Lógica Computacional
Nota 14. Verificación de modelos*

Noé Salomón Hernández S.

1. Lógica temporal de tiempo lineal

La *lógica temporal de tiempo lineal* (*linear-time temporal logic*), o LTL, es una lógica temporal con conectivos que nos permiten hacer referencia al futuro. Modela el tiempo como una secuencia de estados, extendiéndose infinitamente al futuro. Esta secuencia de estados es llamada trayectoria de computación. Como el futuro no está determinado, consideramos muchas trayectorias representando diferentes futuros posibles, uno de los cuales es el que en verdad se cumple.

Trabajamos con un conjunto fijo `Atoms` de fórmulas atómicas, que representan hechos atómicos que se satisfacen en el sistema.

1.1. Sintaxis de LTL

Definición 1.1. LTL tiene la siguiente sintaxis dada en forma de Backus Naur:

$$\begin{aligned}\varphi ::= \top \mid \bot \mid p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi)\\ (\mathrm{X}\varphi) \mid (\mathrm{F}\varphi) \mid (\mathrm{G}\varphi) \mid (\varphi \mathrm{U} \varphi) \mid (\varphi \mathrm{W} \varphi) \mid (\varphi \mathrm{R} \varphi)\end{aligned}$$

donde p es cualquier átomo proposicional tomado del conjunto `Atoms`.

Los conectivos X, F, G, U, R y W son llamados *conectivos temporales*. X significa 'próximo estado', F significa 'algún estado futuro', y G significa 'todos los estados futuros (globalmente)'. Los restantes tres, U, R y W son llamados 'hasta', 'libera', y 'hasta-débil' respectivamente.

Convención. Los conectivos unarios ($\neg$ y los conectivos temporales X, F y G) tienen mayor precedencia. Enseguida vienen en orden de precedencia U, R y W; luego vienen $\wedge$ y $\vee$; y después viene $\to$. Esta convención nos permite dejar a un lado ciertos paréntesis. Por ejemplo,

- $(((\mathrm{F}\,p) \wedge (\mathrm{G}\,q)) \to (p \mathrm{W}\, r))$ se reescribe como $\mathrm{F}\,p \wedge \mathrm{G}\,q \to p \mathrm{W}\, r$,
- $(\mathrm{F}\,(p \to \mathrm{G}\,r) \vee ((\neg q)\,\mathrm{U}\,p))$ se reescribe como $\mathrm{F}\,(p \to \mathrm{G}\,r) \vee \neg q\,\mathrm{U}\,p$,
- $(p \mathrm{W}\,(q \mathrm{W}\, r))$ se reescribe como $p \mathrm{W}\,(q \mathrm{W}\, r)$,
- $((\mathrm{G}\,(\mathrm{F}\,p)) \to (\mathrm{F}\,(q \vee s)))$ se reescribe como $\mathrm{G}\,\mathrm{F}\,p \to \mathrm{F}\,(q \vee s)$.

*Esta nota se base en el libro de Huth y Ryan, *Logic in Computer Science*.

1.2. Semántica de LTL

La clase de sistemas que estamos interesados en verificar usando LTL pueden ser modelados como sistemas de transición. Un sistema de transición modela un sistema por medio de *estados* (estructura estática) y *transiciones* (estructura dinámica).

Definición 1.2. Un sistema de transición (o estructura de Kripke) $\mathcal{M} = (S, \to, L)$ es un conjunto de estados dotado de una relación de transición $\to$ (una relación binaria sobre S), tal que cada estado $s \in S$ tiene algún $s' \in S$ con $s \to s'$, y una función de etiquetamiento $L : S \to \mathcal{P}(\texttt{Atoms})$.

Una buena forma de pensar acerca de L es simplemente como una asignación de valores de verdad a todas las variables proposicionales, como se hacía en lógica proposicional (a esto lo llamamos una *interpretación*). La diferencia es que tenemos *más de un estado*, así que esta asignación depende del estado s en el que se encuentra el sistema: $L(s)$ contiene a todos los átomos que son verdaderos en el estado s.

Una manera conveniente de expresar toda la información de un sistema de transición $\mathcal{M}$ es usar gráficas dirigidas cuyos nodos (estados) contienen todos los átomos proposicionales que son verdaderos en ese estado. Un ejemplo de tales gráficas dirigidas se aprecia en la Figura 1.

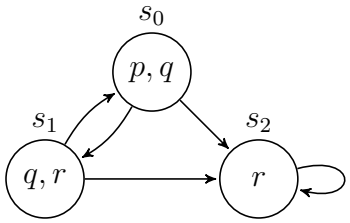


Figura 1: Una representación de $\mathcal{M} = (S, \to, L)$ como una gráfica dirigida. Etiquetamos al estado s con l syss $l \in L(s)$.

El requisito en la Definición 1.2 que para todo $s \in S$ hay al menos un $s' \in S$ con $s \to s'$ significa que no hay estado en el sistema que esté 'bloqueado'. Es un detalle técnico que no representa ninguna restricción real sobre los sistemas que se pueden modelar. Si un sistema se llega a bloquear, podemos siempre agregar un estado extra s_d representando el bloqueo, junto con nuevas transiciones $s \to s_d$ para cada s que se bloquee en el modelo original, y también $s_d \to s_d$.

Definición 1.3. Una trayectoria en un modelo $\mathcal{M} = (S, \to, L)$ es una secuencia infinita de estados $s_1, s_2, \ldots$ en S tal que, para cada $i \geq 1$, $s_i \to s_{i+1}$. Escribimos la trayectoria como $s_1 \to s_2 \to \ldots$

Considere la trayectoria $\pi = s_1 \to s_2 \to \ldots$ que representa un posible futuro en nuestro sistema: primero se encuentra en el estado s_1, luego en el estado s_2, y así sucesivamente. Escribimos π^i para el sufijo comenzando en s_i, es decir, $s_i \to s_{i+1} \to \ldots$

Es útil visualizar todas las posibles trayectorias de computación originadas en un estado s al desenrollar el sistema de transición obteniendo un árbol de cómputo infinito. Las trayectorias de ejecución de un modelo $\mathcal{M}$ son representadas explícitamente en el árbol obtenido al desenrollar el modelo. Si desenrollamos el modelo en la Figura 1 para el estado inicial s_0, obtenemos el árbol infinito en la Figura 2.

Definición 1.4. Sea $\mathcal{M} = (S, \to, L)$ un modelo y $\pi = s_1 \to s_2 \to \ldots$ una trayectoria en $\mathcal{M}$. Determinar si π satisface a una fórmula en LTL se lleva a cabo mediante la relación de satisfacción $\models$ como sigue:

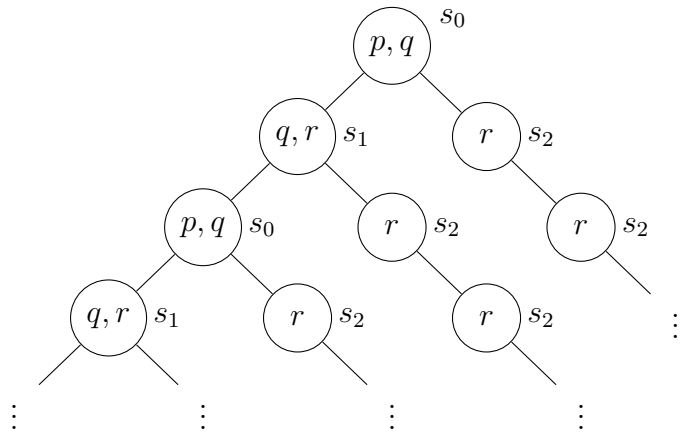


Figura 2: El sistema en la Figura 1 desenrollado como un árbol de todas las trayectorias de ejecución comenzando en el estado s_0.

