Bases Formales de la Computación

Gerardo M. Sarria M.

Pontificia Universidad Javeriana

3 de abril de 2009

LÓGICAS MODALES

Contenido

¿Qué es la lógica Modal?

- Variedad de diferentes sistemas
- Dificultad para dar una definición que cubra a todos
- Respuesta superficial: una lógica que tiene una modalidad o muchas modalidades en ella.

¿Qué es la lógica Modal?

- Variedad de diferentes sistemas
- Dificultad para dar una definición que cubra a todos
- Respuesta superficial: una lógica que tiene una modalidad o muchas modalidades en ella.

Una modalidad es un conectivo que toma una fórmula (o fórmulas) y produce una nueva fórmula con un nuevo significado.

¿Qué es la lógica Modal?

Similar los conectivos lógicos clásicos: $\neg$ toma una fórmula A y produce una nueva fórmula $\neg A$, o $\rightarrow$ toma dos fórmulas A y B y produce una fórmula $A \rightarrow B$.

La única diferencia es que en la lógica clásica, el valor de verdad de $\neg A$ está determinado únicamente por el valor de A, y el valor de $A \rightarrow B$ es una función de los valores de A y B.

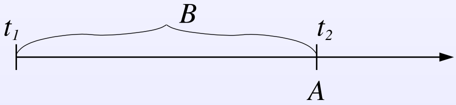
¡Las modalidades no son funciones de verdad!

Ejemplos (Modalidades Unarias)

- $\Box A$: "es necesario que A"
- $\Diamond A$: "es posible que A"
- $\mathrm{G}\ A$: "siempre en el futuro, A será cierto"
- $\mathrm{F}\ A$: "en algún momento en el futuro, A será cierto"
- $\mathrm{P}\ A$: "en algún momento en el pasado, A fue cierto"
- $\mathrm{K}_i\ A$: "el agente i sabe que A"
- $\mathrm{B}_i\ A$: "el agente i cree que A"
- $[prog]A$: "después de cualquier ejecución del programa $prog$, el estado satisface la propiedad A"
- $\langle prog\rangle A$: "existe una ejecución del programa $prog$, cuyo resultado en un estado satisface la propiedad A"

Ejemplos (Modalidades Binarias)

- $A \to B$
- $\mathrm{U}(A, B)$: "hasta que A sea cierta, B es cierta"



Lógica Modal Básica: Sintaxis

Alfabeto:

- Un conjunto de variables proposicionales $Prop = \{p_1, p_2, \ldots, \}$
- Conectivos booleanos $\neg$ y $\rightarrow$ ($\wedge$, $\vee$ y $\leftrightarrow$ se pueden definir)
- Modalidad (unaria) $\Box$ ($\Diamond$ se puede definir)

Una fórmula bien formada:

$$A, B, \ldots ::= p \mid \neg A \mid A \rightarrow B \mid \Box A$$

Definiciones

- $A \lor B = \neg A \to B$
- $A \land B = \neg(A \to \neg B)$
- $A \leftrightarrow B = (A \to B) \land (B \to A)$
- $\Diamond A = \neg \Box \neg A$

Semántica

¿Cómo describir axiomáticamente las fórmulas válidas?

Semántica

¿Cómo describir axiomáticamente las fórmulas válidas?

Respuesta: Las estructuras de los posibles mundos!

La idea es usar grafos $(W, R), R \subseteq W \times W$, como modelos para la lógica modal y pensar en W como el conjunto de posibles mundos y R como una relación alternativa.

$$(W, R, x) \models \Box A \qquad \text{iff} \quad (W, R, y) \models A \qquad \forall y.\ x\ R\ y$$

