo, por lo que mantiene ocupado el tenedor que requiere el filósofo i = 3 mod 4 para comer, por lo que permanece hambriento.
- i = 2 mod 4. i es par, por lo que es un filósofo tipo R. Supongamos que se vuelve hambriento, pero el filósofo i = 1 mod 4 se apoderó del tenedor a su derecha, por lo que permanece hambriento.

Notemos que si n no es múltiplo de 4, el último bloque de filósofos será de tamaño menor que 4, quedará trunco. La única consideración que hace falta tomar en esos casos, es que suponiendo este alto nivel de contundencia, sólo uno de ellos podrá empezar a comer y los otros quedaron esperándolo (De manera similar al bloque de 4. Se consideran bloques de tamaño 4 porque en bloques de tamaño mayor, se apreciaría más de un filósofo capaces de comer concurrentemente con los otros y precisamente queremos medir la concurrencia de esta solución).

Por lo tanto, la concurrencia del algoritmo LR es $\lceil n/4 \rceil$*-concurrente*.

A continuación, estudiaremos otra solución que es óptimo en cuanto a concurrencia para el problema de los filósofos comensales.

Algoritmo LLR

Esta solución, se le debe a S. P. Rana y D. K. Banerji y fue postulada en 1986. Es mejor que el algoritmo LR en términos de concurrencia.

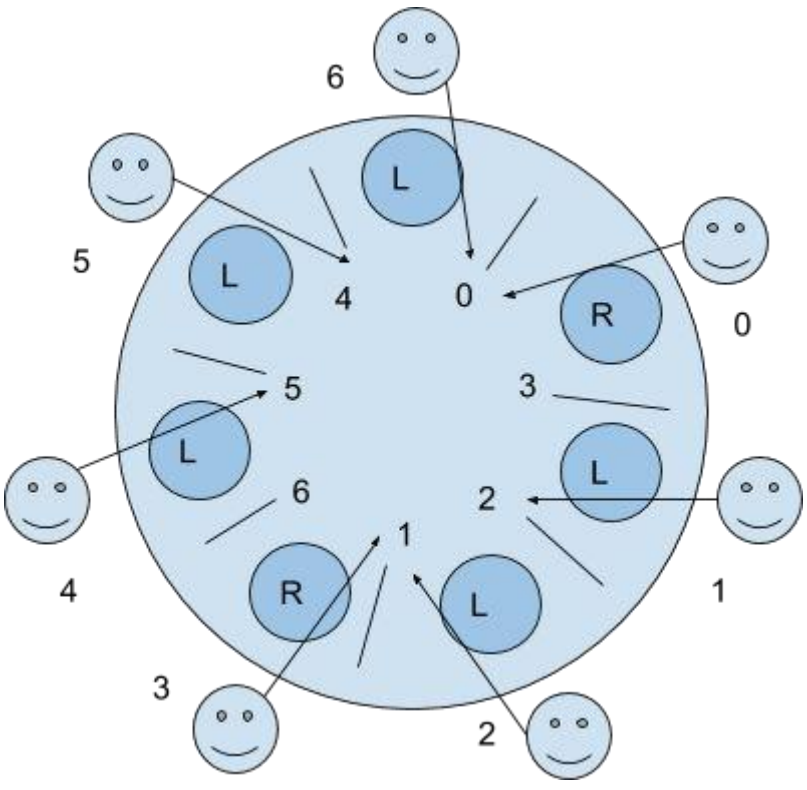
La solución consiste en distribuir filósofos zurdos (L) y diestros (R) de la siguiente manera: el filósofo i es tipo R si i es divisible entre 3. Es tipo L en otro caso.

Diseño de la solución

Vamos a enumerar los tenedores de manera similar al algoritmo LR: empezamos con el tenedor 0 y consideramos alguno otro a distancia i, la distancia también medida en sentido horario:

- Si i mod 3 es 1, entonces se numera ese tenedor como i +2.
- Si i mod 3 es 2, entonces se numera el tenedor cómo i.
- En otro caso, (i mod 3 es 0), se le asigna el identificador i -2.

Esto garantiza un orden total entre los tenedores, y los filósofos lo que hacen es tratar de apoderarse de sus tenedores en orden ascendente del valor numérico de los identificadores de los tenedores.



Notemos que, de manera análoga al algoritmo LR, la competencia de por los tenedores sólo es 2 a 2 cada 3 filósofos. Cada uno puede apoderarse del primer tenedor sin tanto problema y puesto que los tiempos de comida son finitos, en algún momento todos los filósofos tendrán la oportunidad de comer en algún momento durante la ejecución.

Por lo tanto, el algoritmo LLR es libre de hambruna y libre de abrazos mortales.

Ahora, considere una cadena de tres filósofos LLR, donde al menos alguno de ellos podrá empezar a comer (en el tiempo que estamos considerando y bajo las restricciones del problema). Notemos que esta cadena se repite cada 3 filósofos por construcción de la solución. Sean estos 3 filósofos p_0, p_1 y p_2. Supongamos que p_0 está hambriento y apenas va a competir por el tenedor f_3 (que tiene p_1).

Si p_1 no está compitiendo, entonces p_0 puede obtener su segundo tenedor y comenzar a comer. En caso contrario, supongamos que p_1 logra apoderarse del tenedor f_3 que comparte con p_0. Como p_1 es zurdo, tomar f_3 significa que posee ambos tenedores y puede empezar a comer.

Finalmente, está el caso en el que p_1 no haya podido empezar a comer por causa de p_2. Esto sólo puede pasar si p_2 le ganó el tenedor f_2 a p_1, pero puesto que p_2 es distro, eso significa que ya posee ambos tenedores y puede empezar a comer.

Sin importar cómo sean las combinaciones de estos tres filósofos en las posibles cadenas que nos permite construir el algoritmo LLR: LLR, LLL, RLL, LRL; siempre habrá al menos uno de esos tres que podrá empezar a comer.

Por lo tanto el algoritmo LLR es $\lfloor n/3 \rfloor$*-concurrente*. El algoritmo es óptimo como solución apropiativa al problema de los filósofos comensales.

Referencias

Gadi Taubenfeld. Synchronization algorithms and concurrent programming, Ed. Pearson Prentice Hall, Israel, pp 263 - 267.