1. $\pi \models \top$
2. $\pi \not\models \bot$
3. $\pi \models p$ syss $p \in L(s_1)$
4. $\pi \models \neg\varphi$ syss $\pi \not\models \varphi$
5. $\pi \models \varphi_1 \wedge \varphi_2$ syss $\pi \models \varphi_1$ y $\pi \models \varphi_2$
6. $\pi \models \varphi_1 \vee \varphi_2$ syss $\pi \models \varphi_1$ o $\pi \models \varphi_2$
7. $\pi \models \varphi_1 \rightarrow \varphi_2$ syss $\pi \models \varphi_2$ siempre que $\pi \models \varphi_1$
8. $\pi \models \mathrm{X}\,\varphi$ syss $\pi^2 \models \varphi$
9. $\pi \models \mathrm{G}\,\varphi$ syss para toda $i \geq 1$, $\pi^i \models \varphi$
10. $\pi \models \mathrm{F}\,\varphi$ syss existe alguna $i \geq 1$ tal que $\pi^i \models \varphi$
11. $\pi \models \varphi\,\mathrm{U}\,\psi$ syss existe alguna $i \geq 1$ tal que $\pi^i \models \psi$ y para toda $j = 1, \ldots, i-1$ tenemos $\pi^j \models \varphi$
12. $\pi \models \varphi\,\mathrm{W}\,\psi$ syss, o bien, existe alguna $i \geq 1$ tal que $\pi^i \models \psi$ y para toda $j = 1, \ldots, i-1$ tenemos $\pi^j \models \varphi$; o, para toda $k \geq 1$ tenemos $\pi^k \models \varphi$
13. $\pi \models \varphi\,\mathrm{R}\,\psi$ syss, o bien, existe alguna $i \geq 1$ tal que $\pi^i \models \varphi$ y para toda $j = 1, \ldots, i$ tenemos $\pi^j \models \psi$; o, para toda $k \geq 1$ tenemos $\pi^k \models \psi$

La cláusula 8 elimina el primer estado de la trayectoria, para crear una trayectoria comenzando en el 'siguiente' (segundo) estado.

Note que la cláusula 3 indica que los átomos son evaluados en el primer estado a lo largo de la trayectoria en consideración. Esto no significa que todos los átomos presentes en una fórmula LTL se refieren al primer estado de la trayectoria; si están dentro del alcance de un conectivo temporal,

por ejemplo $\mathrm{G}\,(p \rightarrow \mathrm{X}\,q)$, entonces determinar si se satisface o no involucra tomar sufijos de la trayectoria en consideración, y los átomos se refieren al primer estado de dichos sufijos.

Las cláusulas 11-13 lidian con los conectivos temporales binarios. U, que representa 'hasta', es el más común. La fórmula $\varphi\,\mathrm{U}\,\psi$ se satisface sobre una trayectoria si φ se satisface continuamente *hasta* que ψ se satisface. $\varphi\,\mathrm{U}\,\psi$ requiere que ψ *se satisfaga* en un estado futuro.

Los otros conectivos binarios son W, que representa 'hasta-débil', y R, que representa 'libera'. El hasta-débil es como U, excepto que $\varphi\,\mathrm{W}\,\psi$ no requiere que ψ se satisfaga en algún momento a lo largo de la trayectoria en cuestión, lo cual es requerido por $\varphi\,\mathrm{U}\,\psi$. R es el dual de U; es decir, $\varphi\,\mathrm{R}\,\psi$ es equivalente a $\neg(\neg\varphi\,\mathrm{U}\,\neg\psi)$. Se llama 'libera' (o *release* en inglés) porque la cláusula 13 indica que ψ debe permanecer verdadera hasta el momento (incluyendo tal momento) en que φ se vuelva verdadera, si es que se alcanza ese instante; decimos así que φ 'libera' a ψ. R y W son en realidad muy similares; la diferencia es que intercambian los papeles de φ y ψ, y la cláusula para W llega a $i-1$ mientras que R llega a i.

Hemos definido una relación de satisfacción entre trayectorias y fórmulas de LTL. Sin embargo, para verificar sistemas, nos gustaría decir que un modelo como un todo satisface una fórmula LTL. Esto se cumple siempre que ***cada*** posible trayectoria de ejecución del modelo satisface a la fórmula.

Definición 1.5. Suponga que $\mathcal{M} = (S, \rightarrow, L)$ es un modelo, $s \in S$, y φ es una fórmula LTL. Escribimos $\mathcal{M}, s \models \varphi$ si, ***para toda*** trayectoria de ejecución π de $\mathcal{M}$ comenzando en s, tenemos $\pi \models \varphi$.

Si $\mathcal{M}$ se puede inferir del contexto, podemos abreviar $\mathcal{M}, s \models \varphi$ como $s \models \varphi$. Debe ser claro que hemos descrito las formalidades de los fundamentos de un procedimiento que, dado φ, $\mathcal{M}$ y s, puede verificar si $\mathcal{M}, s \models \varphi$ se satisface.

En español, una frase como *nadé hasta los 18 años* puede decir dos cosas, (i) desde una edad temprana nadé y lo suspendí a los 18 años, o (ii) a partir de los 18 años comencé a nadar. La definición que le hemos dado en LTL corresponde al significado (i). Notemos también que ni la versión fuerte (U) ni la versión débil (W) del hasta dice algo acerca de lo que pasa después de que el hasta se ha cumplido. Esto otra vez contrasta con el significado de (i) que indica que una persona nadó desde pequeño y continúo haciéndolo hasta los 18 años, además de tal oración se entiende que la persona ya no nada desde los 18 años en adelante. Esta es una diferencia con la semántica del hasta en la lógica temporal. Podemos expresar la oración (i) al combinar U con otros conectivos; por ejemplo, con la fórmula $n\,\mathrm{U}\,(d \wedge \mathrm{G}\,\neg n)$, donde n representa 'nado' y d representa 'cumplo 18 años'.

Ejercicios 1.6. Sea $\mathcal{M}$ el modelo de la Figura 1. Determine si se satisfacen las siguientes relaciones:

1. $\mathcal{M}, s_0 \models p \vee q$
2. $\mathcal{M}, s_0 \models r \rightarrow q$
3. $\mathcal{M}, s_0 \models r \wedge \mathrm{X}\,r$
4. $\mathcal{M}, s_0 \models \mathrm{G}\,r$
5. $\mathcal{M}, s_0 \models \mathrm{G}\,(p \vee r)$
6. $\mathcal{M}, s_0 \models \mathrm{F}\,\mathrm{G}\,r$
7. $\mathcal{M}, s_0 \models \mathrm{G}\,\mathrm{F}\,r$
8. $\mathcal{M}, s_0 \models \neg(q\,\mathrm{U}\,r)$
9. $\mathcal{M}, s_2 \models (p \wedge r)\,\mathrm{W}\,r$
10. $\mathcal{M}, s_1 \models p\mathrm{R}\,r$

1.3. Patrones prácticos de especificaciones

¿Qué clase de propiedades importantes en la práctica pueden ser verificadas con fórmulas de LTL? Listamos unas cuantas que exhiben patrones comunes de encontrar. Supongamos que las fórmulas atómicas incluyen a admitida e iniciado. Quizá necesitemos de algunas de las siguientes propiedades de los sistemas en el mundo real.

- Para cualquier estado, si una solicitud para algún recurso se presenta, entonces será eventualmente admitida:
$$\mathrm{G}\,(\mathsf{solicitud} \rightarrow \mathrm{F}\,\mathsf{admitida})$$
- Un cierto proceso se habilita de manera infinitamente frecuente sobre toda trayectoria de ejecución:
$$\mathrm{G}\,\mathrm{F}\,\mathsf{habilita}$$
- Sin importar lo que ocurra, cierto proceso eventualmente estará permanentemente bloqueado:
$$\mathrm{F}\,\mathrm{G}\,\mathsf{bloqueado}$$
- Si el proceso se habilita de manera infinitamente frecuente, entonces se ejecutara de manera infinitamente frecuente:
$$\mathrm{G}\,\mathrm{F}\,\mathsf{habilita} \rightarrow \mathrm{G}\,\mathrm{F}\,\mathsf{ejecutara}$$
- Es imposible llegar a un estado iniciado sin que esté listo:
$$\mathrm{G}\,\neg(\mathsf{iniciado} \wedge \neg\mathsf{listo})$$
Bajo una interpretación sobre estados ($s \models \varphi$), no podemos en LTL afirmar que *es posible* llegar a un estado iniciado sin que esté listo debido a que no podemos expresar la existencia de trayectorias. La negación de la fórmula anterior, i.e., $\neg\,\mathrm{G}\,\neg(\mathsf{iniciado} \wedge \neg\mathsf{listo})$ indica que *todas* las trayectorias eventualmente llegarán a un estado iniciado sin que esté listo.