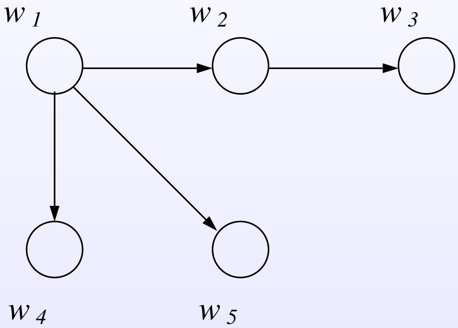
Semántica de Kripke

Una estructura de Kripke es una tupla $M = \langle W, R, V\rangle$, donde

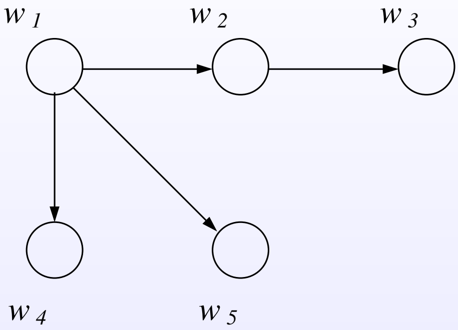
- W es un conjunto no vacío (posibles mundos)
- $R \subseteq W \times W$ es una relación de accesibilidad
- $V : (Prop \times W) \rightarrow \{\texttt{true}, \texttt{false}\}$ es una función de valuación

El grafo proveerá la información de cuáles variables proposicionales serán verdad en cuáles vértices.

Ejemplo

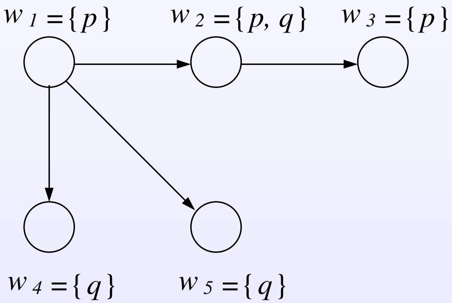


Ejemplo



$$V(p, w_1) = \texttt{true}, V(q, w_1) = \texttt{false}$$
$$V(p, w_2) = \texttt{true}, V(q, w_2) = \texttt{true}$$
$$V(p, w_3) = \texttt{true}, V(q, w_3) = \texttt{false}$$
$$V(p, w_4) = \texttt{false}, V(q, w_4) = \texttt{true}$$
$$V(p, w_5) = \texttt{false}, V(q, w_5) = \texttt{true}$$

Ejemplo

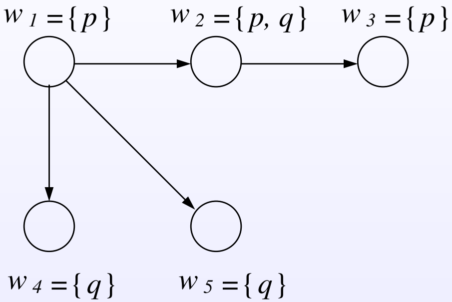


Significado de las Fórmulas

Dado $M = \langle W, R, V \rangle$ y $w \in W$, se define lo que significa para una fórmula ser verdad (noción de satisfacer) en un mundo w de un modelo M:

$$
\begin{array}{lll}
M, w \vDash p & \text{sii} & V(p, w) = \texttt{true} \\
M, w \vDash \neg A & \text{sii} & M, w \nvDash A \\
M, w \vDash A \to B & \text{sii} & M, w \nvDash A \ o \ M, w \vDash B \\
M, w \vDash \Box A & \text{sii} & \text{para todo } v \text{ accesible desde } w \\
 & & (\forall v \ t.q. \ R(w, v)), M, v \vDash A
\end{array}
$$

Ejemplo



$$M, w_1 \vDash \Box q$$
$$M, w_1 \vDash \neg \Box p$$
$$M, w_1 \vDash \neg \Box \neg p$$
$$M, w_1 \vDash \Diamond p$$
$$M, w_1 \vDash \Diamond \Box p$$

Validez (y Satisfacibilidad)

- Una fórmula A es cierta en un modelo M si se satisface en todos los mundos de M.
- Una fórmula A es válida si es cierta en todos los modelos.
- Una fórmula es satisfacible si su negación no es válida (si se satisface en por lo menos un mundo de un modelo)

Validez (y Satisfacibilidad)

Ejemplos:

- $\Box p \to \Box p$ es válido (tautología proposicional)
- $\Box(p \to p)$ es válido (porque $p \to p$ es verdad en todos lo mundos accesibles, donde sea que se esté)
- $\Box p \to p$ no es válido (el conjunto $\{\Box p, \neg p\}$ es satisfacible en algunos mundos).

