EXAMEN DE VALIDACIÓN Y VERIFICACIÓN DEL SOFTWARE. 1/7/2019.

APELLIDOS Y NOMBRE: .....

1) **(70 pts)** Un acusado puede ir a juicio j y, como consecuencia, conocer una sentencia s a la que puede recurrir r. Formula los siguientes enunciados en LTL

   – No puede ser sentenciado sin un juicio previo

   Es decir, no puede ser que nunca haya habido juicio y nos encontremos con una sentencia

   $$\neg(\neg j \ \mathcal{U} \ s)$$

   Esta fórmula es equivalente a:

   $$j \ \mathcal{V} \ \neg s$$

   que usa el operador "release" y dice que hay un estado que cumple j anterior a cada estado que no cumpla $\neg s$ (es decir, que cumpla s).

   Otra solución interesante que han encontrado algunos alumnos es usando "weak until":

   $$\neg s \ \mathcal{W} \ j$$

   Para entender mejor el efecto, lo podemos desarrollar como:

   $$\begin{aligned} \neg s \ \mathcal{W} \ j &\equiv \Box\neg s \vee (\neg s \ \mathcal{U} \ j) \\ &\equiv \neg\Diamond \vee (\neg s \ \mathcal{U} \ j) \\ &\equiv \Diamond s \rightarrow (\neg s \ \mathcal{U} \ j) \end{aligned}$$

   que dice que, si en algún momento futuro se va a dictar sentencia, entonces nos encontraremos con un juicio j sin que aparezca sentencia antes de j. Esta solución admite que ese juicio y la sentencia sean simultáneas. Si queremos obligar a que la sentencia sea estrictamente posterior (el juicio estrictamente anterior), habría que ajustarlo como:

   $$\Diamond s \rightarrow (\neg s \ \mathcal{U} \ (j \wedge \neg s)) \equiv \neg s \ \mathcal{W} \ (\neg s \wedge j)$$

   que, en realidad, ya es equivalente a $j \ \mathcal{V} \ \neg s$ (ver equivalencias en transparencias de tema 2).

   – Si va a juicio, finalmente recibe una sentencia

   $$\Box(j \rightarrow \Diamond s)$$

   – A partir de un momento, ya no se puede recurrir más

   $$\Diamond\Box\neg r$$

   o también $\neg\Box\Diamond r$ (no se puede recurrir un número infinito de veces)

– No puede ser juzgado dos veces sin recurso de por medio

Una forma sencilla de representarlo es

$$\Box(j \rightarrow \bigcirc(\neg j \ \mathcal{W} \ (r \wedge \neg j)))$$

que viene a decir que, si es juzgado en un momento dado, entonces, o bien no vuelve a ser juzgado, o si no, habrá obligatoriamente un recurso sin que vuelva a ser juzgado antes. El uso de $\bigcirc$ es necesario porque, de lo contrario, tendríamos que la condición $\neg j$ del $\mathcal{W}$ comenzaría siendo cierta en el estado actual, y el antecedente de la implicación supone que ese estado cumple j. La fórmula de arriba es equivalente a

$$\Box(j \rightarrow \bigcirc(r \ \mathcal{V} \ \neg j))$$

que dice que, si se da j, entonces después hay un r anterior a todo estado que vuelva a cumplir j.

Otra forma alternativa de representarlo es:

$$\Box(j \wedge \bigcirc\Diamond j \rightarrow \bigcirc(\neg j \ \mathcal{U}(r \wedge \neg j)))$$

Si tengo dos juicios, entonces, a partir del siguiente estado, aparece un recurso sin que se haya dado juicio.

– Sólo se puede recurrir, si hay sentencia previa y no recurrida

Garantizar que el primer recurso tiene una sentencia previa es sencillo. Basta con prohibir que aparezca un recurso sin que antes haya habido sentencia, esto es, $\neg(\neg s \,\mathcal{U}\, r)$ o, lo que es equivalente, $s \ \mathcal{V} \ \neg r$. Ahora bien, esto hay que garantizarlo para los sucesivos recursos que vayan apareciendo. Es decir, si en el estado actual hubo un nuevo recurso (con sentencia previa), entonces el siguiente recurso debe tener de nuevo una sentencia previa, y así sucesivamente. Esto podemos contrarlo añadiendo la implicación $r \rightarrow \bigcirc(s \ \mathcal{V} \ \neg r)$ en todo estado de la trayectoria. En resumen, tendríamos la fórmula:

$$(s \ \mathcal{V} \ \neg r) \wedge \Box(r \rightarrow \bigcirc(s \ \mathcal{V} \ \neg r))$$

2) **(50 pts)** Probar si la siguientes fórmulas son equivalentes o encontrar un contraejemplo:

$$\alpha \stackrel{def}{=} \Diamond(p \rightarrow q) \qquad\qquad \beta \stackrel{def}{=} \neg\Box p \vee \Diamond q$$

Se puede probar por sucesivas transformaciones que mantienen la equivalencia:

$$\begin{array}{lcll} \Diamond(p \rightarrow q) & \equiv & \Diamond(\neg p \vee q) & \text{Propiedad de la implicación} \\ & \equiv & \Diamond\neg p \ \vee \ \Diamond q & \text{distributiva de } \Diamond \text{ frente a } \vee \\ & \equiv & \neg\Box p \ \vee \ \Diamond q & \text{De Morgan entre } \Diamond \text{ y } \Box \end{array}$$

3) **(20 pts)** Tenemos un programa en PROMELA llamado `ejemplo.pml` y ejecutamos el comando `spin ejemplo.pml` que termina imprimiendo el mensaje `timeout`. Explica que significa este mensaje.

Aunque parezca sugerirlo, este mensaje *no* significa que exista un tiempo prefijado de espera que se haya agotado. Lo que indica es que, durante la simulación, existe uno o más procesos activos (que no han alcanzado un final de ejecución) y que en ninguno de ellos existe una instrucción habilitada para ejecutar a continuación. Esto es debido a que todos los procesos activos se encuentran esperando a que alguna condición se vuelva cierta, pero ninguno de ellos puede ya modificar el estado de las variables: es decir, es un abrazo mortal o *deadlock*.