Trabajo práctico N° 4

Abril 2025

Estudiante Emanuel Nicolás Herrador

Ejercicio 2

Veamos cada una por separado. Comencemos con $\Box\Box\varphi \to \Box\varphi$. Notemos que por definición tenemos lo siguiente:

$$\begin{aligned}
\sigma \vDash \Box\Box\varphi &\iff \forall j \geq 0 : \sigma[j..] \vDash \Box\varphi \\
&\iff \forall j \geq 0 : \forall j_2 \geq 0 : \sigma[j..][j_2..] \vDash \varphi \\
&\iff \forall i \geq 0 : \sigma[i..] \vDash \varphi \\
&\iff \sigma \vDash \Box\varphi
\end{aligned}$$

Luego, es trivial que se verifica la implicación porque son fórmulas equivalentes.

Para $\Box(\varphi \wedge \psi) \to \Box\varphi \wedge \Box\psi$ veamos que:

$$\begin{aligned}
\sigma \vDash \Box(\varphi \wedge \psi) &\iff \forall j \geq 0 : \sigma[j..] \vDash \varphi \wedge \psi \\
&\iff \forall j \geq 0 : \sigma[j..] \vDash \varphi \wedge \sigma[j..] \vDash \psi \\
&\iff (\forall j \geq 0 : \sigma[j..] \vDash \varphi) \wedge (\forall j_2 \geq 0 : \sigma[j_2..] \vDash \psi) \\
&\iff \sigma \vDash \Box\varphi \wedge \sigma \vDash \Box\psi \\
&\iff \sigma \vDash \Box\varphi \wedge \Box\psi
\end{aligned}$$

Con ello, resulta trivial la implicación al ser fórmulas equivalentes.

Y, por último, queda ver que $\Diamond(\varphi \vee \psi) \to \Diamond\varphi \vee \Diamond\psi$:

$$\begin{aligned}
\sigma \vDash \Diamond(\varphi \vee \psi) &\iff \exists j \geq 0 : \sigma[j..] \vDash (\varphi \vee \psi) \\
&\iff \exists j \geq 0 : \sigma[j..] \vDash \varphi \vee \sigma[j..] \vDash \psi \\
&\iff (\exists j \geq 0 : \sigma[j..] \vDash \varphi) \vee (\exists j_2 \geq 0 : \sigma[j_2..] \vDash \psi) \\
&\iff \sigma \vDash \Diamond\varphi \vee \sigma \vDash \Diamond\psi \\
&\iff \sigma \vDash (\Diamond\varphi \vee \Diamond\psi)
\end{aligned}$$

Luego, al ser equivalentes, claramente se cumple la implicación.

Ejercicio 3

Ver la validez de la fórmula equivale a ver si se cumple la equivalencia dada por $\Box(p \wedge \Diamond q) \equiv \Box(p\mathrm{U}q)$. Para ello, una forma de refutar la validez es mostrar una traza que cumple una propiedad pero la otra no. Veamos que $(\{q\})^\omega$ cumple claramente $\Box(p\mathrm{U}q)$ pero no $\Box(p \wedge \Diamond q)$. Luego, se refuta. ■

Ejercicio 4

Se pretende demostrar formalmente que *strong fairness* implica *weak fairness*. Es decir, es quiere ver que $(\Box\Diamond\varphi \to \Box\Diamond\psi) \to (\Diamond\Box\varphi \to \Box\Diamond\psi)$.

Para ello, expandamos cada una de las partes:

$$\begin{aligned}
\Box\Diamond\varphi \to \Box\Diamond\psi &\equiv \neg\Box\Diamond\varphi \vee \Box\Diamond\psi \\
\Diamond\Box\varphi \to \Box\Diamond\psi &\equiv \neg\Diamond\Box\varphi \vee \Box\Diamond\psi
\end{aligned}$$

Ahora, supongamos que *strong fairness* vale para un σ dado que $\sigma \vDash \Box\Diamond\psi$. Luego, es claro que, por la anterior expansión, también vale *weak fairness* para σ.

