

# Lógica modal computacional

LTL – linear temporal logic

Daniel Gorín y Sergio Mera



1er cuatrimestre de 2010

# Propiedades temporales de un sistema

- Pensemos en propiedades de sistemas reactivos.
- Algunas propiedades que interesan en un semáforo:
  - La luz verde y la roja nunca se prenden simultáneamente
  - Una vez en rojo, la luz se volverá verde luego de haber estado amarilla por algún tiempo.
- Dos tipos importantes de propiedades:
  - Safety:** el sistema no ingresa en un estado inválido
  - Liveness:** el sistema siempre responde como debe
- Otra forma de pensarlo:
  - Safety:** alcanza con mirar ejecuciones finitas
  - Liveness:** no alcanza con mirar ejecuciones finitas

# Propositional Linear Temporal Logics (PLTL)

## Sintaxis

$$\text{FORM} ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid X\varphi \mid \varphi U \psi$$

## Los modelos son *trazas*

- Una traza es una secuencia infinita de estados del sistema.
- Es decir,  $\sigma = s_0s_1s_2 \dots$  y cada  $s_i$  es una valuación prop.
- Se puede pensar como un modelo de Kripke con  $R$  lineal.

## Semántica

Sea  $\sigma = s_0s_1s_2 \dots$  y llamemos  $\sigma^i = s_is_{i+1} \dots$ . Definimos:

$\sigma \models p$	sii	$p \in s_0$
$\sigma \models \neg\varphi$	sii	$\sigma \not\models \varphi$
$\sigma \models \varphi \wedge \psi$	sii	$\sigma \models \varphi$ y $\sigma \models \psi$
$\sigma \models X\varphi$	sii	$\sigma^1 \models \varphi$
$\sigma \models \varphi U \psi$	sii	$\exists j \geq 0. (\sigma^j \models \psi \text{ y } (\forall 0 \leq k < j, \sigma^k \models \varphi))$

# Ejemplos

- Operadores derivados:

$$F\varphi \equiv \top U \varphi$$

$$G\varphi \equiv \neg F\neg\varphi$$

- Vuelta al semáforo
  - La luz verde y la roja nunca se prenden simultáneamente

$$\neg F(\text{green} \wedge \text{red})$$

- Una vez en rojo, la luz se volverá verde luego de haber estado amarilla por algún tiempo entre el rojo y el verde.

$$G(\text{red} \rightarrow (\text{red} U (\text{yellow} \wedge (\text{yellow} U \text{green}))))$$

## A qué llamamos *model-checking*?

*Model-checking* (de trazas) es model-checking

- Dada una traza  $\sigma$  y una fórmula  $\varphi$ , vale  $\sigma \models \varphi$ ?
- En términos de IS, podemos pensarlo como un *monitoreo*.

*Model-checking* (en “verificación”) no es model-checking!

- Sea  $\mathcal{M}$  una abstracción de un sistema  $S$  (ie, un *modelo* de  $S$ )
- Y sea  $\varphi$  una propiedad expresada en LTL.
- Verificar si  $\mathcal{M}$  cumple  $\varphi$  es comprobar si vale:

$\sigma \models \varphi$ , para toda traza (ejecución)  $\sigma$  de  $\mathcal{M}$

- Es decir, si vale  $\mathcal{C} \models \varphi$  con  $\mathcal{C} = \{\sigma \mid \sigma \text{ es una traza de } \mathcal{M}\}$ .
- ¡Esto es validez respecto a una clase de modelos!

## Model-checking, lenguajes y autómatas

- Podemos ver una traza como un *string* sobre alfabeto  $2^{\text{Prop}}$
  - Sea  $\mathcal{L}(\mathcal{M})$  el conjunto de trazas que puede generar  $\mathcal{M}$
  - Y sea  $\mathcal{L}(\varphi) = \{\sigma \mid \sigma \models \varphi\}$ , las trazas que satisfacen  $\varphi$ .
  - Entonces,  $\mathcal{M}$  cumple la propiedad  $\varphi$  sii  $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\varphi)$ .
  - O lo que es equivalente:  $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\varphi)^c = \emptyset$
  - Si los representáramos con autómatas  $A_{\mathcal{M}}$  y  $A_{\varphi}$ 
    - Podemos computar un autómata que acepta  $\mathcal{L}(\mathcal{M})$  y  $\mathcal{L}(\varphi)$
    - Y usar un algoritmo de alcanzabilidad para ver si su lenguaje es vacío
    - Son operaciones polinomiales en tamaño de los autómatas!
- 
- Por propiedades de *liveness*, los strings deben ser *infinitos*
  - Los autómatas “comunes” reconocen strings *finitos*