Una clase de afirmaciones que no se pueden decir en LTL son enunciados que declaran la existencia de una trayectoria, tales como:

- Desde cualquier estado *es posible llegar a un estado de* reinicio, i.e., hay una trayectoria desde cualquier estado a un estado satisfaciendo reinicio.
- El elevador *puede* permanecer inactivo en el tercer piso con sus puertas cerradas, i.e., desde el estado en el cual se está en el tercer piso, hay una trayectoria a lo largo de la cual permanece allí.

1.4. Equivalencias importantes entre fórmulas LTL

Definición 1.7. Decimos que dos fórmulas LTL φ y ψ son equivalentes semánticamente, o simplemente equivalentes, lo que denotamos como $\varphi \equiv \psi$, si para todo modelo $\mathcal{M}$ y toda trayectoria π: $\pi \models \varphi$ syss $\pi \models \psi$.

F y G son duales uno del otro, y X es dual consigo mismo:

$$\neg \mathrm{G}\,\varphi \equiv \mathrm{F}\,\neg\varphi \qquad \neg \mathrm{F}\,\varphi \equiv \mathrm{G}\,\neg\varphi \qquad \neg \mathrm{X}\,\varphi \equiv \mathrm{X}\,\neg\varphi$$

además U y R son duales uno del otro

$$\neg(\varphi\,\mathrm{U}\,\psi) \equiv \neg\varphi\,\mathrm{R}\,\neg\psi \qquad \neg(\varphi\,\mathrm{R}\,\psi) \equiv \neg\varphi\,\mathrm{U}\,\neg\psi$$

También es el caso que F se distribuye sobre $\vee$ y G sobre $\wedge$, i.e.,

$$\begin{aligned} \mathrm{F}\,(\varphi \vee \psi) &\equiv \mathrm{F}\,\varphi \vee \mathrm{F}\,\psi \\ \mathrm{G}\,(\varphi \wedge \psi) &\equiv \mathrm{G}\,\varphi \wedge \mathrm{G}\,\psi \end{aligned}$$

Aquí hay dos equivalencias más en LTL:

$$\mathrm{F}\,\varphi \equiv \top\,\mathrm{U}\,\varphi \qquad \mathrm{G}\,\varphi \equiv \bot\,\mathrm{R}\,\varphi$$

Mostramos una equivalencia más que relaciona la versión fuerte y débil del 'hasta', U y W. El hasta fuerte puede ser visto como un hasta débil junto con la condición de que la eventualidad debe ocurrir:

$$\varphi\,\mathrm{U}\,\psi \equiv \varphi\,\mathrm{W}\,\psi \wedge \mathrm{F}\,\psi$$

Esta equivalencia se demuestra suponiendo primero que una trayectoria satisface $\varphi\,\mathrm{U}\,\psi$. Entonces, por la cláusula 11, tenemos $i \geq 1$ tal que $\pi^i \models \psi$ y para toda $j = 1, \ldots, i-1$ tenemos $\pi^j \models \varphi$. Considerando la cláusula 12, lo anterior implica $\varphi\,\mathrm{W}\,\psi$, y considerando la cláusula 10, esto también implica $\mathrm{F}\,\psi$. Así, para todas las trayectorias π, si $\pi \models \varphi\,\mathrm{U}\,\psi$ entonces $\pi \models \varphi\,\mathrm{W}\,\psi \wedge \mathrm{F}\,\psi$. Segundo, supongamos π tal que $\pi \models \varphi\,\mathrm{W}\,\psi \wedge \mathrm{F}\,\psi$. Por la cláusula 5 tenemos $\pi \models \varphi\,\mathrm{W}\,\psi$ y $\pi \models \mathrm{F}\,\psi$. Que $\pi \models \varphi\,\mathrm{W}\,\psi$ implica que:

1. o bien, existe alguna $i \geq 1$ tal que $\pi^i \models \psi$ y para toda $j = 1, \ldots, i-1$ tenemos $\pi^j \models \varphi$;
2. o, para toda $k \geq 1$ tenemos $\pi^k \models \varphi$

Si se cumple 1, también se cumple $\pi \models \varphi\,\mathrm{U}\,\psi$. Si se cumple 2, recordamos que se cumple $\pi \models \mathrm{F}\,\psi$, y en conjunto tenemos que existe alguna $i \geq 1$ tal que $\pi^i \models \psi$ y para toda $k \geq 1$ tenemos $\pi^k \models \varphi$, esto hace que se cumpla $\pi \models \varphi\,\mathrm{U}\,\psi$. Así, para todas las trayectorias π, si $\pi \models \varphi\,\mathrm{W}\,\psi \wedge \mathrm{F}\,\psi$ entonces $\pi \models \varphi\,\mathrm{U}\,\psi$.

Escribir W en términos de U es posible: W es como U pero permitimos la posibilidad que ψ nunca ocurra:

$$\varphi\,\mathrm{W}\,\psi \equiv \varphi\,\mathrm{U}\,\psi \vee \mathrm{G}\,\varphi.$$

También nos percatamos que los comportamientos de R y W son similares; son diferentes en que intercambian los argumentos φ y ψ, además de que W llega hasta $i-1$ mientras que R hasta i. Por lo tanto, R y W pueden ser definidas en términos uno del otro, como sigue

$$\varphi\,\mathrm{W}\,\psi \equiv \psi\,\mathrm{R}\,(\varphi \vee \psi) \tag{1.4.1}$$

$$\varphi\,\mathrm{R}\,\psi \equiv \psi\,\mathrm{W}\,(\varphi \wedge \psi). \tag{1.4.2}$$

1.5. Conjunto adecuado de conectivos para LTL

Como en lógica proposicional, en LTL hay cierta redundancia entre los conectivos temporales. Sabemos que el conjunto $\{\bot, \wedge, \neg\}$ forma un conjunto adecuado de conectivos, ya que $\rightarrow, \vee, \top$, etc., pueden ser escritos en términos de los tres elementos del conjunto.

Conjuntos adecuados de conectivos también existen para LTL. Tomando en cuenta que X es independiente de los otros conectivos, ya que no ayuda a definir ninguno de los otros conectivos ni puede ser obtenido por una combinación de ellos, encontramos a $\{\mathrm{U}, \mathrm{X}\}$, $\{\mathrm{R}, \mathrm{X}\}$ y $\{\mathrm{W}, \mathrm{X}\}$ como los conjuntos adecuados para LTL porque:

- R y W pueden ser definidos a partir de U, por la dualidad $\varphi \,\mathrm{R}\, \psi \equiv \neg(\neg\varphi \,\mathrm{U}\, \neg\psi)$ y la equivalencia 1.4.1, respectivamente.

- U y W pueden ser definidos a partir de R, por la dualidad $\varphi \,\mathrm{U}\, \psi \equiv \neg(\neg\varphi \,\mathrm{R}\, \neg\psi)$ y la equivalencia 1.4.1, respectivamente.

- R y U pueden ser definidos a partir de W, por la equivalencia 1.4.2 y la dualidad $\varphi \,\mathrm{U}\, \psi \equiv \neg(\neg\varphi \,\mathrm{R}\, \neg\psi)$ mediante 1.4.2, respectivamente.

