

**EXAMEN DE VALIDACIÓN Y VERIFICACIÓN DEL SOFTWARE. 5/7/2021.**

**APELLIDOS Y NOMBRE:** .....

1) **(70 pts)** Un cronómetro cuenta con un botón ( $b$ ) de inicio/parada. El cronómetro puede estar parado  $p$  o en marcha  $\neg p$ . Además, el tiempo puede estar a cero 00:00:00 ( $c$ ) o no. Formula los siguientes enunciados en LTL

- Cuando pulsamos el botón, si estaba parado, se pone en marcha y viceversa, a no ser que lo mantengamos pulsado justo después.
- Si mantenemos pulsado el botón dos situaciones seguidas, se para y se pone a cero inmediatamente después
- Si no se pulsa el botón, el reloj permanece en el mismo estado (parado/en marcha) de un momento al siguiente
- El reloj pasa a cero un número infinito de veces
- Nunca pulsamos el botón indefinidamente

2) **(50 pts)**Dadas las fórmulas:

$$\alpha \stackrel{def}{=} \neg p \mathcal{U} \square p \qquad \beta \stackrel{def}{=} \diamond \square p$$

demostrar cada dirección de la equivalencia o, si no se cumple, presentar un contraejemplo

$\models \alpha \rightarrow \beta$  ¿se cumple? [ ]-Sí [ ]-No

Explicación:

$\models \beta \rightarrow \alpha$  ¿se cumple? [ ]-Sí [ ]-No

Explicación:

3) **(20 pts)** Explica brevemente en qué consiste la técnica de reducción de orden parcial (partial order reduction).

### Satisfaction of a temporal formula

Let  $M = s_0, s_1, \dots$  with  $i \geq 0$ . We say that  $M, i \models \alpha$  when:

- $M, i \models p$  if  $p \in s_i$  (for  $p \in \Sigma$ )
- $M, i \models \Box\alpha$  if  $M, j \models \alpha$  for all  $j \geq i$
- $M, i \models \Diamond\alpha$  if  $M, j \models \alpha$  for some  $j \geq i$
- $M, i \models \bigcirc\alpha$  if  $M, i + 1 \models \alpha$
- $M, i \models \alpha \mathcal{U} \beta$  if there exists  $n \geq i$ ,  $M, n \models \beta$  and  $M, j \models \alpha$  for all  $i \leq j < n$ .
- $M, i \models \alpha \mathcal{W} \beta$  if  $M, i \models \Box\alpha$  or  $M, i \models \alpha \mathcal{U} \beta$

### Kamp's translation

Temporal formula  $\alpha$  at time point  $i$  becomes  $MFO(<)$  formula  $\alpha(i)$

$$\begin{aligned} (p)(i) &\stackrel{def}{=} p(i) \\ (\neg\alpha)(i) &\stackrel{def}{=} \neg\alpha(i) \\ (\alpha \vee \beta)(i) &\stackrel{def}{=} \alpha(i) \vee \beta(i) \\ (\alpha \wedge \beta)(i) &\stackrel{def}{=} \alpha(i) \wedge \beta(i) \\ (\bigcirc\alpha)(i) &\stackrel{def}{=} \alpha(i + 1) \\ (\Diamond\alpha)(i) &\stackrel{def}{=} \exists j \geq i : \alpha(j) \\ (\Box\alpha)(i) &\stackrel{def}{=} \forall j \geq i : \alpha(j) \\ (\alpha \mathcal{U} \beta)(i) &\stackrel{def}{=} \exists j \geq i : (\beta(j) \wedge (\forall k \in i..j - 1 : \alpha(k))) \\ (\alpha \mathcal{V} \beta)(i) &\stackrel{def}{=} \forall j \geq i : (\beta(j) \vee (\exists k \in i..j - 1 : \alpha(k))) \end{aligned}$$

- **(2 puntos del total de la asignatura):** (equivale a práctica model checking) El siguiente código PROMELA:

```
/* Fichero ejemplo.pml */
bool wantP=false, wantQ=false;

active proctype P() {
  do
    :: printf("Non-critical section P\n");
    wantP=true;
    !wantQ;
  cs: printf("Critical section P\n");
    wantP=false;
  od
}

active proctype Q() {
  do
    :: printf("Non-critical section Q\n");
    wantQ=true;
    !wantP;
  cs: printf("Critical section Q\n");
    wantQ=false;
  od
}
```

pretende proteger con exclusión mutua el acceso a la sección crítica de dos procesos concurrentes P, Q.

1. Completa el siguiente comando para chequear la exclusión mutua en SPIN escribiendo la fórmula correspondiente en el hueco subrayado:

```
spin -a -f '_____ ' ejemplo.pml
```

2. Tras ejecutar (correctamente) el comando de arriba se comprueba que no hay error de exclusión mutua, pero sí de interbloqueo. Explica brevemente cómo puede darse dicha situación con un ejemplo de ejecución entrelazada que lo provoque.

¿Cómo arreglarías este problema sin perder la exclusión mutua?