Proyecto: Simulación de Ascensores

Conceptos y Modelos de Programación
Pontificia Universidad Javeriana
Cali - Valle

Prof.: Javier Andrés Mena Zapata

23 de julio de 2008

1. El escenario del ascensor

Asuma que se tiene un edificio con n ascensores y m pisos. Cuando un ascensor es llamado, se desearía atender la petición de una forma razonable: el ascensor será seleccionado de forma tal que pueda atender la petición tan pronto como sea posible.

Imaginemos que existe actualmente un edicio con tres ascensores ($n = 3$) y cinco pisos ($m = 5$). Los pisos uno y cinco tienen un único botón de llamada, los pisos restantes tienen dos botones de llamada, cada uno para *abajo* y *arriba* respectivamente.

Es posible pedir un ascensor presionando uno de los botones. Un petición es atendida seleccionando a un ascensor para que se detenga en el piso. Además, cada ascensor tiene 5 botones (o, en general m botones) adentro, numerados con los números de los pisos. Estos botones detendrán el ascensor en el piso apropiado.

La tarea consiste en escribir un programa que controle el sistema de ascensores. Se construirá una solución que modele los ascensores y pisos por medio de agentes que reaccionan a los mensajes entrante y posiblemente envían mensajes a otros agentes.

Es bien sabido que una buena práctica es modelar cada agente del problema por un programa de agente.

2. Planificando Ascensores

Soponga que el botón *arriba* es presionado en el cuarto piso. Es necesario decidir cual ascensor debería parar en el cuarto piso. Para ello se usará el siguiente algoritmo:

- El agente para el cuarto piso envía un mensaje a cada uno de los agentes de los ascensores, preguntando cuanto tardaría llegar al piso 4. También le

dice a los ascensores que existe una persona en el piso 4 que quiere subir.

- Cada ascensor que recibe este mensaje calcula cuanto tiempo le tomaría llegar al piso 4 y contesta con un mensaje que contiene el *tiempo de espera*.
- El agente del piso selecciona el ascensor con el mínimo tiempo del espera propuesto y responde con un mensaje de `reservar`. Todos los demás ascensores son informados por un mensaje de `rechazar`.

Se asume que un agente de ascensor está bloqueado después de que ha enviado una estimación de tiempo, y se mantiene bloqueado hasta que recibe un mensaje de `reservar` o `rechazar`.

Esta estrategia funciona y evita a bloqueos, en caso de que se todos los mensajes de petición enviados a los ascensores se reciban en el mismo orden.

3. Modelamiento

Se modelará el escenario con tres diferentes tipos de agente, los cuales incluyen los agentes para el ascensor y los pisos mencionados anteriormente, junto con agentes que modelarán la cabina del ascensor.

El agente de piso. El único mensaje que un agente de piso recibe es que se ha presionado un botón. Esto se modela, ya sea por un mensaje `boton(arriba)` o `boton(abajo)` dependiendo de cual botón se haya presionado. Cuando se recibe un mensaje que indique que un botón se ha presionado, el mejor ascensor es seleccionado como se describió anteriormente.

El estado de un agente de de piso contiene su número de piso y una colección de todos agentes de ascensor. El estado está codificado como un registro:

```
estado(piso:I ascensores:As)
```

donde `I` es el número de piso y `As` es la lista de agentes de ascensor.

El agente de ascensor. Un agente de ascensor entiende los siguientes mensajes:

- `peticion(piso:I dir:D respuesta:R)` el cual es enviado por el agente de piso para el piso `I`. `D` puede ser `arriba` o `abajo` dependiendo del botón presionado en el piso `I`.

  La respuesta `A` es una variable no ligada. La agente de ascensor liga la variable a un registro `propuesta(tiempo:T estado:S)` donde `T` es el *tiempoespera*. `S` es una variable no ligada.

  La variable no ligada `S` es ligada en su momento, ya sea a `reservar` o `rechazar` por el agente de piso. El mejor agente de ascensor recibe un `reservar` mientras que los demás obtienen un `rechazar`. Si el agente recibe un `reservar`, este debe incluir el piso número `I` en los pisos los cuales debe visitar (la lista de pisos a ser visitados es llamada *listaparadas*).

- `llegada(piso:I accion:A)` es enviado por el agente de cabina cuando la cabina ha llegado al piso `I`. El agente de ascensor ahora debe determinar qué acción tomará la cabina, ligado la variable `A` así:

  1. `parar`: la cabina abrirá y cerrará sus puertas. Esto toma 5 unidades de tiempo.
  2. `arriba`: la cabina se moverá un piso hacia arriba. Esto toma 1 unidad de tiempo.
  3. `abajo`: la cabina se moverá un piso hacia abajo. Esto toma 1 unidad de tiempo.
  4. `esperar`: la cabina se queda donde se encuentra por una unidad de tiempo.

  Después de que la cabina ha ejecutado su acción, esta envía de nuevo un mensaje de tipo `llegada` al gente del ascensor

- `parar(piso:I)` le pide al ascensor que se detenga en el piso `I` (este mensaje viene desde los botones internos del ascensor).

El estado del agente del ascensor es un registro `estado(piso:I paradas:S)` donde `I` es el piso en el cual el agente de ascensor se encuentra actualmente (esto es conocido a partir de los mensajes enviados por el agente de cabina) y `S` es la lista de paradas.

La lista de paradas es na lista de enteros describiendo en cuáles pisos el agente de ascensor debe detenerse. Inicialmente la lista está vacía. Los números de pisos son insertados en la lista de paradas, cuando un agente de ascensor ha recibido un mensaje de `peticion` y subsecuentemente también ha sido confirmado que su oferta ha sido la mejor.