En caso de que valiera *strong fairness* pero solo porque $\sigma \vDash \neg\Box\Diamond\varphi$, podemos notar lo siguiente:

$$\begin{aligned}
\sigma \vDash \neg\Box\Diamond\varphi &\iff \neg(\forall i \geq 0 : \exists j \geq 0 : \sigma[i..][j..] \vDash \varphi) \\
&\iff \exists i \geq 0 : \forall j \geq 0 : \sigma[i..][j..] \vDash \neg\varphi \\
&\iff \exists i \geq 0 : \forall j \geq i : \sigma[j..] \vDash \neg\varphi
\end{aligned} \tag{4.1}$$

Ahora, nuestro objetivo es llegar a $\sigma \vDash \neg\Diamond\Box\varphi$, lo cual significa que:

$$\begin{aligned} \sigma \vDash \neg\Diamond\Box\varphi &\iff \neg(\exists i \geq 0 : \forall j \geq 0 : \sigma[i..][j..] \vDash \neg\varphi) \\ &\iff \forall i \geq 0 : \exists j \geq 0 : \sigma[i..][j..] \vDash \neg\varphi \\ &\iff \forall i \geq 0 : \exists j \geq i : \sigma[j..] \vDash \neg\varphi \end{aligned} \tag{4.2}$$

Si $\sigma \vDash \neg\Box\Diamond\varphi$, entonces por (4.1) es de la forma $(\emptyset + \{\varphi\} + \{\psi\} + \{\varphi, \psi\})^*(\emptyset + \{\psi\})^\omega$. Con ello, claramente $\sigma \vDash \neg\Diamond\Box\varphi$ por (4.2) porque sea j el índice más chico a partir del cual todo elemento de la traza cumple $\neg\varphi$, entonces $\forall 0 \leq i \leq j : \sigma[j..] \vDash \neg\varphi$ y $\forall j \leq i : \sigma[i..] \vDash \neg\varphi$.

Por esto, se demuestra por completo que *strong fairness* implica *weak fairness* dado que si una traza satisface la primera, entonces también satisfará la segunda. ■

Ejercicio 5

Veamos cada una:

- "Si φ es cierto durante la ejecución del programa, eventualmente ocurrirá ψ": $\Box(\varphi \to \Diamond\psi)$.
- "No es posible que φ y ψ ocurran simultáneamente durante la ejecución del programa": $\neg\Diamond(\varphi \wedge \psi)$.
- "Cada vez que φ sea cierto, ψ también lo es": $\Box(\varphi \to \psi)$.
- "φ no deja de ocurrir, al menos hasta que ψ ocurra": $\varphi \mathrm{WU} \psi \equiv \Box\varphi \vee (\varphi \mathrm{U} \psi)$.

Ejercicio 6

Se define el operador *while* como:

$$\sigma \vDash \varphi \mathrm{W} \psi \iff \forall i \geq 0 : (\forall 0 \leq j \leq i : \sigma[j..] \vDash \psi) \Rightarrow \sigma[i..] \vDash \varphi \tag{6.1}$$

Para poder resolver los siguientes items de este ejercicio, va a ser útil desglosar la definición para entenderla a detalle. Tomemos la definición de (6.1) y digamos que tenemos un x tal que $\forall 0 \leq j \leq x : \sigma[j..] \vDash \psi$. Ahora, por la definición, sabemos que $\sigma[x..] \vDash \varphi$. Además, es trivial notar que $\forall 0 \leq j \leq x - 1 : \sigma[j..] \vDash \psi$, por lo que por definición $\sigma[(x-1)..] \vDash \varphi$. Expandiendo esta idea, llegamos a que $\forall 0 \leq j \leq x : \sigma[j..] \vDash \varphi$.

