

# Lógica modal computacional

## Decidibilidad y complejidad de lógicas modales (ii)

Daniel Gorín y Sergio Mera



1er cuatrimestre de 2010

# Repaso

## La última vez que nos vimos, vimos que...

- $KAlt_1$  es NP-completa (usando funciones de selección).
- $K$  *no* tiene la propiedad de modelos polinimiales:
  - Dimos una familia de fórmulas satisfacibles  $\varphi_k$
  - Para cada  $k$ ,  $|\varphi_k| \in O(k^3)$
  - $\varphi_k$  fuerza que sus modelos sean árboles binarios completos
  - Luego, todo modelo de  $\varphi_k$  tiene al menos  $2^k$  nodos
- El problema de  $K$ -satisfacibilidad está en PSPACE:
  - Podemos adivinar de a una rama del modelo por vez.
  - Esto lo mostramos usando Hintikka sets.
  - La profundidad de una rama puede ser lineal en la fórmula.
  - Obtuvimos un algoritmo no-det. de espacio polinomial.
  - Y sabíamos que  $PSPACE = NPSPACE$ .

# Lógicas robustas

## Muchas variantes de K también están en PSPACE

- K + nominales y @
- K + counting modalities  $\langle r \rangle_{\geq n} \varphi$
- K + funciones parciales
- K + operadores de pasado  $\langle r \rangle^{-1} \varphi$
- S4 ( $r$  es una relación transitiva)
- ...
- ¡pero cuidado con las combinaciones!

## Los operadores “globales” nos suelen mover a EXPTIME

- K + la modalidad universal A
- K + el operador de clausura transitiva  $\langle r \rangle^* \varphi$
- ...

## ¿Cómo probar si $K$ es PSPACE-completa?

- Necesitamos probar que  $K$  es PSPACE-hard.
- Alcanza con poder reducir polinomialmente un problema PSPACE-completo.
- Usaremos el problema canónico: validez para QBF.

# Quantified Boolean Formulas (QBF)

## Sintaxis

- $\varphi ::= p \mid \neg p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists p\varphi \mid \forall p\varphi$
- *Sentencia*: fórmula sin variables libres
- *Forma prenexa*:  $Q_1p_1 \dots Q_np_n\theta(p_1, \dots, p_n)$ ,  $\theta$  proposicional.

## Semántica

$$\begin{aligned}v \models p &\Leftrightarrow v(p) = 1 \\v \models \neg p &\Leftrightarrow v(p) = 0 \\v \models \varphi \vee \psi &\Leftrightarrow v \models \varphi \text{ ó } v \models \psi \\v \models \varphi \wedge \psi &\Leftrightarrow v \models \varphi \text{ y } v \models \psi \\v \models \exists p\varphi &\Leftrightarrow v[p \mapsto 1] \models \varphi \text{ ó } v[p \mapsto 0] \models \varphi \\v \models \forall p\varphi &\Leftrightarrow v[p \mapsto 1] \models \varphi \text{ y } v[p \mapsto 0] \models \varphi\end{aligned}$$

# Validez de fórmulas de QBF

## Teorema

Decidir la validez de una fórmula de QBF es un problema PSPACE-completo.

## Ejercicio

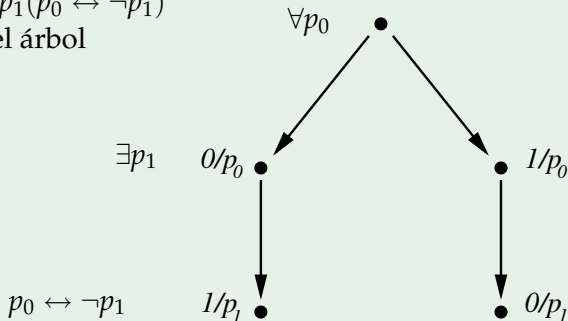
Mostrar que model-checking de lógica de primer orden es PSPACE-completo.

# Satisfacibilidad de K es PSPACE-completa

Idea

Validez de QBF equivale encontrar un árbol binario...