¿Qué se puede expresar en la lógica modal?

Utilidades:

- los posibles mundos son estados en una computación,
- R es una relación de transición,
- V nos dice cuáles propiedades son ciertas en cuál estado.

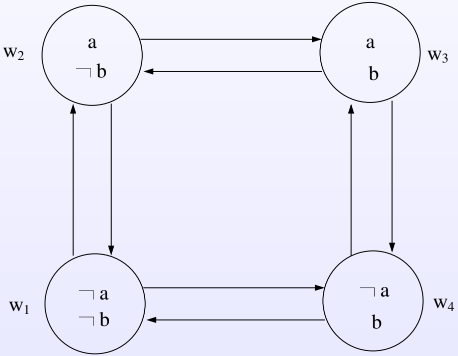
¿Qué se puede expresar en la lógica modal?

Ejemplo:

- Suponga que se tienen dos procesos/agentes A y B.
- Cada uno tiene una variable booleana local (A tiene a, B tiene b).
- Todo lo que hacen es: cambian el valor de su variable; se suspenden; luego vuelven a cambiar el valor de nuevo.
- Se asume que sus acciones son intercaladas (no se ejecutan simultaneamente)

¿Qué se puede expresar en la lógica modal?

Ejemplo:



¿Qué se puede expresar en la lógica modal?

¿Qué podemos afirmar de este sistema en la lógica modal?

¿Qué se puede expresar en la lógica modal?

¿Qué podemos afirmar de este sistema en la lógica modal?

- $\Diamond \neg a \ \wedge \ \Diamond a$

¿Qué se puede expresar en la lógica modal?

¿Qué podemos afirmar de este sistema en la lógica modal?

- $\Diamond\neg a \;\wedge\; \Diamond a$
- $\Diamond\neg b \;\wedge\; \Diamond b$

¿Qué se puede expresar en la lógica modal?

¿Qué podemos afirmar de este sistema en la lógica modal?

- $\Diamond\neg a \ \wedge \ \Diamond a$
- $\Diamond\neg b \ \wedge \ \Diamond b$
- $a \ \wedge \ b \rightarrow \Box(\neg a \ \vee \ \neg b)$

¿Qué se puede expresar en la lógica modal?

¿Qué podemos afirmar de este sistema en la lógica modal?

- $\Diamond\neg a \ \wedge \ \Diamond a$
- $\Diamond\neg b \ \wedge \ \Diamond b$
- $a \ \wedge \ b \rightarrow \Box(\neg a \ \vee \ \neg b)$
- $a \ \wedge \ b \rightarrow \Diamond\Diamond(a \ \wedge \ b)$

¿Qué se puede expresar en la lógica modal?

¿Qué podemos afirmar de este sistema en la lógica modal?

- $\Diamond\neg a \ \wedge \ \Diamond a$
- $\Diamond\neg b \ \wedge \ \Diamond b$
- $a \ \wedge \ b \rightarrow \Box(\neg a \ \vee \ \neg b)$
- $a \ \wedge \ b \rightarrow \Diamond \Diamond (a \ \wedge \ b)$

Básicamente, cuales estados se pueden alcanzar y en cuantos pasos.

¿Qué se puede expresar en la lógica modal?

¿Qué NO podemos afirmar?

¿Qué se puede expresar en la lógica modal?

¿Qué NO podemos afirmar?

- No podemos decir que algo es alcanzable en principio: tenemos que decir "alcanzable en n pasos".

¿Qué se puede expresar en la lógica modal?

¿Qué NO podemos afirmar?

- No podemos decir que algo es alcanzable en principio: tenemos que decir "alcanzable en n pasos".
- No podemos decir cual acción (por cual proceso) no llevará a cual estado.