Con lo anterior, la definición puede pasar a ser:

$$\sigma \vDash \varphi \mathrm{W} \psi \iff \forall i \geq 0 : (\forall 0 \leq j \leq i : \sigma[j..] \vDash \psi) \Rightarrow (\forall 0 \leq j \leq i : \sigma[j..] \vDash \varphi) \tag{6.2}$$

El **primer punto** solicita expresar el operador *while* en términos del *until* y expresiones booleanas. Para ello, podremos pensar que siempre en el prefijo se cumple $\psi \to \varphi$ hasta que deja de cumplirse ψ. Es decir: $\varphi \mathrm{W} \psi \equiv ((\psi \to \varphi)\mathrm{U}\neg\psi) \vee \Box(\psi \wedge \varphi)$. Lo único a modificar es el *globally*, el cual se puede considerar como $\neg(\mathrm{T\ U}\neg(\psi \wedge \varphi))$ donde T es una tautología. Notar que en este caso se busca que no se cumpla que en algún momento suceda $\neg(\psi \wedge \varphi)$. Por ello, decimos que: $\varphi \mathrm{W} \psi \equiv ((\psi \to \varphi)\mathrm{U}\neg\psi) \vee \neg(\mathrm{T\ U}\neg(\psi \wedge \varphi))$.

Finalmente, el **segundo punto** pide expresar el operator *until* en términos del *while* y operaciones booleanas. Aquí, como queremos que el until sea $\varphi \mathrm{U} \psi$ y dada la definición vista anteriormente, tomaremos algo de la forma $\gamma \mathrm{W}\neg\psi \wedge \Diamond\psi$. De este modo, tendremos que $\gamma \mathrm{W}\neg\psi \wedge \Diamond\psi \equiv (((\neg\psi \to \gamma)\mathrm{U}\psi) \vee \Box(\neg\psi \wedge \gamma)) \wedge \Diamond\psi$. Como no puede cumplirse al mismo tiempo $\Box(\neg\psi \wedge \gamma)$ y $\Diamond\psi$, tenemos que $\gamma \mathrm{W}\neg\psi \wedge \Diamond\psi \equiv (\neg\psi \to \gamma)\mathrm{U}\psi \wedge \Diamond\psi$. Ahora, al ser un until, $\Diamond\psi$ resulta redundante dado que el until ya garantiza que en algún momento ψ sucederá. Por ello, podemos considerar $\gamma \mathrm{W}\neg\psi \wedge \Diamond\psi \equiv (\neg\psi \to \gamma)\mathrm{U}\psi$. Finalmente, se busca que $\neg\psi \to \gamma \equiv \varphi$. Si bien esto no es posible, podemos considerar $\gamma = \varphi$ tal que $\varphi \mathrm{W}\neg\psi \wedge \Diamond\psi \equiv (\neg\psi \to \varphi)\mathrm{U}\psi$.

Si bien este último punto es diferente al until que queremos, igualmente es válido dado que cuando valga por primera vez ψ este marcará el final del until. En las posiciones anteriores a este final, se cumplirá $\neg\psi$ por lo que el implica determina que vale φ en cada paso anterior. Por ello, finalmente, consideramos $\varphi \mathrm{U} \psi \equiv \varphi \mathrm{W}\neg\psi \wedge \Diamond\psi$. El único paso que resta realizar es la traducción de $\Diamond\psi$ con el uso del *while*. Para ello, tenemos $\Diamond\psi \equiv \neg(\neg\psi \mathrm{W\ T})$ dado que no se cumple que mientras suceda "cualquier propiedad" no se cumpla ψ. Notar que el while aquí junto a la tautología ayuda a expresar el globally. Finalmente, la equivalencia queda como $\varphi \mathrm{U} \psi \equiv (\varphi \mathrm{W}\neg\psi) \wedge \neg(\neg\psi \mathrm{W\ T})$.

Ejercicio 7

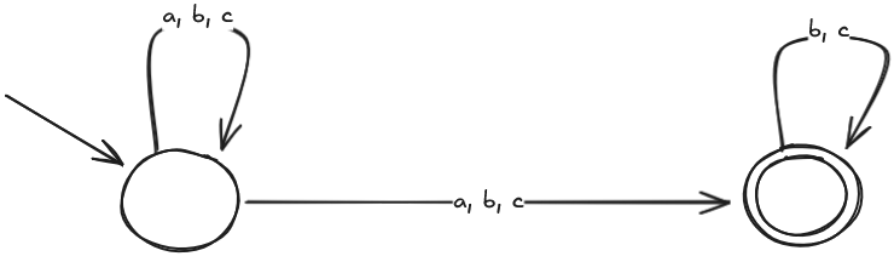
Queremos definir el operador Z tal que $\varphi Z \psi$ represente la propiedad de que siempre entre dos posiciones distintas en las cuales valga φ, existe una en la que vale ψ. Considero que estar entre medio de las otras dos implica que no se consideran los extremos.