# Autómatas finitos (FA)

## Repaso

### Definición (FA)

- Un FA es una 5-tupla  $A = \langle \Sigma, S, S_0, \rho, F \rangle$  donde
  - $\Sigma \neq \emptyset$  es el alfabeto (finito)
  - $S$  es el conjunto de estados (finito)
  - $S_0 \subseteq S$  es el conjunto de estados iniciales ( $S_0 \neq \emptyset$ )
  - $\rho : S \times \Sigma \rightarrow 2^S$  es la función de transición
  - $F \subseteq S$  es el conjunto de estados finales
- $A$  es un autómata finito *determinístico* (DFA) si, además:
  - $|S_0| = 1$
  - $|\rho(s, a)| \leq 1$  para todo  $s$  y  $a$

### Definición (Corrida y aceptación)

- Una corrida de  $A$  sobre  $w \in \Sigma^k$  es una  $s \in S^k$  tal que:
  - 1  $s_0 \in S_0$
  - 2  $s_{i+1} \in \rho(s_i, w_i)$
- $A$  acepta  $w$  si una corrida de  $w$  en  $A$ ,  $s_0 \dots s_k$ , cumple  $s_k \in F$

# Autómatas de Büchi (BA)

Un autómata para palabras infinitas

## Definición (BA)

- Un BA es una 5-tupla  $A = \langle \Sigma, S, s_0, \rho, F \rangle$  donde
  - $\Sigma \neq \emptyset$  es el alfabeto (finito)
  - $S$  es el conjunto de estados (finito)
  - $S_0 \subseteq S$  es el conjunto de estados iniciales ( $S_0 \neq \emptyset$ )
  - $\rho : S \times \Sigma \rightarrow 2^S$  es la función de transición
  - $F \subseteq S$  es el conjunto de estados finales
- O sea... es igual! (ídem para determinístico – DBA)

## Definición (Aceptación)

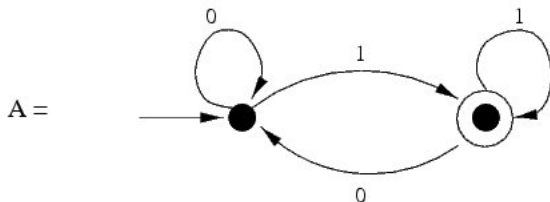
$A$  acepta  $w \in \Sigma^\omega$  si una corrida de  $w$  en  $A$ ,  $s \in S^\omega$ , cumple:

$$\text{inf}(s) \cap F \neq \emptyset$$

donde  $\text{inf}(s) = \{s_i \mid s_i \text{ ocurre infinitas veces en } s\}$

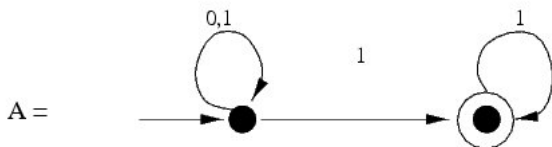


# Ejemplos



- Qué lenguaje reconoce  $A$  como AF?  
 $\mathcal{L}(A) = (0|1)^*1$
- Qué lenguaje reconoce  $A$  como BA?  
 $\mathcal{L}(A) = ((0|1)^*1)^\omega$ ,  
los strings con un número infinito de unos.

# Ejemplos



- Qué lenguaje reconoce  $A$  como FA?  
 $\mathcal{L}(A) = (0|1)^*1$
- Qué lenguaje reconoce  $A$  como BA?  
 $\mathcal{L}(A) = (0|1)^*1^\omega$ ,  
los strings con un número finito de ceros.