1.6. Verificación de modelos: exclusión mutua

Cuando procesos concurrentes comparten algún recurso, puede ser necesario asegurarse que no tienen acceso a dicho recurso al mismo tiempo. Tampoco sería deseable que varios procesos editen simultáneamente el mismo archivo. Por lo tanto identificamos ciertas *secciones críticas* del código de cada proceso y acordamos que solo un proceso puede estar en su sección crítica a la vez. La sección crítica debe incluir todos los accesos a los recursos *compartidos*. El problema es encontrar un *protocolo* para determinar a qué proceso se le permite entrar a su sección crítica a la vez. Ya que encontramos uno que creemos que sí funciona, corroboramos nuestra solución verificando que cumplen algunas propiedades esperadas, como las siguientes:

Seguridad: únicamente un proceso está en su sección crítica a la vez. Esta propiedad de seguridad no es suficiente, ya que un protocolo que excluye permanentemente a cada proceso de su sección crítica sería seguro, pero no sería útil. Requerimos lo siguiente:

Vitalidad: Una vez que cualquier proceso solicite entrar a su sección crítica, eventualmente se le permitirá ejecutar tal sección.

No bloqueo: Un proceso siempre puede solicitar entrar a su sección crítica.

Modelaremos dos procesos, cada uno de los cuales está en su sección no crítica (n), o intenta entrar a su sección crítica (t), o se encuentra en su sección crítica (c). Cada proceso individualmente lleva a cabo transiciones en el ciclo $n \rightarrow t \rightarrow c \rightarrow n \rightarrow t \rightarrow c \rightarrow \ldots$ pero la ejecución de cada proceso se entrelaza con la del otro. Considere el protocolo que se describe en la estructura de Kripke $\mathcal{M}$ dada en la Figura 3. Los dos procesos inician en sus secciones no críticas, en el estado s_0. El estado s_0 es el único estado inicial, indicado por la arista entrante sin origen. Cualquiera de los dos procesos podría ahora intentar ingresar a su sección crítica, pero solo uno de ellos puede realizar una transición a la vez (tenemos un *entrelazado asíncrono*). A cada paso, un calendarizador (no especificado) determina qué proceso se ejecuta. Así, hay una arista de transición del estado s_0 al estado s_1 y al estado s_5. Desde s_1 (i.e. proceso 1 intentado entrar a su sección crítica, proceso 2 no en su sección crítica) dos cosas pueden suceder: o bien el proceso 1 lleva a cabo un paso de la

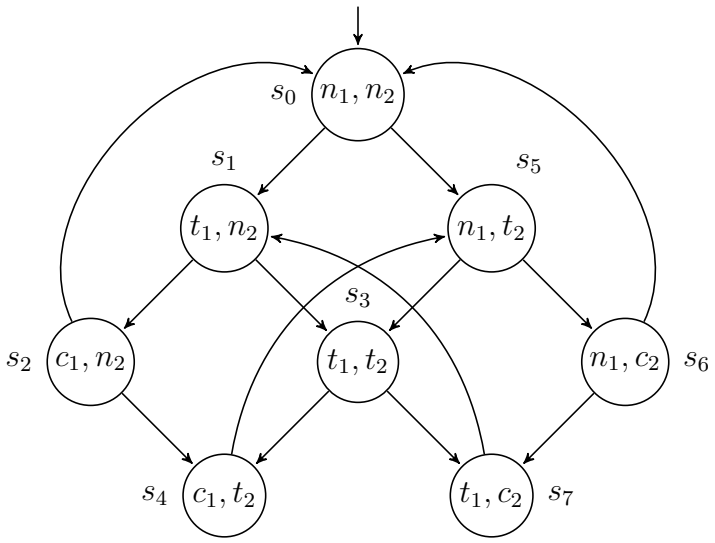


Figura 3: Modelo para la exclusión mutua.

ejecución (nos movemos ahora al estado s_2), o es turno del proceso 2 (así que vamos a s_3). Observe que no todo proceso puede ejecutarse en cada estado. Por ejemplo, el proceso 1 no puede moverse en el estado s_7, ya que no puede entrar a su sección crítica hasta que el proceso 2 salga de su sección crítica.

Verificaremos las propiedades que mencionamos arriba al describirlas como fórmulas de la lógica temporal. No todas ellas son expresables como fórmulas LTL.

Seguridad: Se expresa en LTL como $\mathrm{G}\,\neg(c_1 \wedge c_2)$. Claramente esta fórmula se satisface en todo estado del modelo.

Vitalidad: Se expresa en LTL como $\mathrm{G}\,(t_1 \rightarrow \mathrm{F}\,c_1)$. No se satisface en el estado inicial, porque podemos encontrar una trayectoria comenzando en el estado inicial a lo largo de la cual hay un estado, a saber s_1, en el que t_1 es verdadero pero de allí c_1 es falso en lo que resta de la trayectoria. Tal ejecución es $s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_7 \rightarrow s_1 \rightarrow s_3 \rightarrow s_7 \rightarrow \ldots$ en esta trayectoria c_1 siempre es falso.

No bloqueo: Consideremos sólo al proceso 1. Nos gustaría expresar esta propiedad como: *para cada estado que satisface n_1, existe un sucesor que satisface t_1*. Desafortunadamente, este cuantificador existencial sobre trayectorias no puede ser expresado en LTL. Puede ser expresado en CTL (lógica de árbol de computación), que estudiaremos enseguida.

2. Lógica de árbol de computación

En nuestro análisis previo de LTL, observamos que las fórmulas de LTL son evaluadas sobre *trayectorias*. Definimos que un *estado* de un sistema satisface una fórmula LTL si *todas las trayectorias* desde el estado dado la satisfacen. LTL implícitamente cuantifica universalmente sobre trayectorias. Por lo tanto, las propiedades que afirman la existencia de un trayectoria no pueden ser expresadas en LTL. Este problema puede ser parcialmente aliviado al considerar la negación de la propiedad en cuestión, e interpretar el resultado como corresponde. Para verificar que hay una trayectoria desde el estado s satisfaciendo la fórmula LTL φ, verificamos si todas las trayec-

torias satisfacen $\neg\varphi$; una respuesta positiva a este planteamiento indica que s no satisface a φ, y viceversa. Sin embargo, las propiedades que *combinan* cuantificadores de trayectorias universales y existenciales no pueden en general ser verificadas usando este planteamiento, porque el complemento de la fórmula tiene aún una *combinación* de cuantificadores. Una propiedad que combina cuantificadores de trayectorias es: 'para todas las trayectorias, si se cumple p, entonces existe una trayectoria sobre la cual q siempre es cierta', dicha propiedad no pude ser expresada en LTL pero en CTL sí, su especificación es $\mathrm{A}\,\mathrm{G}\,(p \to \mathrm{E}\,\mathrm{G}\,q)$.

Las lógicas de tiempo ramificado resuelven este problema al permitirnos cuantificar explícitamente sobre trayectorias.

2.1. Sintaxis de CTL

La *lógica de árbol de computación*, o CTL (*Computation Tree Logic*), es una lógica *de tiempo ramificado*, lo que significa que su modelo de tiempo es una estructura de árbol en la que el futuro no está determinado; hay diferentes trayectorias en el futuro, cualquiera de ellas podría ser la trayectoria 'real' que se lleva a cabo.

Definición 2.1. Definimos las fórmulas CTL inductivamente mediante la gramática en forma de Backus Naur:

$$\begin{aligned}\varphi ::= {} & \top \mid \bot \mid p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \\ & \mathrm{A}\,\mathrm{X}\varphi \mid \mathrm{E}\,\mathrm{X}\varphi \mid \mathrm{A}\,\mathrm{F}\varphi \mid \mathrm{E}\,\mathrm{F}\varphi \mid \mathrm{A}\,\mathrm{G}\varphi \mid \mathrm{E}\,\mathrm{G}\varphi \mid \mathrm{A}[\varphi\,\mathrm{U}\,\varphi] \mid \mathrm{E}[\varphi\,\mathrm{U}\,\varphi]\end{aligned}$$

donde p toma valores de un conjunto de fórmulas atómicas.