Motivo de ello, la definición que doy es la siguiente: $\square(\varphi \rightarrow \neg \bigcirc (\neg\psi \mathrm{U} \varphi))$. Es decir, me concentro en la negación de la propiedad para definirla. Considero que cuando "veo" un φ, entonces a partir de la siguiente posición no puede pasar que veo otro φ antes de algún ψ.

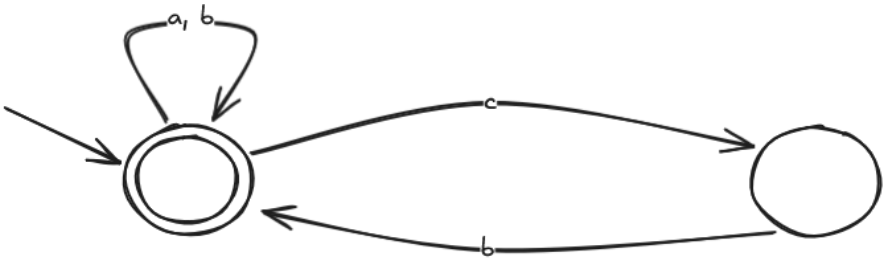
Ejercicio 8

Veamos cada autómata con alfabeto $\{a, b, c\}$ por separado.

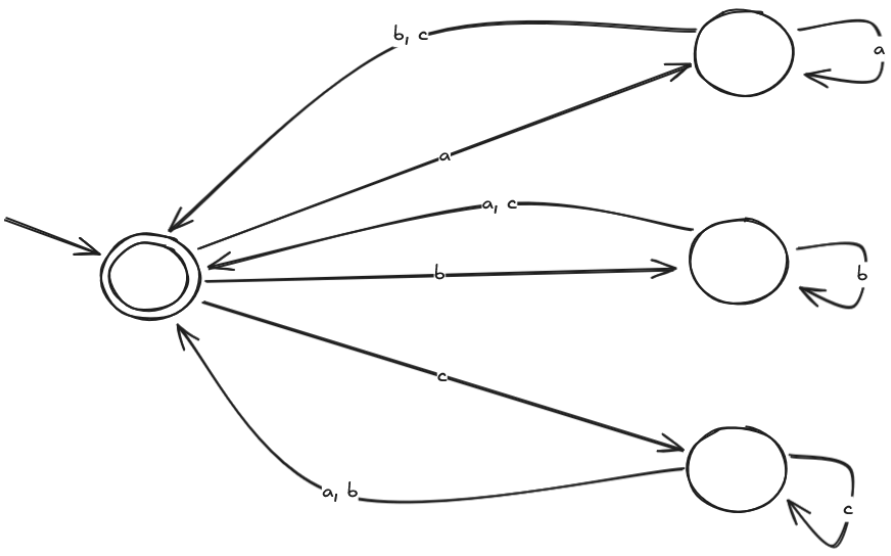
Para el caso de conjunto de cadenas infinitas con una cantidad finita de a tenemos el siguiente diagrama:



Para el conjunto de cadenas infinitas en que cada ocurrencia de c viene inmediatamente seguida de una ocurrencia de b tenemos:



Para el conjunto de cadenas infinitas que no terminen en una secuencia infinita de a ni en una de b ni en una de c tenemos:



Y para el conjunto de cadenas infinitas con cantidades finitas de a, b y c, como el alfabeto es $\{a, b, c\}$, entonces el conjunto es claramente vacío. Por ello, el autómata de Büchi puede ser cualquiera siempre y cuando el conjunto de estados aceptados sea vacío.

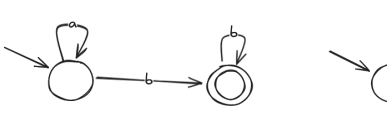
Ejercicio 9

Notemos que el diagrama de la izquierda representa el lenguaje ω-regular $b^*a(a+b)^\omega$ mientras que el de la derecha a $(a+b)^*a(a+b)^\omega$. Por ello, aceptan el mismo lenguaje ω-regular porque el no determinismo de la transición a en el autómata de la derecha se puede resolver tomando siempre la transición que conecta ambos estados. Se puede verificar que esto no modifica el lenguaje del autómata porque en ambos estados puede tomar las mismas acciones, por lo que se sigue cumpliendo que toda traza con al menos una a sea aceptada.