¿Qué se puede expresar en la lógica modal?

¿Qué NO podemos afirmar?

- No podemos decir que algo es alcanzable en principio: tenemos que decir "alcanzable en n pasos".
- No podemos decir cual acción (por cual proceso) no llevará a cual estado.
- No podemos decir "existe una ejecución que empieza en w_1 donde b es siempre falso".

¿Qué se puede expresar en la lógica modal?

¿Qué NO podemos afirmar?

- No podemos decir que algo es alcanzable en principio: tenemos que decir "alcanzable en n pasos".
- No podemos decir cual acción (por cual proceso) no llevará a cual estado.
- No podemos decir "existe una ejecución que empieza en w_1 donde b es siempre falso".
- No podemos decir lo que el agente A conoce del agente B.

Lógica Temporal

El propósito de la lógica temporal es razonar sobre el tiempo (en filosofía), y sobre el comportamiento de sistemas que evolucionan en el tiempo (en ciencias de la computación).

Lógica Temporal

El propósito de la lógica temporal es razonar sobre el tiempo (en filosofía), y sobre el comportamiento de sistemas que evolucionan en el tiempo (en ciencias de la computación).

¿Cuál es la estructura del tiempo?

Gerardo M.
Sarria M.

Lógica Temporal

El propósito de la lógica temporal es razonar sobre el tiempo (en filosofía), y sobre el comportamiento de sistemas que evolucionan en el tiempo (en ciencias de la computación).

¿Cuál es la estructura del tiempo?

Un flujo del tiempo es una pareja $(T, <)$, donde T es un conjunto no vacío de puntos de tiempo, y $<$ es una relación binaria e irreflexiva sobre T.

Lógica Temporal

Decisión fundamental: ¿lineal o ramificada?.

Lógica Temporal

Decisión fundamental: ¿lineal o ramificada?.

Lineal:
$(T, <)$ es lineal si, para todo $x, y \in T$ con $x \neq y$, se tiene que $x < y$ o $y < x$.

Propiedades:

- Se puede limitar

- *Al pasado*: Existe un $x \in T$ tal que $x \leq y$ para todo $y \in T$ (**genesis**)
- *Al futuro*: Existe un $x \in T$ tal que $x \geq y$ para todo $y \in T$ (**día del juicio**)

Lógica Temporal

Decisión fundamental: ¿lineal o ramificada?.

Lineal:
$(T, <)$ es lineal si, para todo $x, y \in T$ con $x \neq y$, se tiene que $x < y$ o $y < x$.

Propiedades:

- Se puede limitar
- Se puede discretizar

- Si $x \in T$ no es el génesis, entonces existe $y \in T$ tal que $y < x$ y $y < z < x$ no se mantiene para ningún $z \in T$.
- Si $x \in T$ no es el día del juicio, entonces existe $y \in T$ tal que $x < y$ y $x < z < y$ no se mantiene para ningún $z \in T$.

Lógica Temporal

Decisión fundamental: ¿lineal o ramificada?.

Lineal:
$(T, <)$ es lineal si, para todo $x, y \in T$ con $x \neq y$, se tiene que $x < y$ o $y < x$.

Propiedades:

- Se puede limitar
- Se puede discretizar
- Tiene densidad

Para todo $x, y \in T$ con $x < y$, existe un $z \in T$ tal que $x < z < y$.

Lógica Temporal

Decisión fundamental: ¿lineal o ramificada?.

Lineal:
$(T, <)$ es lineal si, para todo $x, y \in T$ con $x \neq y$, se tiene que $x < y$ o $y < x$.

Propiedades:

- Se puede limitar
- Se puede discretizar
- Tiene densidad
- Completitud (tiene un mínimo límite superior)

Gerardo M. Sarria M.

Ramificada:

- al futuro
- al pasado

Lógica Temporal

Ramificada:

- al futuro
- al pasado

Opción más popular: lineal al pasado y ramificada al futuro, i.e.

para cada $x \in T$, el conjunto $\{y \in T \mid y < x\}$ es lineal ordenado por $<$

Lógica Temporal

¿Qué flujo de tiempo tenemos que usar?

