Programación Concurrente

Práctica 5: Software Transactional Memory

1. Modelar usando variables transaccionales un buffer de capacidad 1 que provee las operaciones `put` y `get`. Se utilizarán variables de tipo `TVar (Maybe t)` para representar buffers con valores de tipo `t`.

   a) Definir las operaciones de manera tal que las mismas sean no bloqueantes, es decir, tienen el tipo:

   ```
   get :: TVar (Maybe t) -> STM (Maybe t)
   put :: TVar (Maybe t) -> t -> STM Bool
   ```

   b) Dar un programa que crea un buffer con valor inicial 0 y luego ejecute concurrentemente una operación de escritura del valor 1 y una de lectura. El programa debe mostrar por pantalla el resultado que se obtiene de la lectura del buffer.

   c) Dar una implementación de buffer donde las dos operaciones son bloqueantes, es decir, se debe esperar a que esté vacío para escribir y lleno para leer. Se espera que las operaciones tengan el siguiente tipo

   ```
   get :: TVar (Maybe b) -> STM b
   put :: TVar (Maybe t) -> t -> STM ()
   ```

   d) Definir una una operación `read` que permite leer (sin tomar) el contenido del buffer . Si el buffer está vacío, la operación se bloquea hasta tanto se agregue algún valor.

2. Implementar un semáforo en términos de STM. Mostrar un fragmento de código en el cual un cliente toma atómicamente dos permisos.

3. Se desea administrar la asignación de recursos, donde se debe mantener una cola de recursos de un determinado tipo y permite que se tomen y liberen a través de dos operaciones tomar (que devuelve el primer elemento de la cola) y liberar (que agrega a la cola el elemento que se libera).

   a) Dar una implementación utilizando STM.

   b) Mostrar como un proceso pueda tomar y liberar una cantidad arbitraria de recursos al mismo tiempo.

4. Considerar el juego de las bolitas, donde existen dos tipos de participantes: los *generadores* y los *consumidores* de bolitas. Los generadores crean bolitas de a una a la vez y las colocan en una bolsa (suman un punto por cada bolita que logran colocar en la bolsa). Los consumidores toman pares de bolitas y suman un punto por cada par (pero no pueden tomar mas de un par por vez). No se conoce a priori la cantidad de participantes que están jugando, pero se debe garantizar en todo momento que la bolsa no contenga mas bolitas que la cantidad de generadores que se encuentran jugando.

5. Dar una solución al problema de los fumadores. En el mismo hay tres procesos fumadores y un proceso proveedor. Cada uno de los procesos fumadores hará un cigarrillo y lo fumará. Para hacer un cigarrillo requiere tabaco, papel y fósforos. Cada proceso fumador tiene uno de los tres elementos. Es decir, un proceso tiene tabaco, otro tiene papel y el tercero tiene fósforos. El proveedor tiene disponibilidad no acotada de cualquiera de los elementos. Periódicamente coloca dos elementos sobre la mesa, y el fumador que tiene el tercer elemento los toma y fuma su cigarrillo.

6. Dar una solución al problema de los filósofos comensales.

   Ayuda: Puede modelar los tenedores como buffers (usando la resolución del ejercicio 1.b).

7. Considerar el problema de la pizzería. En el bar X se fabrican dos tipos de pizzas: las chicas y las grandes. El maestro pizzero va colocando de a una las pizzas listas en una barra para que los clientes se sirvan y pasen luego por la caja. Lamentablemente todos los clientes tienen buen apetito y se comportan de la siguiente manera: prefieren siempre tomar un pizza grande, pero si no lo logran se conforman con dos pequeñas. Los clientes son mal educados y no esperan en una cola (todos compiten por las pizzas). Dar una solución utilizando STM en donde cada cliente y el maestro pizzero son modelados como threads.

8. Implementar un tipo de datos Pila concurrente que permite la ejecución de las operaciones `push` y `pop`.