Para  $\forall p_0 \exists p_1 (p_0 \leftrightarrow \neg p_1)$   
tenemos el árbol



¡Y vimos cómo forzar árboles binarios con una fórmula modal!

# Satisfacibilidad de K es PSPACE-completa

Repasemos nuestros ladrillos

- $B_i$  fuerza dos sucesores, uno para cada valor de  $p_i$ :

$$B_i := \diamond p_{i+1} \wedge \diamond \neg p_{i+1}$$

- $S_i$  propaga los valores de  $p_i$  y  $\neg p_i$  al siguiente nivel:

$$S_i := (p_i \rightarrow \Box p_i) \wedge (\neg p_i \rightarrow \Box \neg p_i)$$

- $L_{ki}$  asegura que un nodo esté en el nivel  $i$  y sólo en ese:

$$L_{ki} := \bigwedge_{j \in \{0 \dots k\} \setminus \{i\}} \neg l_j \wedge l_i$$



# Satisfacibilidad de K es PSPACE-completa

La reducción de QBF-validez a K-satisfacibilidad

Dada  $\varphi = Q_1 p_1 \dots Q_k p_k \theta(p_1 \dots p_k)$ ,  $f(\varphi)$  es la conjunción de:

$$\begin{array}{cccccccccccc} L_{k0} & \wedge & \square L_{k1} & \wedge & \square^2 L_{k2} & \wedge & \square^3 L_{k3} & \wedge & \dots & \wedge & \square^{k-1} L_{kk-1} & \wedge & \square^k L_{kk} \\ \diamond T & \wedge & \square \diamond T & \wedge & \square^2 \diamond T & \wedge & \square^3 \diamond T & \wedge & \dots & \wedge & \square^{k-1} \diamond T & & \\ & & \square S_1 & \wedge & \square^2 S_1 & \wedge & \square^3 S_1 & \wedge & \dots & \wedge & \square^{k-1} S_1 & & \\ & & & & \square^2 S_2 & \wedge & \square^3 S_2 & \wedge & \dots & \wedge & \square^{k-1} S_2 & & \\ & & & & & & \square^3 S_3 & \wedge & \dots & \wedge & \square^{k-1} S_3 & & \\ & & & & & & & & & & \vdots & & \\ & & & & & & & & & & \square^{k-1} S_{k-1} & & \end{array}$$

$$\bigwedge_{Q_i = \forall} \square^i B_i \wedge \square^k \theta$$

- Notar que  $f(\varphi)$  es computable en tiempo polinomial

# Satisfacibilidad de K es PSPACE-completa

## Teorema

$\varphi$  es válida en QBF sii  $f(\varphi)$  es K-satisfacible.

## Corolario

Satisfacibilidad de K es PSPACE-completa.

Se puede mostrar un resultado más general

“Toda lógica entre K y S4 es PSPACE-completa”.

$K + A$ , agregamos la modalidad universal.

### Semántica

- $\mathcal{M}, w \models A\varphi$  sii  $\mathcal{M}, v \models \varphi$  para **todo**  $v$
- $\mathcal{M}, w \models E\varphi$  sii  $\mathcal{M}, v \models \varphi$  para **algún**  $v$

- E es un “diamante” y A es un “box”.
- Se pueden pensar como modalidades sobre una relación total.

# Aspectos computacionales de $K + A$

## Model checking

- I. Es decidable
- II. Está en PTIME (e.g., usando programación dinámica)
- III. Es fácil de implementar de manera eficiente

¿Por qué es menos complejo en  $K + A$  que en primer orden?

# Aspectos computacionales de $K + A$

## Satisfacibilidad

- I. Es decidible (reducción a FO2)
- II. ¿Podemos ver que está en PSPACE como hicimos con K?
- III.