Lógica Temporal

¿Qué flujo de tiempo tenemos que usar?

¡Depende de la aplicación!

Sistemas Reactivos

La aplicación principal de la lógica temporal en ciencias de la computación es la verificación de sistemas concurrentes y reactivos de estado finito.

Ejemplos: microprocesadores, sistemas operativos, protocolos de redes, software de aviación.

La verificación de dichos sistemas es una tarea importante y difícil.

Sistemas Reactivos

Estado-Finito. Un estado es una "foto" del sistema, que captura los valores de las variables en un instante de tiempo. Sistemas de estado-finito solo pueden tener un número finito de estados.

Sistemas Reactivos

Estado-Finito. Un estado es una “foto” del sistema, que captura los valores de las variables en un instante de tiempo. Sistemas de estado-finito solo pueden tener un número finito de estados.

Sistema Reactivo. Interactua con el ambiente de manera frecuente y usualmente no termina.

Sistemas Reactivos

Estado-Finito. Un estado es una “foto” del sistema, que captura los valores de las variables en un instante de tiempo. Sistemas de estado-finito solo pueden tener un número finito de estados.

Sistema Reactivo. Interactua con el ambiente de manera frecuente y usualmente no termina.

Sistema Concurrente. Consiste en múltiples procesos que interaccionan entre sí. Un proceso no conoce el estado interno de los otros. Puede ser visto como una colección de sistemas reactivos.

Sistemas Reactivos

Estado-Finito. Un estado es una "foto" del sistema, que captura los valores de las variables en un instante de tiempo. Sistemas de estado-finito solo pueden tener un número finito de estados.

Sistema Reactivo. Interactua con el ambiente de manera frecuente y usualmente no termina.

Sistema Concurrente. Consiste en múltiples procesos que interaccionan entre sí. Un proceso no conoce el estado interno de los otros. Puede ser visto como una colección de sistemas reactivos.

Verificación. Dada la descripción (formal) de un sistema y su comportamiento esperado, se chequea si el sistema de verdad cumple con este comportamiento.

Estructuras de Kripke

Sea PL un conjunto finito de *letras proposicionales*. Una estructura Kripke sobre PL es una tupla $K = \langle S, S_I, R, V \rangle$ con

- S un conjunto no vacío de estados,
- $S_I \subseteq S$ un conjunto de estados iniciales,
- $R \subseteq S \times S$ una relación de transición que es total, i.e. para cada estado $s \in S$, existe un estado $s' \in S$ tal que $s \; R \; s'$,
- $V : S \to 2^{PL}$ una función valuación.

Estructuras de Kripke

Ejemplo:
Considere el siguiente protocolo de exclusión mutua:

```
      task body ProcA is
      begin
        loop
(0)       Non_Critical_Section_A;
(1)       loop exit when Turn = 0; end loop;
(2)       Critical_Section_A;
(3)       Turn := 1;
        end loop;
      end ProcA;
```

Estructuras de Kripke

```
        task body ProcB is
        begin
          loop
(0)         Non_Critical_Section_B;
(1)         loop exit when Turn = 1; end loop;
(2)         Critical_Section_B;
(3)         Turn := 0;
          end loop;
        end ProcB;
```

Se asume que los procesos corren de manera asíncrona. El orden de ejecución es indeterminado.

Estructuras de Kripke

Entonces definimos un conjunto de letras proposicionales:

$$
\begin{aligned}
PL = & \{(T = i) \mid i \in \{0,1\}\} \cup \\
& \{(X = i) \mid X \in \{A, B\} \ \wedge \ i \in \{0,1,2,3\}\}
\end{aligned}
$$

Intuitivamente, $(T = i)$ significa que `Turn` se le ha asignado i, y $(X = i)$ significa que el proceso X está en la línea i.

