CI4251 - Programación Funcional Avanzada
Tareas 4 y 5

Ernesto Hernández-Novich
86-17791
<emhn@usb.ve>

Junio 4, 2010

1. Concurrencia (4 puntos)

El tipo de datos `Chan` está implantado aprovechando `MVar`. Use `MVar` para desarrollar una librería que provea el tipo de datos abstracto `BoundedChan` junto con un API idéntico al ofrecido por `Chan`.

La única diferencia entre ambos APIs es que la función `newBoundedChan` definida en su módulo recibe un parámetro adicional `Int` que indica el número máximo de items sin leer que pueden estar en el `BoundedChan`.

Si se alcanza dicho límite, las llamadas a `writeBoundedChan` deben bloquearse hasta que algún hilo use `readBoundedChan` para consumir algún valor.

Escriba un programa principal que demuestre la correcta operación de su librería combinando dos hilos: un productor "rápido" que utilice un `BoundedChan` para suministrar valores hacia un consumidor "lento".

2. Paralelismo (6 puntos)

El **colapso** de un número entero corresponde a la suma de sus dígitos, proceso que puede seguirse recursivamente hasta llegar a un dígito final. Por ejemplo, el colapso de 134957 es 29, que a su vez colapsa en 11 que finalmente colapsa en 2.

Definiremos el colapso de un arreglo de enteros como el colapso de la suma de los colapsos de los enteros que le componen.

Escriba una función no paralela

```
collapse :: [Int] -> Int
```

que calcule el colapso de un arreglo de enteros. Evalúe su desempeño trabajando sobre arreglos que tomen al menos dos minutos en ser procesados hasta obtener el resultado (¡aproveche `QuickCheck`!).

Ahora escriba

```
pcollapse :: [Int] -> Int
```

que efectúe el mismo cálculo pero empleando paralelismo implícito.

Compare el desempeño de las funciones `collapse` y `pcollapse` operando sobre un sólo núcleo. Luego, ajuste `pcollapse` para obtener el máximo desempeño posible sobre **dos** núcleos.

La solución que exhiba el mejor desempeño sobre **mi** máquina recibirá un (1) punto adicional.