# Modelos “exponencialmente profundos” en $K + A$

## Intuición

### Vamos a ver que...

- Para cada  $n > 0$  existe una fórmula  $\kappa_n$  tal que:
  - $\kappa_n$  es satisfacible
  - Todo modelo para  $\kappa_n$  tiene una rama con al menos  $2^n$  nodos

### De donde se concluye que...

- No podemos repetir la prueba de PSPACE para  $K$
- (donde adivinábamos de a una rama del modelo por vez)

# Modelos “exponencialmente profundos” en $K + A$

Sumando en base 2

## Idea para construir $\kappa_n$

- Usamos  $n$  proposiciones  $q_0, \dots, q_{n-1}$ .
- Cada asignación codifica un número entre 0 y  $2^n - 1$
- Queremos que un nodo a nivel  $i$  tenga una asignación que codifique  $i$

## ¿Cómo se suma 1 en binario?

- El caso fácil (el dígito menos significativo es 0):

$$\begin{array}{r} 10011010 \\ + \quad \quad 1 \\ \hline 10011011 \end{array}$$

- El caso general:

$$\begin{array}{r} 10011011 \\ + \quad \quad 1 \\ \hline 10011100 \end{array}$$

# Modelos “exponencialmente profundos” en $K + A$

Ladrillos para armar  $\kappa_n$

## $INC_i$

- Fuerza el valor del siguiente nivel (sumando 1),
- pero sólo si el valor del actual tiene el primer 0 en el bit  $i$

- Caso fácil

$$INC_0 := \neg q_0 \rightarrow (\Box q_0 \wedge \bigwedge_{j>0} ((q_j \rightarrow \Box q_j) \wedge (\neg q_j \rightarrow \Box \neg q_j)))$$

- Caso general

$$INC_{i+1} := (\neg q_{i+1} \wedge \bigwedge_{j=0}^i q_j) \rightarrow \left( \begin{array}{l} \Box(q_{i+1} \wedge \bigwedge_{j=0}^i \neg q_j) \quad \wedge \\ \bigwedge_{l>i+1} (q_l \rightarrow \Box q_l) \quad \wedge \\ \bigwedge_{l>i+1} (\neg q_l \rightarrow \Box \neg q_l) \end{array} \right)$$



# Modelos “exponencialmente profundos” en $K + A$

Finalmente,  $\kappa_n$

Definimos  $\kappa_n$  como

$$(\neg q_{n-1} \wedge \cdots \wedge \neg q_0) \wedge \mathbf{A} \left( \bigwedge_{i=0}^{n-1} \text{INC}_i \right) \wedge \mathbf{A} \diamond \top$$

- $\kappa_n$  tiene tamaño  $\mathcal{O}(n^2)$  pero todo modelo que la satisfaga tiene un camino sin repeticiones de longitud  $2^n$ .
- La misma técnica se puede usar sobre otras modalidades “globales” (e.g., operador de clausura transitiva)

# Satisfacibilidad de $K + A$ está en EXPTIME

## Idea

- Sabemos que si  $\varphi$  es satisfacible, tiene modelo exponencial.
- Veremos que, además:
  - hay una cantidad exponencial de modelos a considerar, y
  - cada uno de estos modelos es exponencial
  - y se puede construir en una cantidad de pasos exponencial.
- Esto nos da un algoritmo determinístico que corre en tiempo exponencial.
- La técnica se llama “eliminación de Hintikka sets”.

# Satisfacibilidad de $K + A$ está en EXPTIME

Hintikka sets – repaso

## Clausura de un conjunto de fórmulas $\Sigma$ ( $Cl(\Sigma)$ )

$$Cl(\Sigma) = \{\varphi \mid \varphi \text{ ocurre en } \Sigma\} \cup \{\bar{\varphi} \mid \varphi \text{ ocurre en } \Sigma\}$$

## Intuición

$Cl(\Sigma)$  es el conjunto de “fórmulas relevantes” de  $\Sigma$ .