Estructuras de Kripke

Luego definimos una estructura Kripke $K = \langle S, S_I, R, V \rangle$ de la siguiente manera:

- $S = \{0,1\} \times \{0,1,2,3\} \times \{0,1,2,3\}$
- $S_I = \{(0,0,0),(1,0,0)\}$
- $R = R_A \cup R_B$, donde
  - $R_A = \{(t, p_A, p_B), (t', p'_A, p'_B) \mid p_B = p'_B \ y$
    1. $p_A \in \{0,2,3\} \implies p'_A = p_A + 1 \mod 4 \ \wedge \ t' = t$
    2. $t = 0 \ \wedge \ p_A = 1 \implies p'_A = 2$
    3. $t = 1 \ \wedge \ p_A = 1 \implies p'_A = 1$
    4. $p_A = 3 \implies t' = 1\}$
  - R_B es definido de la misma manera
- $V(t, p_A, p_B) = \{(T = t), (A = p_A), (B = p_B)\}, \forall (t, p_A, p_B) \in S$

Estructuras de Kripke

Una computación de K es una secuencia infinita $s_0 s_1 \cdots$ de estados tal que $s_0 \in S_I$ y $s_i \; R \; s_{i+1}$ para todo $i \geq 0$.

Ejemplo:

$$(0,0,0),(0,1,0),(0,1,1),(0,2,1),(0,3,1),(1,0,1),(1,0,2),\ldots$$

Dicha computación corresponde a una ejecución (asíncrona) del sistema concurrente con los procesos A y B.

Estructuras de Kripke

Propiedades interesantes derivadas del ejemplo:

- Exclusión mutua: ¿pueden A y B estar en la línea (2) al mismo tiempo?

Estructuras de Kripke

Propiedades interesantes derivadas del ejemplo:

- Exclusión mutua: ¿pueden A y B estar en la línea (2) al mismo tiempo? (cierto)

Estructuras de Kripke

Propiedades interesantes derivadas del ejemplo:

- Exclusión mutua: ¿pueden A y B estar en la línea (2) al mismo tiempo? (cierto)
- Accesibilidad garantizada: si el proceso $X \in \{A, B\}$ está en la línea (2), ¿se garantiza que finalmente llegará a la línea (3)?

Estructuras de Kripke

Propiedades interesantes derivadas del ejemplo:

- Exclusión mutua: ¿pueden A y B estar en la línea (2) al mismo tiempo? (cierto)
- Accesibilidad garantizada: si el proceso $X \in \{A, B\}$ está en la línea (2), ¿se garantiza que finalmente llegará a la línea (3)? (cierto, pero solo en computaciones que ejecutan ambos procesos A y B de manera infinita)

Estructuras de Kripke

Las estructuras de Kripke pueden ser no deterministas, i.e. para un $s \in S$, el conjunto $\{s' | s \; R \; s'\}$ puede tener una cardinalidad arbitraria.

En general hay más de una computación.

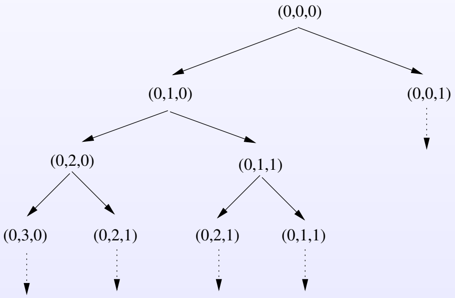
Podemos organizarlas todas en un árbol de computación.

Informalmente, para $s \in S_I$, el árbol (infinito) de computación $T(K, s)$ de K en $s \in S$ es construido inductivamente así:

- use s como la raíz;
- para cada hoja s', agregue un sucesor $\{t \in S | s' \; R \; t\}$.

Estructuras de Kripke

El árbol de computación para el ejemplo anterior que empieza en $(0, 0, 0)$ es:



Para verificar propiedades se consideran las computaciones simples o el árbol entero.