Notemos que cada uno de los conectivos temporales de CTL consiste de un par de símbolos. El primero en el par es A o E. El símbolo A significa 'a lo largo de todas las trayectorias' (*inevitablemente*) y E significa 'a lo largo de al menos una trayectoria' (*posiblemente*). El segundo símbolo en el par es X, F, G o U, que se identifican como 'siguiente estado', 'algún estado futuro', 'todos los estados futuros' y 'hasta', respectivamente. El par de símbolos en $\mathrm{E}[\varphi_1\,\mathrm{U}\,\varphi_2]$ es $\mathrm{E}\,\mathrm{U}$. En CTL, los pares de símbolos son indivisibles.

Típicamente el símbolo W (la versión débil del hasta) y R (libera o *release*) no son incluidos en CTL ya que se pueden derivar de los otros conectivos.

Convención. Los conectivos unarios ($\neg$ y los conectivos temporales $\mathrm{A}\,\mathrm{X}$, $\mathrm{E}\,\mathrm{X}$, $\mathrm{A}\,\mathrm{F}$, $\mathrm{E}\,\mathrm{F}$, $\mathrm{A}\,\mathrm{G}$ y $\mathrm{E}\,\mathrm{G}$) tienen mayor precedencia. Enseguida vienen en orden de precedencia $\wedge$ y $\vee$; y después vienen $\to$, $\mathrm{A}\,\mathrm{U}$ y $\mathrm{E}\,\mathrm{U}$. La precedencia del 'hasta' cambia respecto a LTL ya que en CTL nos servimos de los corchetes de $\mathrm{A}[\varphi\,\mathrm{U}\,\varphi]$ y de $\mathrm{E}[\varphi\,\mathrm{U}\,\varphi]$ para delimitar a sus argumentos.

Notemos que las presencias de A y E deben estar seguidas por uno de los siguientes símbolos: G, F, X y U; cuando les sigue U, debe ser de la forma $\mathrm{A}[\varphi\,\mathrm{U}\,\psi]$. También tengamos presente que $\mathrm{A}\,\mathrm{U}$ y $\mathrm{E}\,\mathrm{U}$ son conectivos binarios que combinan notación infija y prefija. En notación infija pura escribiríamos $\varphi_1\,\mathrm{AU}\,\varphi_2$, mientras que en notación prefija pura tendríamos $\mathrm{A}\,\mathrm{U}\,(\varphi_1, \varphi_2)$.

2.2. Semántica de CTL

Las fórmulas en CTL son interpretadas sobre sistemas de transiciones, ver Definición 1.2. Sea $\mathcal{M} = (S, \to, L)$ uno de tales modelos, $s \in S$ y φ una fórmula de CTL. La definición de $\mathcal{M}, s \models \varphi$ se basa en la estructura recursiva de φ, y puede ser entendida en palabras como sigue:

- Si φ es atómica, la satisfacción está determinada por L.
- Si el conectivo principal de φ es un conectivo booleano ($\neg$, $\top$, $\wedge$, $\vee$, etc.), entonces la satisfacción de φ corresponde a la definición usual de tabla de verdad una vez aplicada recursión exhaustiva a φ.
- Si el conectivo principal es un operador que inicia con A, se satisface φ si todas las trayectorias comenzando en s satisfacen la 'fórmula LTL' que resulta de quitar el símbolo A.
- Similarmente, si el conectivo principal es un operador que inicia con E, se satisface φ si alguna trayectoria comenzando en s satisface la 'fórmula LTL' que resulta de quitar el símbolo E.

En los últimos dos casos, el resultado de quitar A o E no es estrictamente una fórmula LTL, porque se pueden tener As o Es adicionales. Sin embargo, nos ocuparemos de estos símbolos adicionales a su tiempo gracias a la recursión.

La definición formal de $\mathcal{M}, s \models \varphi$ es más detallada:

Definición 2.2. Sea $\mathcal{M} = (S, \to, L)$ un modelo para CTL, $s \in S$, y φ una fórmula de CTL. La relación $\mathcal{M}, s \models \varphi$ se define por inducción estructural de φ:

1. $\mathcal{M}, s \models \top$ y $\mathcal{M}, s \not\models \bot$.
2. $\mathcal{M}, s \models p$ syss $p \in L(s)$.
3. $\mathcal{M}, s \models \neg\varphi$ syss $\mathcal{M}, s \not\models \varphi$.
4. $\mathcal{M}, s \models \varphi_1 \vee \varphi_2$ syss $\mathcal{M}, s \models \varphi_1$ o $\mathcal{M}, s \models \varphi_2$.
5. $\mathcal{M}, s \models \varphi_1 \wedge \varphi_2$ syss $\mathcal{M}, s \models \varphi_1$ y $\mathcal{M}, s \models \varphi_2$.
6. $\mathcal{M}, s \models \varphi_1 \to \varphi_2$ syss $\mathcal{M}, s \not\models \varphi_1$ o $\mathcal{M}, s \models \varphi_2$.
7. $\mathcal{M}, s \models \mathrm{A}\,\mathrm{X}\,\varphi$ syss para todo s_1 tal que $s \to s_1$ tenemos $\mathcal{M}, s_1 \models \varphi$. Así, $\mathrm{A}\,\mathrm{X}\,\varphi$ dice: 'en cada siguiente estado'.
8. $\mathcal{M}, s \models \mathrm{E}\,\mathrm{X}\,\varphi$ syss para algún s_1 tal que $s \to s_1$ tenemos $\mathcal{M}, s_1 \models \varphi$. Así, $\mathrm{E}\,\mathrm{X}\,\varphi$ dice: 'en algún siguiente estado'.
9. $\mathcal{M}, s \models \mathrm{A}\,\mathrm{G}\,\varphi$ syss para todas las trayectorias $s_1 \to s_2 \to s_3 \to \ldots$, donde $s = s_1$, y para todo s_i a lo largo de dichas trayectorias, tenemos $\mathcal{M}, s_i \models \varphi$. Es decir: 'para todas las trayectorias de ejecución comenzando en s la propiedad φ se satisface globalmente'.
10. $\mathcal{M}, s \models \mathrm{E}\,\mathrm{G}\,\varphi$ syss existe una trayectoria $s_1 \to s_2 \to s_3 \to \ldots$, donde $s = s_1$, y para todo s_i a lo largo de dicha trayectoria, tenemos $\mathcal{M}, s_i \models \varphi$. Es decir: 'existe una trayectoria de ejecución comenzando en s tal que la propiedad φ se satisface globalmente en tal trayectoria'.
11. $\mathcal{M}, s \models \mathrm{A}\,\mathrm{F}\,\varphi$ syss para todas las trayectorias $s_1 \to s_2 \to s_3 \to \ldots$, donde $s = s_1$, hay un s_i a lo largo de dichas trayectorias tal que $\mathcal{M}, s_i \models \varphi$. Es decir: 'para todas las trayectorias de ejecución comenzando en s habrá algún estado futuro que satisfaga φ'.
12. $\mathcal{M}, s \models \mathrm{E}\,\mathrm{F}\,\varphi$ syss existe una trayectoria $s_1 \to s_2 \to s_3 \to \ldots$, donde $s = s_1$, y hay un s_i a lo largo de dicha trayectoria tal que $\mathcal{M}, s_i \models \varphi$. Es decir: 'existe una trayectoria de ejecución comenzando en s tal que φ se satisface en algún estado futuro'.

13. $\mathcal{M}, s \models \mathrm{A}[\varphi_1 \,\mathrm{U}\, \varphi_2]$ syss para todas las trayectorias $s_1 \to s_2 \to s_3 \to \ldots$, donde $s = s_1$, hay un s_i a lo largo de dichas trayectorias tal que $\mathcal{M}, s_i \models \varphi_2$, y, para toda $j < i$, tenemos $\mathcal{M}, s_j \models \varphi_1$. Es decir: 'todas las trayectorias de ejecución comenzando en s satisfacen que φ_1 se cumple hasta que φ_2 también se cumpla'.