## Hintikka sets

Decimos que  $H \subseteq Cl(\Sigma)$  es un *Hintikka set para*  $\Sigma$  si cumple:

- I.  $\varphi \in Cl(\Sigma) \Rightarrow \varphi \in H$  sii  $\bar{\varphi} \notin H$
- II.  $\varphi \wedge \psi \in Cl(\Sigma) \Rightarrow \varphi \wedge \psi \in H$  sii  $\varphi \in H$  y  $\psi \in H$
- III.  $E\varphi \in Cl(\Sigma) \Rightarrow \varphi \in H$  implica  $E\varphi \in H$

# Satisfacibilidad de $K + A$ está en EXPTIME

$Hin_C(\Sigma)$

## Notación

- $\Box(C) = \{\varphi \mid \Box\varphi \in C\}$
- $A(C) = \{\varphi \mid A\varphi \in C\}$
- $Hin(\Sigma) = \{H \mid H \text{ es un Hinitikka set para } \Sigma\}$
- $Hin_C(\Sigma) = \{H \mid H \in Hin(\Sigma) \text{ y } A(H) = C\}$

## Idea

- Para cada  $C \subseteq A(Cl(\Sigma))$ , intentamos armar un modelo  $\mathcal{M}_C$ .
- Si  $\mathcal{M}_C$  está definido, entonces  $\mathcal{M} \models A\varphi \forall \varphi \in C$ .
- La idea es ver que:

$\Sigma$  es satisfacible sii  $\exists C \subseteq A(Cl(\Sigma))$  tal que  $\mathcal{M}_C, w \models \Sigma$

# Satisfacibilidad de $K + A$ está en EXPTIME

## Eliminación de Hintikka sets

### Caso base: $\mathcal{M}_C^0$ .

Dado  $\Sigma$  y  $C \subseteq \mathbf{A}(\text{Cl}(\Sigma))$ , definimos  $\mathcal{M}_C^0 = \langle W_C^0, R_C^0, V_C^0 \rangle$  donde:

- $W_C^0 = \text{Hin}_C(\Sigma)$
- $(H, H') \in R_C^0$  sii  $\forall \varphi \in H', \diamond\varphi \in \text{Cl}(\Sigma)$  implica  $\diamond\varphi \in H$ .
- $V_C^0(p) = \{H \in W_C^0 \mid p \in H\}$

### Paso de eliminación: $\mathcal{M}_C^{n+1}$

- Supongamos que  $\mathcal{M}_C^n$  está definido (i.e.,  $W_C^n \neq \emptyset$ ).
- Decimos que  $H$  es **satisfecho en  $n$**  si, para todo  $\varphi$ :
  - I.  $\diamond\varphi \in H$  implica  $\exists H' \in W_C^n$  tal que  $\varphi \in H'$  y  $(H, H') \in R_C^n$ .
  - II.  $\mathbf{E}\varphi \in H$  implica  $\exists H' \in W_C^n$  tal que  $\varphi \in H'$ .
- $\mathcal{M}_C^{n+1}$ : restricción de  $\mathcal{M}_C^n$  a los  $H \in W_C^n$  satisfechos en  $n$ .

# Satisfacibilidad de $K + A$ está en EXPTIME

Eliminación de Hintikka sets –  $\mathcal{M}_C$

- Como  $Hin_C(\Sigma)$  es finito y  $W^{n+1} \subseteq W^n$ , el proceso converge.
- Pero notar que  $W^{n+1}$  podría estar vacío.
- $\mathcal{M}_C$  es la estructura tal que  $\mathcal{M}^{n+1} = \mathcal{M}^n$  (cuando  $W^n \neq \emptyset$ ).
- $|Hin_C(\Sigma)|$  es exponencial en  $|\Sigma|$ , luego podemos obtener  $\mathcal{M}_C$  en  $O(2^{|\Sigma|})$  pasos.

# Satisfacibilidad de $K + A$ está en EXPTIME

Eliminación de Hintikka sets – algunos lemas

## Lema

Si  $\mathcal{M}_C$  está definido (con  $C \subseteq \mathbf{A}(\text{Cl}(\Sigma))$ ), entonces  $\forall H \in W_C$ :

- I.  $\forall \diamond\chi \in \text{Cl}(\Sigma)$ ,  $\diamond\chi \in H$  sii  $\exists H' \in W$ ,  $\chi \in H'$  y  $(H, H') \in R_C$ .
- II.  $\forall \mathbf{E}\chi \in \text{Cl}(\Sigma)$ ,  $\mathbf{E}\chi \in H$  sii  $\exists H' \in W$ ,  $\chi \in H'$ .