Sintaxis

El conjunto de fórmulas en la lógica temporal lineal (LTL) es el menor conjunto donde

- cada letra proposicional $p \in PL$ es una fórmula,
- si A y B son fórmulas, entonces $A \wedge B$ y $\neg A$ también lo son,
- si A y B son fórmulas, entonces $\circ A$ y $A \mathcal{U} B$ también lo son.

Semántica

Una estructura LTL M es una secuencia infinita $S_0 S_1 \cdots$ con $S_i \in 2^{PL}$ para todo $i \geq 0$.

Una fórmula LTL en M en un tiempo $n \in \mathbb{N}$ se satisface en los siguientes casos:

$$
\begin{array}{lll}
M, n \models p & \text{sii} & p \in S_n, \forall p \in PL \\
M, n \models \neg A & \text{sii} & M, n \not\models A \\
M, n \models A \wedge B & \text{sii} & M, n \models A \text{ y } M, n \models B \\
M, n \models \circ A & \text{sii} & M, n+1 \models A \\
M, n \models A \mathcal{U} B & \text{sii} & \exists m \geq n : M, m \models A \wedge \\
 & & \forall k \in \{n, \ldots, m-1\} : M, k \models B
\end{array}
$$

Abreviaciones

- El Diamante futuro

Abreviaciones

- El Diamante futuro
  $\Diamond A := \texttt{true}\,\mathcal{U}\,A$

  $M, n \vDash \Diamond A$ sii $\exists m \geq n : M, m \vDash A$

Abreviaciones

- El Diamante futuro
  $\Diamond A := \texttt{true}\,\mathcal{U}\,A$

  $M, n \vDash \Diamond A$ sii $\exists m \geq n : M, m \vDash A$

- La caja futuro

Abreviaciones

- El Diamante futuro
  $\Diamond A := \texttt{true}\,\mathcal{U}\,A$

  $$M, n \models \Diamond A \text{ sii } \exists m \geq n : M, m \models A$$

- La caja futuro
  $\Box A := \neg \Diamond \neg A$

  $$M, n \models \Box A \text{ sii } \forall m \geq n : M, m \models A$$

Gerardo M. Sarria M.

Abreviaciones

Ejercicio:
¿Cómo se expresaría el operador release $\mathcal{R}$?

$A \,\mathcal{R}\, B$, significa que B siempre es cierto a menos que sea liberado (*released*) por A.

Abreviaciones

Ejercicio:
¿Cómo se expresaría el operador release $\mathcal{R}$?

$A \mathcal{R} B$, significa que B siempre es cierto a menos que sea liberado (*released*) por A.

$A \mathcal{R} B := \neg(\neg A \mathcal{U} \neg B)$

Abreviaciones

Ejercicio:
¿Cómo se expresaría el operador release $\mathcal{R}$?

$A \mathcal{R} B$, significa que B siempre es cierto a menos que sea liberado (*released*) por A.

$A \mathcal{R} B := \neg(\neg A \mathcal{U} \neg B)$

$M, n \vDash A \mathcal{R} B$ sii $\forall m \geq n : M, k \vDash A \ \forall k < m \implies M, m \vDash B$

Equivalencias

Algunas equivalencias importantes:

$\neg \circ A$	$\equiv$	$\circ \neg A$	auto-dualidad del next
$\Diamond \Diamond A$	$\equiv$	$\Diamond A$	idempotencia del diamante
$\circ \Diamond A$	$\equiv$	$\Diamond \circ A$	conmutación de next con diamante
$A \mathcal{U} B$	$\equiv$	$\neg(\neg A \mathcal{R} \neg B)$	until y release son duales
$A \mathcal{U} B$	$\equiv$	$B \vee (A \wedge \circ(A \mathcal{U} B))$	desarrollo del until
$A \mathcal{R} B$	$\equiv$	$(A \wedge B) \vee (B \wedge \circ(A \mathcal{R} B))$	desarrollo del release

Propiedades Temporales

Una propiedad temporal es un conjunto de estructuras LTL (aquellas en las cuales la propiedad es cierta).