14. $\mathcal{M}, s \models \mathrm{E}[\varphi_1 \,\mathrm{U}\, \varphi_2]$ syss existe una trayectoria $s_1 \to s_2 \to s_3 \to \ldots$, donde $s = s_1$, y en esta trayectoria hay un s_i tal que $\mathcal{M}, s_i \models \varphi_2$, y, para toda $j < i$, tenemos $\mathcal{M}, s_j \models \varphi_1$. Es decir: 'existe una trayectoria de ejecución comenzando en s tal que φ_1 se cumple hasta que φ_2 también se cumpla'.

Las cláusulas 9-14 arriba descritas se refieren a trayectorias de ejecución en los modelos. Es por lo tanto útil visualizar todas las posibles trayectorias de ejecución partiendo de un estado s al desenrollar el sistema de transición para obtener un árbol de computación infinito, de allí la 'lógica de árbol de computación'. Los diagramas en las Figuras 4–7 exhiben esquemas de sistemas cuyos estados iniciales satisfacen las fórmulas $\mathrm{EF}\,\varphi$, $\mathrm{EG}\,\varphi$, $\mathrm{AG}\,\varphi$ y $\mathrm{AF}\,\varphi$, respectivamente.

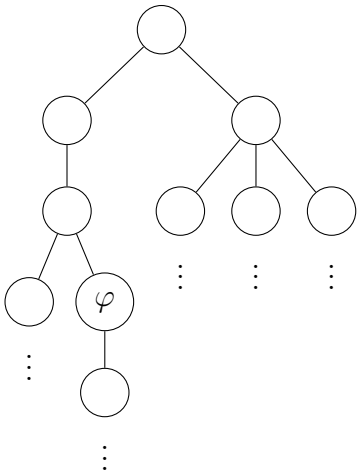


Figura 4: Un sistema cuyo estado inicial satisface $\mathrm{EF}\,\varphi$.

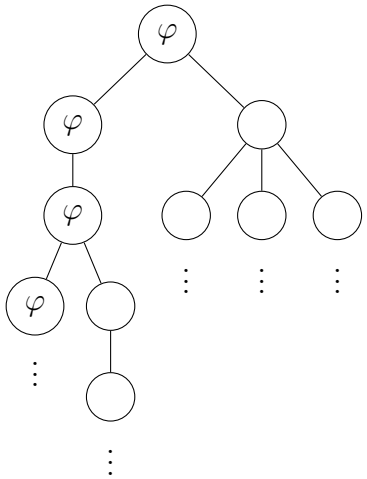


Figura 5: Un estado cuyo estado inicial satisface $\mathrm{EG}\,\varphi$.

Ejercicios 2.3. Sea $\mathcal{M}$ el modelo de la Figura 1. Determine si se satisfacen las siguientes relaciones:

1. $\mathcal{M}, s_0 \models \mathrm{EX}\,q$
2. $\mathcal{M}, s_0 \models \mathrm{EXAG}\,r$
3. $\mathcal{M}, s_0 \models \mathrm{AGAX}\,r$
4. $\mathcal{M}, s_0 \models \mathrm{EFAG}\,r$
5. $\mathcal{M}, s_0 \models \mathrm{EG}\,(p \vee q) \to \mathrm{AG}\,(p \wedge r)$
6. $\mathcal{M}, s_0 \models \mathrm{AG}\,(p \vee q) \to \mathrm{AG}\,r$
7. $\mathcal{M}, s_0 \models \mathrm{E}[(\neg p \vee q)\,\mathrm{U}\,(p \wedge \neg q)]$
8. $\mathcal{M}, s_0 \models \neg\big(\mathrm{A}[(\neg p \vee q)\,\mathrm{U}\,\neg(q \wedge r)]\big)$

2.3. Patrones prácticos de especificación

Es útil considerar algunos ejemplo típicos de fórmulas, y comparar la especificación en CTL con la de LTL. Supongamos que el conjunto de fórmulas atómicas incluye variables como admitida e iniciado.

- Para cualquier estado, si una solicitud para algún recurso se presenta, entonces será eventualmente admitida:

$$\mathrm{AG}\,(\mathsf{solicitud} \to \mathrm{AF}\,\mathsf{admitida}).$$

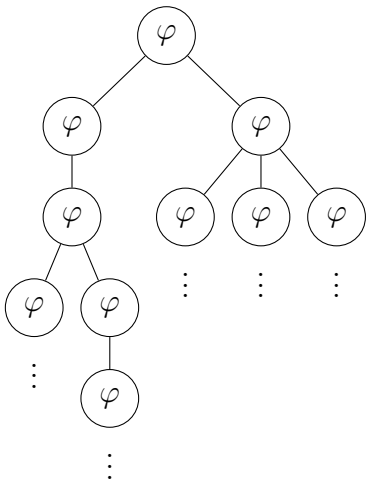


Figura 6: Un sistema cuyo estado inicial satisface $\mathrm{A\,G}\,\varphi$.

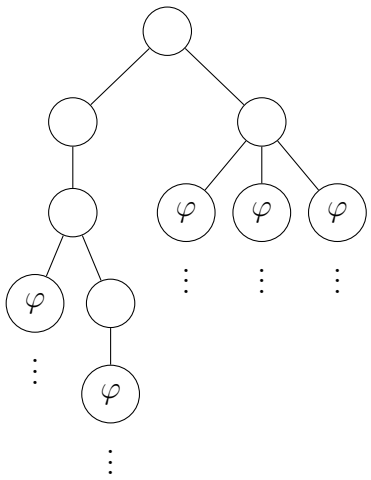


Figura 7: Un estado cuyo estado inicial satisface $\mathrm{A\,F}\,\varphi$.

- Un cierto proceso se habilita de manera infinitamente frecuente sobre toda trayectoria de ejecución:

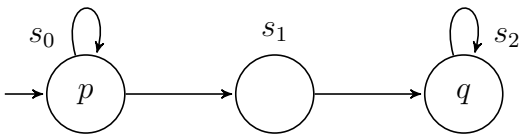
$$\mathrm{A\,G\,(A\,F}\,\mathsf{habilita})$$

- Sin importar lo que ocurra, cierto proceso eventualmente estará permanentemente bloqueado:

$$\mathrm{A\,F\,(A\,G}\,\mathsf{bloqueado})$$

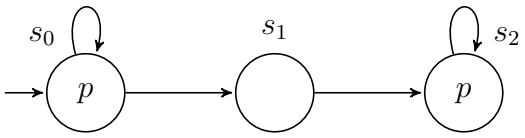
- La propiedad que dice que si el proceso se habilita de manera infinitamente frecuente, entonces se ejecutara de manera infinitamente frecuente, no es expresable en CTL. En particular, no se expresa por $\mathrm{AG\,AF}\,\mathsf{habilita} \to \mathrm{AG\,AF}\,\mathsf{ejecutara}$, ni por cualquier otro uso de A ni E en la fórmula LTL correspondiente. La fórmula CTL anterior expresa que si ***cualquier trayectoria*** habilita al proceso de manera infinitamente frecuente, entonces ***cualquier trayectoria*** ejecutara al proceso de manera infinitamente frecuente; lo cual es más débil que aseverar que *cualquier trayectoria* que habilita al proceso de manera infinitamente frecuente lo ejecutara de manera infinitamente frecuente.