El cálculo de la lista de paradas está íntimamente relacionado con el cálculo de tiempo de espera. Dependiendo de donde sería insertado el piso en la lista de espera, se calcula el tiempo de espera.

4. Insertando pisos en la lista de espera

Implemente una función {`Insertar ListaParadas Ahora Dir Piso`} que retorne una nueva lista de paradas. Aquí `ListaParadas` es la lista de paradas actual, `Ahora` es el piso en el cual el agente de ascensor se encuentra actualmente, `Dir` es bien sea `arriba` o `abajo` dependiendo de cual botón ha sido presionado para el piso `Piso`.

Es importante computar una nueva lista de paradas que minimice el tiempo hasta que el ascensor llegue hasta al piso deseado. Tome en cuenta los siguientes aspectos:

- `Piso` ya está contenido la lista de paradas.
- El ascensor ya está en el piso `Piso`.

- Si el ascensor se mueve desde el piso `A` hasta el `B` con `A<Piso<B` y `Dir` es `arriba`, la inserción debe hacerse entre `A` y `B`.
- Si el ascensor se mueve hacia abajo desde el piso `A` hasta el `B` con `A>Piso>B` y `Dir` es `abajo`, la inserción debe hacerse entre `A` y `B`.
- La inserción debe hacerse tan cerca del frente de la lista como sea posible, mientras se sigan las reglas anteriores.

Por ejemplo, {`Insertar nil 3 up 6`} retorna `[6]`
{`Insertar [5 7 9 3] 4 arriba 6`}, retorna `[5 6 7 9 3]`
{`Insertar [5 7 9 3] 4 abajo 6`, retorna `[5 7 9 6 3]`.
Almacene este programa en un archivo llamado `Insertar.oz`.

5. Computando el tiempo de espera

Implemente una función {`TiempoEspera ListaParadas Ahora Dir Piso`} que retorna el tiempo de espera antes de que una petición pueda ser servida. Aquí `ListaParadas` es la lista actual de paradas, `Ahora` es donde el ascensor se encuentra actualmente, `Dir` es bien sea `arriba` o `abajo` dependiendo de cual botón se haya presionado en el piso `Piso`.

Como se mencionó anteriormente, detener el ascensor toma cinco unidades de tiempo, mientras que mover el ascensor un piso, toma una unidad de tiempo. El tiempo de espera debe ser cumputado con las mismas reglas que tiene la inserción en la lista de paradas

{`TiempoEspera [5 7 9 3] 4 arriba 6`}, retorna `7`.

{`TiempoEspera [5 7 9 3] 4 abajo 6`}, retorna `23`. Pues se debe detener en los piso 5, 7 y 9, y en total recorre 8 pisos.

Almacene este programa en un archivo llamado `TiempoEspera.oz`.

6. Presionado un botón del ascensor

Implemente una función {`PararEn ListaParadas Ahora Piso`} que retorne una nueva lista de paradas cuando una un botón para `Piso` se haya presionado mientras se encuentre dentro del ascensor. Aquí `ListaParadas` es la lista actual de paradas, y `Ahora` es el piso el cual el agente de ascensor se encuentra actualmente.

Siga las mismas reglas para insertar `Piso` en la lista de paradas que en `Insertar`.

Almacene este programa en un archivo llamado `PararEn.oz`.

7. El agente de ascensor

Realice el diagrama de estados en donde se muestre todos los posibles estados y mensajes que envía el agente de ascensor.

Implemente una función {`ProcesoElevador Estado Msg`} para el agente de ascensor la cual implemente el diagrama de estados propuesto para el proceso del agente de elevador.

Implemente la siguiente función

```
fun {NewLift N}
   %% Crea un el nuevo ascensor, número N en el piso 1,
   %% con una lista de paradas vacía
   {NewAgent ProcesoElevador estado(piso:1 paradas:nil)}
end
```

Almacene este programa en un archivo llamado `Ascensor.oz`.

8. El agente de piso

Realice el diagrama de estados en donde se muestre todos los posibles estados y mensajes que envía el agente de piso.

Implemente una función {`ProcesoPiso Estado Msg`} para el agente de piso la cual implemente el diagrama de estados propuesto para el proceso del agente de piso.

Implemente la siguiente función

```
fun {NewFloor N Ascensores}
   %% Crea un agente de piso en el piso N
   {NewAgent ProcesoPiso
    estado(piso:N ascensores:Ls)}
end
```

Almacene este programa en un archivo llamado `Piso.oz`.

9. Colocando todo junto

Se proveen los archivos `Main.oz` y `Cabina.oz` desde la página del curso. Coloque todos los archivos en un mismo directorio y ejecute `Main.oz`, el cual provee una simulación completa del ascensores para un número dado de ascensores y pisos.

Usted puede simular que se presionan los botones enviando los mensajes apropiados a los agentes de elevador o de piso, de forma apropiada.

10. Reconocimiento

Este problema ha sido propuesto por Seif Haridi para el curso CS2104 Conceptos de Lenguajes de Programación.

The problem has been taken from: Joe Armstrong, Robert Virding, Claes WikstrÄom, and Mike Williams, *Concurrent Programming in Erlang*, second edition. Prentice Hall, London, UK, 1996.

11. Especificación del taller

Se debe entregar el taller en grupos de máximo 2 personas. Fecha: Julio 30 de 2008. Debe entregar informe con diagramas de estado y los archivos pedidos, todo comprimido en un archivo ZIP o TGZ. **NO SE ACEPTAN ARCHIVOS RAR**.

Enviar a `javimena@gmail.com`