Entonces una propiedad temporal P puede ser definida usando una fórmula A:

$$P = \{M \mid M, 0 \models A\}$$

Propiedades Temporales

Una propiedad temporal es un conjunto de estructuras LTL (aquellas en las cuales la propiedad es cierta).

Entonces una propiedad temporal P puede ser definida usando una fórmula A:

$$P = \{M \mid M, 0 \vDash A\}$$

Dada una estructura de Kripke K que representa un sistema reactivo y una fórmula LTL A que representa una propiedad temporal, *K satisface A* si $M, 0 \vDash A$ para todas las trazas M de K. Notación: $K \vDash A$.

Propiedades Temporales: Safety

Intuitivamente, una propiedad *safety* afirma que "nada malo pasa".

- *Exclusión mutua*: E.g.

$$\Box\neg((A = 2) \wedge (B = 2))$$

- *No Deadlocks*: En cualquier momento algún proceso debe estar activo:

$$\Box(activo_1 \vee \ldots \vee activo_k)$$

- *Correctitud parcial*: Si A se satisface cuando el programa empieza, entonces B será satisfecho si el programa alcanza un determinado estado:

$$A \to \Box(dist \to B)$$

donde $dist \in PL$ marque el determinado estado.

Propiedades Temporales: Liveness

Intuitivamente, una propiedad *liveness* afirma que "algo bueno pasará".

- *Accesibilidad Garantizada*: E.g.

$$\Box(A = 1 \rightarrow \Diamond(A = 2)) \wedge \Box(B = 1 \rightarrow \Diamond(B = 2))$$

- *Respuesta*: Si se hace una petición, será otorgada:

$$\Box(pet \rightarrow \Diamond(otorg))$$

- *Correctitud Total*: Si A se satisface cuando el programa empieza, el programa termina en un determinado estado donde B se satisface:

$$A \rightarrow \Diamond(dist \wedge B)$$

Propiedades Temporales: Fairness

Cuando se modelan sistemas concurrentes, usualmente es importante hacer algunas suposiciones imparciales o justas.

Se asume que existen k procesos, que $activo_i \in PL$ es cierto en un estado s si el proceso $\#i$ está activo en s, y que $ejecutado_i$ es cierto en un estado s si el proceso $\#i$ ha sido ejecutado para alcanzar s.

Model Checking

El problema de model checking en LTL es el siguiente:

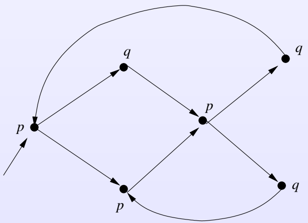
Dada una estructura de Kripke $K = (S, S_I, R, V)$ y una fórmula LTL A, chequear cuándo $K \models A$. (Si todas las trazas M de K satisfacen $M, 0 \models A$).

Model Checking

El problema de model checking en LTL es el siguiente:

Dada una estructura de Kripke $K = (S, S_I, R, V)$ y una fórmula LTL A, chequear cuándo $K \vDash A$. (Si todas las trazas M de K satisfacen $M, 0 \vDash A$).

Ejemplo: La siguiente estructura de Kripke satisface $\square(q \rightarrow \circ\circ\circ p)$. Pero no satisface $\square(p \rightarrow p\,\mathcal{U}\,q)$.



Satisfacibilidad

Una fórmula LTL A es satisfacible si existe una estructura LTL M tal que $M, n \models A$, para algún $n \in \mathbb{N}$.

Dicha estructura es llamada un modelo de A.

Satisfacibilidad

Una fórmula LTL A es satisfacible si existe una estructura LTL M tal que $M, n \models A$, para algún $n \in \mathbb{N}$.

Dicha estructura es llamada un modelo de A.

En verificación, la satisfacibilidad es usada para detectar propiedades contradictorias, i.e. propiedades que no son satisfechas por ninguna computación en ningún sistema reactivo.

Ejercicio:

$$p \wedge \Box(p \rightarrow \circ p) \wedge \Diamond \neg p$$

Fin de la Presentación