  Para ver que $\mathrm{G\,F}\,p \to \mathrm{G\,F}\,q$ (en LTL) no es lo mismo a $\mathrm{AG\,AF}\,p \to \mathrm{AG\,AF}\,q$ (en CTL) consideremos el siguiente modelo:



  La fórmula CTL $\mathrm{AG\,AF}\,p \to \mathrm{AG\,AF}\,q$ es verdadera trivialmente, porque $\mathrm{AG\,AF}\,p$ no se satisface en la trayectoria $s_0 \to s_1 \to s_2 \to s_2 \to \ldots$. La fórmula LTL $\mathrm{G\,F}\,p \to \mathrm{G\,F}\,q$ no es verdadera porque la trayectoria que se cicla infinitamente en s_0 satisface $\mathrm{G\,F}\,p$ pero no $\mathrm{G\,F}\,q$.

  ¿Qué sucede con la fórmula LTL $\mathrm{F\,G}\,p$ y con la fórmula CTL $\mathrm{AF\,AG}\,p$, bajo el sistema que parece a continuación? ¿Las fórmulas se evalúan a lo mismo?



- Es posible llegar a un estado iniciado sin que esté listo:

$$\mathrm{E}\,\mathrm{F}\,(\mathsf{iniciado} \wedge \neg \mathsf{listo})$$

  Para expresar la imposibilidad de esta propiedad, simplemente se niega la fórmula.

- El elevador *puede* permanecer inactivo en el tercer piso con sus puertas cerradas:

$$\mathrm{A}\,\mathrm{G}\,(\mathsf{piso3} \wedge \mathsf{inactivo} \wedge \mathsf{puertacerrada} \rightarrow \mathrm{E}\,\mathrm{G}\,(\mathsf{piso3} \wedge \mathsf{inactivo} \wedge \mathsf{puertacerrada})).$$

- Un proceso siempre puede solicitar entrar a su sección crítica. Recordemos que esta propiedad no es expresable en LTL. Usando las proposiciones de la Figura 3, la propiedad en cuestión se expresa en CTL como:

$$\mathrm{A}\,\mathrm{G}\,(n_1 \rightarrow \mathrm{E}\,\mathrm{X}\,t_1).$$

2.4. Equivalencias importantes entre fórmulas CTL

Definición 2.4. Dos fórmulas de CTL φ y ψ son semánticamente equivalentes si cualquier estado en cualquier modelo llega a satisfacer a una de ellas, entonces también debe satisface a la otra; y escribimos $\varphi \equiv \psi$.

Nos hemos percatado que A es un cuantificador universal sobre trayectorias y E es el correspondiente cuantificador existencial. Además, G y F son también cuantificadores universal y existencial, respectivamente, extendiéndose sobre los estados presentes en una trayectoria particular. Con esto, no es difícil concluir por las leyes de De Morgan que se satisface:

$$\neg \mathrm{E}\,\mathrm{F}\,\varphi \equiv \mathrm{A}\,\mathrm{G}\,\neg\varphi \tag{2.4.1}$$
$$\neg \mathrm{A}\,\mathrm{F}\,\varphi \equiv \mathrm{E}\,\mathrm{G}\,\neg\varphi \tag{2.4.2}$$
$$\neg \mathrm{A}\,\mathrm{X}\,\varphi \equiv \mathrm{E}\,\mathrm{X}\,\neg\varphi$$

También tenemos las siguientes equivalencias:

$$\mathrm{A}\,\mathrm{F}\,\varphi \equiv \mathrm{A}[\top\,\mathrm{U}\,\varphi] \qquad\qquad \mathrm{E}\,\mathrm{F}\,\varphi \equiv \mathrm{E}[\top\,\mathrm{U}\,\varphi]$$

que son similares a las equivalencias correspondientes en LTL.

2.5. Conjunto adecuado de conectivos de CTL

Así como en la lógica proposicional y en LTL existe alguna redundancia entre los conectivos, en CTL también se presenta esto. Por ejemplo, el conectivo A X puede ser escrito como $\neg \mathrm{E}\,\mathrm{X}\,\neg$; y A G , A F , E G y E F pueden ser escritos en términos de A U y E U como sigue: primero, reescribimos $\mathrm{A}\,\mathrm{G}\,\varphi$ como $\neg \mathrm{E}\,\mathrm{F}\,\neg\varphi$ y $\mathrm{E}\,\mathrm{G}\,\varphi$ como $\neg \mathrm{A}\,\mathrm{F}\,\neg\varphi$, esto por las equivalencias 2.4.1 y 2.4.2;

entonces usamos $\mathrm{A}\,\mathrm{F}\,\varphi \equiv \mathrm{A}[\top\,\mathrm{U}\,\varphi]$ y $\mathrm{E}\,\mathrm{F}\,\varphi \equiv \mathrm{E}[\top\,\mathrm{U}\,\varphi]$. Por lo tanto $\mathrm{A}\,\mathrm{U}$, $\mathrm{E}\,\mathrm{U}$ y $\mathrm{E}\,\mathrm{X}$ forman un conjunto adecuado de conectivos temporales.

También $\mathrm{E}\,\mathrm{G}$, $\mathrm{E}\,\mathrm{U}$ y $\mathrm{E}\,\mathrm{X}$ forman un conjunto adecuado, gracias a la equivalencia:

$$\mathrm{A}[\varphi\,\mathrm{U}\,\psi] \equiv \neg\big(\mathrm{E}[\neg\psi\,\mathrm{U}\,(\neg\varphi \wedge \neg\psi)] \vee \mathrm{E}\,\mathrm{G}\,\neg\psi\big).$$

De manera más general, tenemos el siguiente resultado.

Teorema 2.5. *Un conjunto de conectivos temporales en CTL es adecuado si y sólo si contiene al menos uno de* $\{\mathrm{A}\,\mathrm{X}, \mathrm{E}\,\mathrm{X}\}$*, al menos uno de* $\{\mathrm{E}\,\mathrm{G}, \mathrm{A}\,\mathrm{F}, \mathrm{A}\,\mathrm{U}\}$ *y a* $\mathrm{E}\,\mathrm{U}$*.*

Los conectivos temporales $\mathrm{A}\,\mathrm{R}$, $\mathrm{E}\,\mathrm{R}$, $\mathrm{A}\,\mathrm{W}$ y $\mathrm{E}\,\mathrm{W}$ se pueden definir en CTL como sigue:

- $\mathrm{A}[\varphi\,\mathrm{R}\,\psi] = \neg\mathrm{E}[\neg\varphi\,\mathrm{U}\,\neg\psi]$;
- $\mathrm{E}[\varphi\,\mathrm{R}\,\psi] = \neg\mathrm{A}[\neg\varphi\,\mathrm{U}\,\neg\psi]$;
- $\mathrm{A}[\varphi\,\mathrm{W}\,\psi] = \mathrm{A}[\psi\,\mathrm{R}\,(\varphi \vee \psi)]$, luego aplicamos la primer ecuación;
- $\mathrm{E}[\varphi\,\mathrm{W}\,\psi] = \mathrm{E}[\psi\,\mathrm{R}\,(\varphi \vee \psi)]$, luego aplicamos la segunda ecuación.

3. Algoritmo de verificación de modelos para CTL

Como no es posible capturar en una estructura finita el árbol infinito que resulta de desenrollar un sistema de transición, necesitamos tener un entendimiento más profundo de la semántica de CLT, lo que constituirá las bases de un algoritmo eficiente que, dado $\mathcal{M}$, $s \in S$ y φ, determine si $\mathcal{M}, s \models \varphi$ se cumple o no. Hay varias maneras en la que podemos considerar a

$$\mathcal{M}, s \stackrel{?}{\models} \varphi$$

como un problema computacional. Por ejemplo, podemos tener tanto al modelo $\mathcal{M}$, la fórmula φ y al estado s como entrada; lo que esperaríamos sería una respuesta de la forma 'sí' cuando se cumple $\mathcal{M}, s \models \varphi$, y 'no' en otro caso. Otro enfoque es tener como entrada a $\mathcal{M}$ y a φ, obteniendo como salida al conjunto de *todos* los estados s de $\mathcal{M}$ que satisfacen φ.

Resulta que es más fácil ofrecer un algoritmo para resolver el segundo de estos dos problemas. Esto nos da automáticamente una solución al primer problema, ya que podemos simplemente corroborar si el estado inicial s_0 es un elemento del conjunto de salida.

