Examen 3

Cruz Perez Ramón
315008148

January 9, 2021

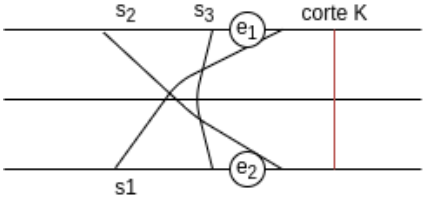
1. **Demuestra que solo hay un corte consistente maximal que precede a cualquier corte.**

   Por contradiccion:
   Sup. que existen dos cortes maximales S_1 y S_2.

   - Caso 1: Los cortes no se cruzan, en cuyo caso S_2 es más reciente:
     Sup. que S_1 esta muy cerca del corte k, entonces por ser maximal es el mas cercano, pero como S_2 tambien es maximal, entonces uno de los dos debe ser el mas cercano.
     Si, S_2 es el más cercano que S_1, entonces S_2 el maximal. ▼
     Si, es el caso que estan al misma distancia, entonces $S_1 = S_2$ ▼
     $\Longrightarrow$ Existe un unico maximal.

   - Caso 2: Si se cruzan, hay un envio de un mensaje despues de S_1:
     Como sabemos que cruzan, significa hay un evento$_1 \in S_1$ y evento$_1 \notin S_2$, y uno evento$_2 \in S_2$ y evento$_1 \notin S_1$
     Sup. que existe un S_3 que cubre esos dos eventos y es maximal, como cubre los dos eventos que y ademas es maximal significa que el S_3 es el unico maximal. ▼

     $\Longrightarrow$ Existe un unico maximal.



2. **Di si es verdadera o falsa la afirmacion dando una pequena justificion:**

(a) Los detectores de fallas no cumplen las propiedades en sistemas asincronos:

Falso. De hecho los detectores fueron implentados para ayudar a resolver el consenso en sistemas asincronos. Pues deben cumplir integridad y precision.

(b) Los detectores de fallas ayudan a resolver el consenso en cualquier sistema

Falso, cada detector cumple con ciertas propidades que es posible que no puedan cumplir en algunos sistemas asincronos. Todo va a depender de los niveles de integridad y precision.

(c) Un solo proceso sospecha de todos los procesos fallidos es la propiedad de integridad fuerte

Falso.
De hecho es integridad debil.
Integridad fuerte: Todo proceso fallido es Eventualmente sospechado por Todo proceso correcto.

(d) Cuando se tiene un detector de fallas, todos los procesos deben tener la misma informacion de fallas todo el tiempo.

Falso. Pues cuando un detector empieza a funcionar correctamente, regularmente suele pasar un tiempo en propagarse la informacion.
Todo depende la integridad, si es fuerte todos tienen la misma informacion. Pero si es debil no, por que solo uno tiene la informacion, pero con un algorimo podemos hacer que se vuelva fuerte.

3. **Sea T el algoritmo para transformar cualquier detector de fallas que cumpla integridad debil (eventualmente un proceso correcto sospecha de todos los procesos fallidos) en uno que cumple integridad fuerte (eventualmente todos los procesos fallidos estan bajo sospecha por cualquier proceso correcto). Demostrar que el Algoritmo 1 satisface precision perpetua y eventual.**

Como el detector cumple Integridad debil. Solo un proceso correcto va a tener la informacion de los fallos, SPG. sea P dicho detector.

- **Fuerte**:
  Sup que el detector cumple precision fuerte. $\implies$ P enviara a sus

sospechosos cuando el detector le de la informacion, i.e P no enviara inf. hasta D_p sea llenado.
Analogamente los demas procesos no enviaran nada, pues el detector solo da inf. a un proceso P.
Entonces, ningun proceso esta bajo sospecha antes de fallar.
Entonces, cada proceso p_i que reciba los sospechosos del P unira su output $= \emptyset$ con los del P .
Por lo tanto, ningun proceso esta bajo sospecha antes de fallar.

- **Debil**:
  Sup que el detector cumple precision debil. $\Longrightarrow$ Como simpre existe algun proceso correcto que nunca esta bajo sospecha.
  Entonces, cuando el detector le de inf. a P sabremos que el proceso que nunca estubo bajo sospecha no esta en D_p, por precicion debil.
  Por lo tanto. Algun proceso correcto nunca esta bajo sospecha, durante la ejecucion de T.

- **Eventualmente fuerte**:
  Sup que el detector cumple precision eventualmente fuerte. $\Longrightarrow$ Dado cierto tiempo Todos los procesos correctos nunca estan bajo sospecha por ningun proceso correcto.
  Entonces, cuando el detector le de inf. a P, sabremos que ya habra pasado un tiempo y todos los procesos correctos no estan bajo sospecha y $supects_p$ sera enviado.
  Por lo tanto, hay un tiempo despues del cual TODOS los procesos correctos nunca estan bajo sospecha por ningun proceso correcto, en T.

- **Eventualmente debil**:
  Sup que el detector cumple precision eventualmente debil. $\Longrightarrow$ Dado un cieto tiempo, algun proceso correcto no estan bajo sospecha por ningun proceso correcto.
  Entonces, cuando el detector le de inf. a P, sabremos que ya habra pasado un tiempo y algun proceso correcto no esta bajo sospecha y $supects_p$ sera enviado.
  Por lo tanto, Hay un tiempo despues del cual algun proceso correxto no estan bajo sospecha por ningun proceso correcto, en T.

4. **Demuestra que el Algoritmo 2 es una implementacion de <>P. Para demostrar que es una implementacion de <>P puedes usar los siguientes lemas.**

   - **Integridad fuerte**:(Todo proceso fallido es Eventualmente sospechado por Todo proceso correcto)

Si, el *timeout* expira se vuelve sospechoso, Linea 27-33.
Entonces, si en algun momento p_i = (proceso sospechoso), deja de mandar mensajes. El *counter* deja de aumentar por que no se cambiaria el *supect* en las Lineas 17-23.

Por lo que, en cierto tiempo los procesos correctos empiezan a recibir el $counter[i]$ sin actulizar y el *timerout* expira o no se actulizaria.
Por lo tanto, todo proceso fallido es eventualmente sospechado por todo proceso carrecto.

- **Eventualmente fuerte**:(Hay un tiempo despues del cual Todos procesos correctos nunca estan bajo sospecha por ningun proceso correcto)
  Para cualesquiera dos procesos p_i, p_j.
  Como p_i aumentara $counter[i]$ cada T tiempo (lemma 3). Nunca se sospechara de p_i, pues siempre llegara a p_j, si se llegara a sospechar como $counter[i]$ es mayor, se quitaria esa sospecha (lemma 4).
  Por el lemma 1 y 2, eventualmente se dejara de sospechar de los procesos correctos.