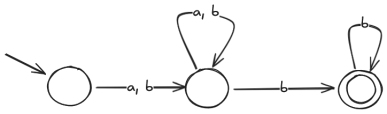
Respecto al lenguaje ω-regular, como bien se dijo antes, es aquél que expresa que toda traza con al menos una a es aceptada. Por motivos de facilidad para notar el prefijo que se quiere, considero la expresión no determinística del autómata de la derecha.

Ejercicio 10

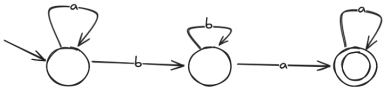
Las expresiones ω-regulares tratadas en los ejercicios 8, 9 y 10 de la anterior guía se presentan a continuación junto con el autómata de Büchi correspondiente:



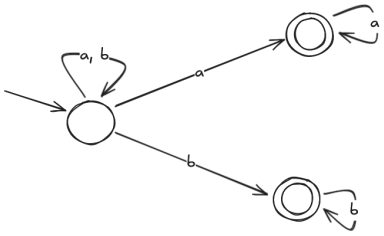
(1) a^*b^ω



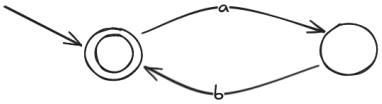
(2) $(b+a)^+b^\omega$



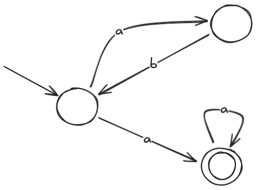
(3) $a^*b^+a^\omega$



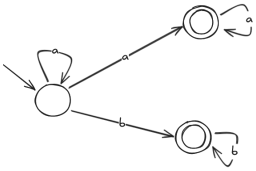
(4) $(a+b)^*(a^\omega+b^\omega)$



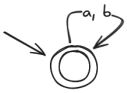
(5) $(ab)^\omega$



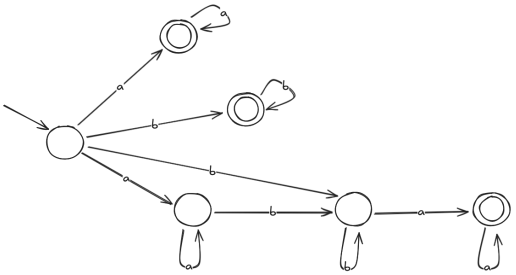
(6) $(ab)^*a^\omega$



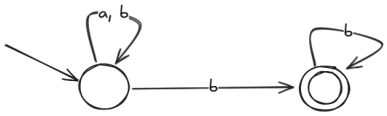
(7) $a^*(a^\omega+b^\omega)$



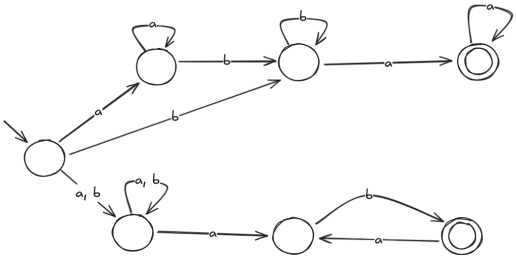
(8) Σ^ω



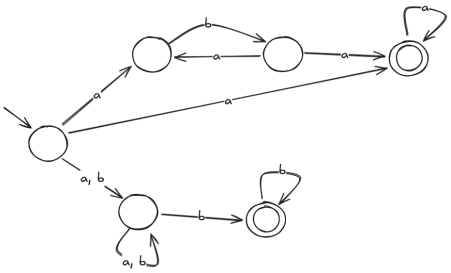
(9) $a^*b^+a^\omega+a^\omega+b^\omega$



(10) $(a^*b^\omega)+(a+b)^*b^\omega=(a+b)^*b^\omega$



(11) $a^*b^+a^\omega+(a+b)^*(ab)^\omega$



(12) $(ab)^*a^\omega+(a+b)^*b^\omega$

Ejercicio 11

Dados $\mathcal{A}_i = (S_i, s_0^i, \delta_i, \Sigma, A_i)$ con $i = 1, 2$ autómatas de Büchi, se define la intersección como:

$$\mathcal{A}_1 \cap \mathcal{A}_2 = (S_1 \times S_2 \times \{0,1,2\}, (s_0^1, s_0^2, 0), \delta, \Sigma, S_1 \times S_2 \times \{2\})$$

donde:

$$(s_1', s_2', i) \in \delta((s_1, s_2, j), a)$$
$$\Updownarrow$$
$$s_1' \in \delta_1(s_1, a) \wedge s_2' \in \delta_2(s_2, a) \wedge \begin{cases} & (j = 0 \wedge s_1 \in A_1 \wedge i = 1) \\ \vee & (j = 0 \wedge s_1 \notin A_1 \wedge i = 0) \\ \vee & (j = 1 \wedge s_2 \in A_2 \wedge i = 2) \\ \vee & (j = 1 \wedge s_2 \notin A_2 \wedge i = 1) \\ \vee & (j = 2 \wedge i = 0) \end{cases}$$

Ejercicio 12

Se pretende demostrar la igualdad $\overline{L(A_\varphi)} = L(A_{\neg\varphi})$. Para ello, sea σ una traza, veamos que:

$$\begin{aligned} \sigma \in \overline{L(A_\varphi)} &\iff \sigma \notin L(A_\varphi) \\ &\iff \sigma \notin L(\varphi) \\ &\iff \sigma \nvDash \varphi \\ &\iff \sigma \vDash \neg\varphi \\ &\iff \sigma \in L(\neg\varphi) \\ &\iff \sigma \in L(A_{\neg\varphi}) \end{aligned}$$

Luego, como $\forall\sigma : \sigma \in \overline{L(A_\varphi)} \iff \sigma \in L(A_{\neg\varphi})$, se demuestra la igualdad de los lenguajes.

Ejercicio 13

Sea φ una fórmula, se pretende dar un algoritmo para corroborar que sea satisfactible, es decir, si es verdadera en algún modelo.

Para ello, notemos que:

$$\begin{aligned} \varphi \text{ es satisfactible} &\iff \exists\sigma : \sigma \vDash \varphi \\ &\iff \exists\sigma : \sigma \in \mathcal{L}(\varphi) \\ &\iff \mathcal{L}(\varphi) \neq \emptyset \\ &\iff \mathcal{L}(\mathcal{A}_\varphi) \neq \emptyset \end{aligned}$$

Luego, entonces, el algoritmo se reduce a:

1. Construir el autómata de Büchi de φ.
2. Calcular el lenguaje de $\mathcal{A}_\varphi$.
3. Corroborar la no vacuidad en el lenguaje.

Ejercicio 14

Se pretende dar un algoritmo que corrobore si un modelo M refina a otro modelo M', es decir, si todas las propiedades que M satisface también las satisface M'.

Para ello, notemos que:

$$\begin{aligned} (\forall\varphi : M \vDash \varphi \Rightarrow M' \vDash \varphi) &\iff \mathcal{L}(M) \subseteq \mathcal{L}(M') \\ &\iff \mathcal{L}(M) \cap \overline{\mathcal{L}(M')} = \emptyset \\ &\iff \mathcal{L}(\mathcal{A}_M) \cap \overline{\mathcal{L}(\mathcal{A}_{M'})} = \emptyset \\ &\iff \mathcal{L}(\mathcal{A}_M) \cap \mathcal{L}(\mathcal{A}_{M'}^c) = \emptyset \end{aligned}$$

Luego, entonces, el algoritmo se reduce a:

1. Construir el autómata de Büchi de $\mathcal{A}_M$.
2. Construir el autómata complemento de Büchi $\mathcal{A}^c_{M'}$.
3. Calcular los lenguajes $\mathcal{L}(\mathcal{A}_M)$ y $\mathcal{L}(\mathcal{A}^c_{M'})$.
4. Corroborar vacuidad en la intersección de los lenguajes.