El algoritmo de etiquetamiento Presentamos ahora un algoritmo que dado un modelo y una fórmula de CTL, regresa el conjunto de estados en los que el modelo satisface la fórmula. No vamos a exigir que el algoritmo maneje cada uno de los conectivos de CTL explícitamente, pues hemos visto que los conectivos $\bot$, $\neg$ y $\wedge$ forman un conjunto adecuado para los conectivos proposicionales; y $\mathrm{A}\,\mathrm{F}$, $\mathrm{E}\,\mathrm{U}$ y $\mathrm{E}\,\mathrm{X}$ forman un conjunto adecuado con respecto a los conectivos temporales. Dada una fórmula arbitraria φ de CTL, simplemente pre-procesaremos φ para escribirla en una forma equivalente en términos del conjunto adecuado de conectivos, entonces llamamos al algoritmo de verificación de modelos. El algoritmo es el siguiente:

Entrada: un sistema de transición $\mathcal{M} = (S, \to, L)$ y una fórmula φ de CTL.
Salida: el conjunto de estados de $\mathcal{M}$ que satisface φ.

Primero, ejecutamos el pre-procesamiento sobre φ mediante la llamada TRADUCIR(φ), así expresamos a φ en términos de los conectivos AF, EU, EX, $\bot$, $\neg$ y $\wedge$ usando las equivalencias dadas anteriormente. Enseguida, etiquetamos a los estados de $\mathcal{M}$ con las subfórmulas de φ que se satisfacen en los estados, iniciando con las subfórmulas más pequeñas y procediendo de manera incluyente hasta obtener a la fórmula φ misma.

Supongamos que ψ es una subfórmula de φ y los estados satisfaciendo *todas* las subfórmulas inmediatas de ψ ya han sido etiquetados. Determinamos por un análisis de casos qué estados etiquetar con ψ. Si ψ es,

- $\bot$: ningún estado es etiquetado con $\bot$.
- p: etiquetamos a s con p si $p \in L(s)$.
- $\psi_1 \wedge \psi_2$: etiquetamos s con $\psi_1 \wedge \psi_2$ si s ya está etiquetado tanto con ψ_1 como con ψ_2.
- $\neg\psi_1$: etiquetamos a s con $\neg\psi_1$ si s no está etiquetado con ψ_1.
- $\mathrm{AF}\,\psi_1$:
  - Cualquier estado s que esté etiquetado con ψ_1, entonces se etiquetará con $\mathrm{AF}\,\psi_1$.
  - Repetir hasta que no haya más cambios: etiquetar cualquier estado con $\mathrm{AF}\,\psi_1$ si ***todos*** sus estados sucesores están etiquetados con $\mathrm{AF}\,\psi_1$.
- $\mathrm{E}[\psi_1\,\mathrm{U}\,\psi_2]$:
  - Cualquier estado s que esté etiquetado con ψ_2, entonces se etiquetará con $\mathrm{E}[\psi_1\,\mathrm{U}\,\psi_2]$.
  - Repetir hasta que no haya más cambios: etiquetar cualquier estado con $\mathrm{E}[\psi_1\,\mathrm{U}\,\psi_2]$ si está etiquetado con ψ_1 y ***al menos uno*** de sus estados sucesores está etiquetado con $\mathrm{E}[\psi_1\,\mathrm{U}\,\psi_2]$.
- $\mathrm{EX}\,\psi_1$: etiquetamos cualquier estado con $\mathrm{EX}\,\psi_1$ si ***uno*** de sus sucesores está etiquetado con ψ_1.

Una vez ejecutado el etiquetamiento para cada una de las subfórmulas de φ (incluyendo a φ mismo), el algoritmo regresa como salida los estados que están etiquetados con φ.

La complejidad de este algoritmo es $\mathcal{O}(f \cdot V \cdot (V + E))$, donde f es el número de conectivos en la fórmula, V es el número de estados y E es el número de transiciones; el algoritmo es lineal en el tamaño de la fórmula y cuadrático en el tamaño del modelo.

Manejando EG directamente En lugar de usar un conjunto adecuado mínimo de conectivos, habría sido posible escribir rutinas similares para los otros conectivos. Esto haría al algoritmo más eficiente. No obstante, los conectivos AG y EG requieren un proceder un tanto distinto del que se tiene para los otros. Aquí está el procedimiento para lidiar con $\mathrm{EG}\,\psi_1$ *directamente*:

- $\mathrm{EG}\,\psi_1$:
  - Etiquetamos ***todos*** los estados con $\mathrm{EG}\,\psi_1$.
  - Para ***cualquier*** estado s que *no* esté etiquetado con ψ_1, *borramos* la etiqueta $\mathrm{EG}\,\psi_1$.
  - Repetir hasta que no haya más cambios: *borrar* la etiqueta $\mathrm{EG}\,\psi_1$ de cualquier estado si ***ninguno*** de sus sucesores está etiquetado con $\mathrm{EG}\,\psi_1$.

En el procedimiento anterior, etiquetamos todos los estados con la subfórmula $\mathrm{E}\,\mathrm{G}\,\psi_1$ para luego reducir este conjunto gradualmente, en lugar de construirlo a partir de nada como se hizo en el caso de $\mathrm{E}[\psi_1\,\mathrm{U}\,\psi_2]$. En realidad no hay diferencia en el resultado final entre el procedimiento para $\mathrm{E}\,\mathrm{G}\,\psi_1$ que acabamos de dar y el que se habría hecho al traducir a $\mathrm{E}\,\mathrm{G}\,\psi_1$ a $\neg\mathrm{A}\,\mathrm{F}\,\neg\psi_1$.

Para determinar qué estados se etiquetan con $\mathrm{A}\,\mathrm{G}\,\psi_1$ podemos seguir un procedimiento similar al de $\mathrm{E}\;\mathrm{G}\;\psi_1$. Esto, sin embargo, se puede ahorrar si el sistema de transición es tal que para *cualesquiera dos estados hay una trayectoria que lleva del uno al otro*, ya que en esta situación basta con verificar que ***todo*** estado está etiquetado con ψ_1 para etiquetar a todos los estados con $\mathrm{A}\,\mathrm{G}\,\psi_1$. De otro modo, ningún estado se etiqueta con $\mathrm{A}\,\mathrm{G}\,\psi_1$.

Ejercicios 3.1. Sobre el modelo abajo representado se ejecutó el algoritmo de etiquetamiento para determinar si la fórmula CTL $\mathrm{A}\,\mathrm{G}\,(p \to \mathrm{A}\,\mathrm{F}\,q)$ se satisface en el estado inicial s_0. Lo primero fue encontrar una fórmula equivalente usando únicamente los operadores $\mathrm{A}\,\mathrm{F}$, $\mathrm{E}\,\mathrm{U}$, $\mathrm{E}\,\mathrm{X}$, $\bot$, $\neg$ y $\wedge$; sin embargo, dejaremos a los operadores $\mathrm{A}\,\mathrm{G}$ y $\mathrm{E}\,\mathrm{G}$ ya que podemos lidiar con ellos directamente como lo indican los párrafos anteriores. El resultado del pre-procesamiento es: $\mathrm{A}\,\mathrm{G}\,\big(\neg(p \wedge \mathrm{E}\,\mathrm{G}\,\neg q)\big)$. A un lado de los estados están las etiquetas que resultan de ejecutar el algoritmo donde ψ es $\neg(p \wedge \mathrm{E}\;\mathrm{G}\;\neg q)$. Concluimos que todos los estados satisfacen a $\mathrm{A}\;\mathrm{G}\;\big(\neg(p \wedge \mathrm{E}\;\mathrm{G}\;\neg q)\big)$, por lo que $\mathrm{A}\,\mathrm{G}\,(p \to \mathrm{A}\,\mathrm{F}\,q)$ sí se satisface en s_0.