# FA vs. BA – parecidos y diferencias

## FA

- Cerrados por  $\cap, \cup$
- Cerrados por  $\cdot^c$
- $\mathcal{L}(FA) = \mathcal{L}(DFA)$
- Decidir si  $\mathcal{L}(A) = \emptyset$  es NLOGSPACE-complete
- Decidir si  $\mathcal{L}(A) = \emptyset^c$  es PSPACE-complete

## BA

- Cerrados por  $\cap$  y  $\cup$
- Cerrados por  $\cdot^c$  (no DBA)
- $\mathcal{L}(BA) \neq \mathcal{L}(DBA)$
- Decidir si  $\mathcal{L}(A) = \emptyset$  es NLOGSPACE-complete
- Decidir si  $\mathcal{L}(A) = \emptyset^c$  es PSPACE-complete

# El problema de emptiness de un BA

## Teorema

Dado un BA  $A = \langle \Sigma, S, S_0, \rho, F \rangle$ ,  $\mathcal{L}(A) \neq \emptyset$  sii existen  $s \in S_0$  y  $t \in F$ , tales que  $t$  es alcanzable desde  $s$  y  $t$  se alcanza a sí mismo.

## Corolario

Dado un BA  $A$ , podemos determinar linealmente si  $\mathcal{L}(A) = \emptyset$ .

## Demostración (Algoritmo)

- Descomponer el BA en Maximal Strongly Connected Components empezando desde  $s \in S_0$  (Alg. de Tarjan)
- Chequear que al menos uno de los MSCC interseca  $F$  en forma no vacía
- Ambos pueden hacerse en tiempo lineal

# Autómatas de Büchi generalizados (GBA)

Más flexibles pero igual de expresivos

## Diferencias en las condiciones de aceptación

- $F \subseteq S$  (BA) vs.  $F = \{F_1, \dots, F_k\}$  con  $F_i \subseteq S$  (GBA)
- En un GBA una corrida  $r$  es aceptada sii
$$\inf(r) \cap F_i \neq \emptyset \text{ para todo } F_i \in F$$

## Teorema

Un GBA es polinomialmente transformable en BA equivalente

## Demostración (Idea)

- Si  $F = \{F_1, \dots, F_k\}$ , tomar como estados  $S \times \{1 \dots k\}$ .
- Si  $(s, a) \mapsto t$  y  $s \in F_i$ ,  $(\langle s, i \rangle, a) \mapsto \langle t, i + 1 \pmod k \rangle$ .
- Si  $(s, a) \mapsto t$  y  $s \notin F_i$ ,  $(\langle s, i \rangle, a) \mapsto \langle t, i \rangle$ .
- $F' = F_k \times \{k\}$

# De STS a autómatas

## Definición (State Transition System – STS)

- Un STS es una tupla  $\langle S, I, R, Label \rangle$  donde
  - $S$  es un conjunto contable no vacío de estados,
  - $I \subseteq S$  es un conjunto de estados iniciales,
  - $R \subseteq S \times S$  es la relación de transición tal que  
 $\forall s \in S. (\exists s' \in S. R(s, s'))$
  - $Label : S \rightarrow 2^{PROP}$  es una valuación que describe cada estado
- Se usan para describir o especificar sistemas

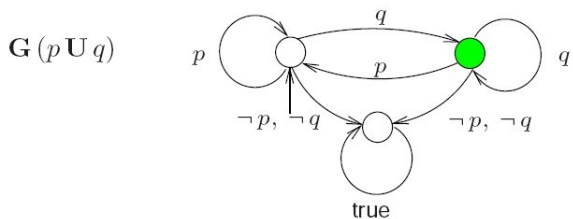
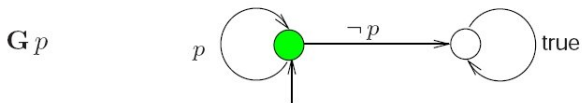
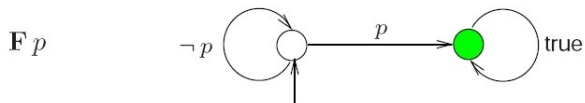
## BA asociado a un STS

- Sea  $S = \langle S, I, R, Label \rangle$  un STS
- Buscamos un BA  $A_S$  que acepte todas y sólo las trazas de  $S$
- Definimos  $A_{\mathcal{M}} = \langle 2^{PROP}, S, I, \rho, S \rangle$  donde

$$\rho(u, a) = \{v \mid R(u, v) \text{ y } a = Label(u)\}$$

# De fórmulas a autómatas

Cuál es la idea?



- Ojo, en las transiciones hay valuaciones!
- Hay muchas traducciones posibles!Cuál es mejor?

# De fórmulas a autómatas, un ejemplo

Los ladrillos

## Definición

$Sub(\varphi)$  es la clausura de  $\varphi$  bajo subfórmulas y neg. simples.