## Demostración

$\Rightarrow$ ) Si no valiera,  $H$  habría sido eliminado.

- $\Leftarrow$ )
- I.  $\mathcal{M}_C$  es un refinamiento de  $\mathcal{M}_0 \Rightarrow (H, H') \in R_C^0 \Rightarrow \diamond\chi \in H$ .
  - II.  $\chi \in H' \Rightarrow \mathbf{E}\chi \in H' \Rightarrow \mathbf{A}\neg\chi \notin H' \Rightarrow \mathbf{A}\neg\chi \notin H \Rightarrow \mathbf{E}\chi \in H$ .

# Satisfacibilidad de $K + A$ está en EXPTIME

Eliminación de Hintikka sets – algunos lemas

## Lema (Truth lemma)

Si  $\mathcal{M}_C$  está definido (con  $C \subseteq \mathbf{A}(\text{Cl}(\Sigma))$ ), entonces vale:

$$\mathcal{M}_C, H \models \varphi \Leftrightarrow \varphi \in H$$

para todo  $H \in W_C$  y todo  $\varphi \in \text{Cl}(\Sigma)$ .

## Demostración

- Sale fácil por inducción en  $\varphi$ , usando el lema anterior.



# Satisfacibilidad de $K + A$ está en EXPTIME

Eliminación de Hintikka sets – ¿para qué?

## Teorema

$\Sigma$  es satisfacible sii existen  $C \subseteq \mathbf{A}(\text{Cl}(\Sigma))$  y  $H$  en el dominio de  $\mathcal{M}_C$  tal que  $\Sigma \subseteq H$ .

## Demostración

$\Leftarrow$ ) Consecuencia directa del Truth Lemma.

$\Rightarrow$ ) Idea:

- Dado  $\mathcal{M}, w \models \Sigma$ , definir  $H_v = \{\varphi \mid \mathcal{M}, v \models \varphi \text{ y } \varphi \in \text{Cl}(\Sigma)\}$  y armar  $\mathcal{M}' = \langle W', R', V' \rangle$  tal que:

$$\begin{aligned}W' &= \{H_v \mid v \in W\} \\R' &= \{(H_v, H_{v'}) \mid (v, v') \in R\} \\V'(p) &= \{H_v \mid p \in H_v\}\end{aligned}$$

- Ver que i)  $\mathcal{M}', H_w \models \Sigma$  y ii)  $\exists C \subseteq \mathbf{A}(\text{Cl}(\Sigma)) \forall v, H_v \in \text{Hin}_C(\Sigma)$ .
- Observar que todo  $H_v$  está en  $\mathcal{M}_C$  (suponer que hay un mínimo que fue eliminado y llegar a un absurdo)

# Satisfacibilidad de $K + A$ está en EXPTIME

Un algoritmo determinístico basado en eliminación de Hintikka sets

EsSat ( $\Sigma$ )

para cada  $C \subseteq A(Cl(\Sigma))$

    calcular  $\mathcal{M}_C$  y si está definido

        para cada  $H$  en el dominio de  $\mathcal{M}_C$

            si  $\Sigma \subseteq H$

                devolver 1

devolver 0

## Observaciones

- EsSat( $\Sigma$ ) computa  $K + A$ -satisfacibilidad de  $\Sigma$  (finito).
- $|A(Cl(\Sigma))| \in O(2^{|\Sigma|})$ .
- Computar  $\mathcal{M}_C$  y recorrer su dominio lleva  $O(2^{|\Sigma|})$  pasos.
- Luego, el algoritmo requiere  $O(2^{|\Sigma|})$  pasos.

¿Será además EXPTIME-completo?