Ejercicio 15

Item 1

Se busca representar la propiedad de safety del problema de lectores-escritores. Para ello, notar que se puede expresar positivamente como:

$$\Box((\text{reader} > 0 \land \text{writer} = 0) \lor (\text{reader} = 0 \land \text{writer} \leq 1))$$

o, de forma más compacta e intuitiva, en su forma negativa:

$$\neg\Diamond(\text{reader} > 0 \land \text{writer} > 0)$$

Item 2

Se busca representar, ahora, fairness incondicional en el problema de lectores y escritores. Recordemos que fairness incondicional significa que el evento debe ocurrir de manera recurrente. Por ello, tenemos:

$$\Box\Diamond(\text{reader} > 0) \land \Box\Diamond(\text{writer} > 0)$$

Ejercicio 16

Para representar la exclusión mutua (i.e., solo uno puede usar la impresora en un momento dado), consideraremos:

$$\neg\Diamond(\text{Itchy.use} \land \text{Scratchy.use})$$

Para representar que el tiempo de uso es finito (i.e., uno no puede estar habilitado durante un tiempo continuo infinito), tenemos:

$$\neg\Diamond(\text{Itchy.request} \land \bigcirc\neg\Diamond\text{I.release}) \land \neg\Diamond(\text{Scratchy.request} \land \bigcirc\neg\Diamond\text{S.release})$$

Ahora, para representar la ausencia de inanición individual (i.e., si uno pretende imprimir algo, ocasionalmente lo podrá hacer), consideraré que lo importante no es el uso de la impresora sino que el usuario haya sido habilitado para su uso. Es decir, se busca que si se hace una request por parte de un usuario, entonces en algún momento este hará un release (lo que significa que la tuvo habilitada durante un tiempo):

$$\Box((\text{Itchy.request} \to \bigcirc\Diamond\text{Itchy.release}) \land (\text{Scratchy.request} \to \bigcirc\Diamond\text{Scratchy.release}))$$

Y, finalmente, para representar el acceso alternante (i.e., el uso de la impresora se hace de forma alternada estricta), consideraré que lo que se alterna son las habilitaciones. Por ello, la mejor forma de encararlo es que siempre que un usuario haga un release, entonces no puede hacer otro release antes de que el otro usuario libere la impresora. Esto es equivalente a decir que si termino un período de habilitación, entonces no puedo terminar otro a continuación sin que otra persona use la impresora antes porque estaríamos rompiendo el acceso alternante. Con esto, nos queda:

$$\neg\Diamond\left((\text{Itchy.release} \land \bigcirc\left(\neg\text{Scratchy.release U Itchy.release}\right))\right) \land$$
$$\neg\Diamond\left((\text{Scratchy.release} \land \bigcirc\left(\neg\text{Itchy.release U Scratchy.release}\right))\right)$$

Ejercicio 17

Las proposiciones atómicas que se considerarán para modelar el problema son las siguientes:

- request.i
- open.i

- move

con $0 \leq i < N$.

Además, recordemos que el ascensor tiene memoria, por lo que una request está "prendida" desde que se inició este hasta que el ascensor abra sus puertas en el piso correspondiente.

Respecto a las fórmulas LTL, tenemos:

1. La puerta no está abierta si no está en el piso correspondiente:

$$\neg\Diamond(\neg\text{request}.i \wedge \text{open}.i) \qquad \forall 0 \leq i < N$$

2. El requerimiento será atendido en algún momento:

$$\Box(\text{request}.i \rightarrow \Diamond\text{open}.i) \qquad \forall 0 \leq i < N$$

3. Siempre retorna a planta baja (lo interpreto como que frecuentemente va a abrir sus puertas ahí):

$$\Box\Diamond\text{open}.0$$

4. Cuando hay un requerimiento para $N-1$, se atiende inmediatamente:

$$\Box(\text{request}.(N-1) \rightarrow (\neg\text{open}.0 \wedge \cdots \wedge \neg\text{open}.(N-2)) \text{ U } \text{open}.(N-1))$$

5. El ascensor no se mueve a menos que se requiera:

$$\Box((\text{request}.0 \vee \cdots \vee \text{request}.(N-1)) \leftrightarrow \text{move})$$