## Definición

Dadas  $\sigma$  y  $\varphi$ ,  $\text{satseq}(\sigma, \varphi) \in (2^{Sub(\varphi)})^\omega$  es la secuencia tal que:

$$\text{satseq}(\sigma, \varphi) = \pi(\sigma, \varphi, 0)\pi(\sigma, \varphi, 1)\pi(\sigma, \varphi, 3) \dots$$

donde  $\pi(\sigma, \varphi, i) = \{\psi \in Sub(\varphi) \mid \sigma^i \models \psi\}$



# De fórmulas a autómatas, un ejemplo

Otra vez los Hintikka sets

Supongamos que  $\sigma \models \varphi$ , entonces se cumple que...

- 1  $\varphi \in \text{satseq}(\sigma, \varphi, 0)$
- 2 si  $p \in \text{Sub}(\varphi)$ ,  $p \in \text{satseq}(\sigma, \varphi, i)$  sii  $p \in \sigma[i]$
- 3 si  $\varphi_1 \wedge \varphi_2 \in \text{Sub}(\varphi)$ ,  $\varphi_1 \wedge \varphi_2 \in \text{satseq}(\sigma, \varphi, i)$  sii  $\varphi_{1,2} \in \sigma$
- 4 si  $\neg\varphi \in \text{Sub}(\varphi)$ ,  $\neg\varphi \in \text{satseq}(\sigma, \varphi, i)$  sii  $\varphi \notin \sigma$
- 5 si  $X\varphi \in \text{Sub}(\varphi)$ ,  $X\varphi \in \text{satseq}(\sigma, \varphi, i)$  sii  $\varphi \in \text{satseq}(\sigma, \varphi, i+1)$
- 6 si  $\varphi_1 U \varphi_2 \in \text{Sub}(\varphi)$ ,  $\varphi_1 U \varphi_2 \in \text{satseq}(\sigma, \varphi, i)$  sii
  - 1  $\varphi_2 \in \text{satseq}(\sigma, \varphi, i)$  ó ( $\varphi_1 \in \text{satseq}(\sigma, \varphi, i)$  y  $\varphi_1 U \varphi_2 \in \text{satseq}(\sigma, \varphi, i+1)$ ), y
  - 2 existe  $j \geq i$  tal que  $\varphi_2 \in \text{satseq}(\sigma, \varphi, j)$

## Definición (Hintikka set para LTL)

Decimos que  $\alpha \subseteq \text{Sub}(\varphi)$  es un Hintikka set para  $\varphi$  si satisface las condiciones 1-4.  $\text{Hin}(\varphi)$  son todos los Hintikka sets para  $\varphi$ .

# De fórmulas a autómatas, un ejemplo

## Construcción del GBA

### Idea

- Dado  $\varphi$ , vamos a construir un GBA  $A_\varphi$  tal que:
  - Acepta una traza  $\sigma$  sii  $\sigma \models \varphi$
  - Una corrida que acepta  $\sigma$ , es una  $\text{satseq}(\sigma, \varphi)$

$$A_\varphi = \langle 2^{\text{PROP}}, \text{Hin}(\varphi), S_0, \rho, \{F_1, \dots, F_k\} \rangle$$

donde

- $S_0 = \{\alpha \mid \alpha \in \text{Hin}(\varphi), \varphi \in \alpha\}$
- $\rho(\alpha, s) = \emptyset$ , si  $s \not\subseteq \alpha$
- $\rho(\alpha, s) = \{\alpha' \mid \alpha' \text{ cumple R1 y R2}\}$ , si  $s \subseteq \alpha$ 
  - R1  $X\varphi \in \alpha$  sii  $\varphi \in \alpha'$
  - R2  $\varphi_1 U \varphi_2 \in \alpha$  sii  $\varphi_2 \in \alpha$  ó  $(\varphi_1 \in \alpha \text{ y } \varphi_1 U \varphi_2 \in \alpha')$
- $\{F_1, \dots, F_k\} = \{F_{\varphi_1 U \varphi_2} \mid \varphi_1 U \varphi_2 \in \text{Sub}(\varphi)\}$ 
  - donde  $F_{\varphi_1 U \varphi_2} = \{\alpha \mid \varphi_1 U \varphi_2 \notin \alpha \text{ o } \varphi_2 \in \alpha\}$