

Lógica Computacional y Demostración  
Automática

Carlos Areces  
areces@loria.fr  
<http://www.loria.fr/~areces>

INRIA Nancy Grand Est, France

Diciembre 2008

Dicembre 2008

# Lo que hacemos hoy

- ▶ Describiendo estructuras relacionales
- ▶ Lógicas modales
- ▶ Sintaxis y semántica
- ▶ Lógicas híbridas
- ▶ Model Checking
- ▶ Aplicaciones
- ▶ El model checker `mcheck`

---

· Lógica Computacional y Demostración Automática

INRIA Nancy Grand Est

- : Lógica Computacional y Demostración Automática INRIA Nancy Grand Est

## Estructuras Simples / Lenguajes Simples

Pensemos en un **grafo coloreado**:

Diagram illustrating a colored graph structure with nodes and edges, labeled with logical expressions involving colors (green, pink, blue, orange) and their negations.

Nodes and their associated labels:

- Top-left node (pink):  $\neg S^{orange}$
- Top-right node (pink):  $\neg S^{green}$ ,  $\neg S^{orange}$
- Bottom-left node (blue):  $S^{green} \wedge S^{pink}$ ,  $S^{green} \wedge S^{pink}$
- Bottom-right node (green):  $S^{blue}$ ,  $S^{blue}$
- Bottom-center node (pink):  $S^{blue}$

Edges and associated labels:

- Edge from Top-left to Top-right:  $\neg S^{orange}$
- Edge from Top-right to Bottom-right:  $S^{blue}$
- Edge from Bottom-right to Bottom-center:  $S^{blue}$
- Edge from Bottom-center to Bottom-left:  $S^{blue}$
- Edge from Bottom-left to Top-left:  $S^{green} \wedge S^{pink}$
- Edge from Bottom-left to Top-right:  $S^{green} \wedge S^{pink}$
- Edge from Bottom-left to Bottom-right:  $S^{green} \wedge S^{pink}$

Query:  $S^{green} \wedge pink \rightarrow S^{green} \wedge S^{pink} ?$

Diagram illustrating a network of states and transitions:

- Top-left node (pink):  $\neg S_{orange}$
- Top-right node (pink with blue dot):  $\neg S_{green}$ ,  $\neg S_{blue}$ ,  $\neg S_{orange}$
- Bottom-left node (pink with blue dot):  $S_{blue}$
- Central node (blue with green, pink, and blue dots):  $S_{green} \wedge S_{pink}$ ,  $S_{(green \wedge pink)}$
- Right node (green with pink dot):  $S_{blue}$

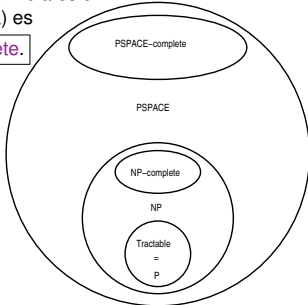
Transitions and logical relationships:

- Arrows point from the central node to the top-left, top-right, and bottom-left nodes.
- Arrows point from the top-right and bottom-left nodes to the right node.
- A self-loop arrow is on the right node.
- Logical expression below the right node:  $S_{(green \wedge pink)} \rightarrow S_{green} \wedge S_{pink} ?$

INRIA Nancy Grand Est

## Estructuras Simples / Lenguajes Simples

- ▶ Aun para este lenguaje (aparentemente) **muy simple**, decidir si una fórmula es un teorema (i.e., siempre cierta) es **PSPACE-completo**.



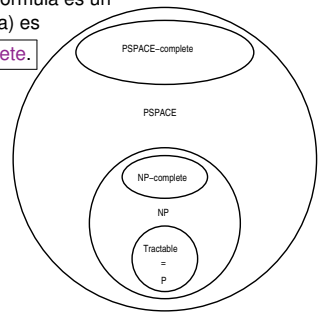
```
graph TD; PSPACE_complete([PSPACE-complete]) --- PSPACE([PSPACE]); PSPACE --- NP([NP]); NP --- Tractable([Tractable = P]);
```

Diagram illustrating complexity classes:

- PSPACE-complete (outermost set)
- PSPACE (containing PSPACE-complete)
- NP (containing PSPACE)
- Tractable (= P) (containing NP)

Footer: Lógica Computacional y Demostración Automática | INRIA Nancy Grand Est

- PSPACE-complete.



INRIA Nancy Grand Est

## Una Familia de Lenguajes

- El lenguaje que describimos puede ser **inadecuado**:
- En **área de lógica modal**, investiga el **rango** de posibles lenguajes que pueden usarse para describir estructuras relacionales.
- Las preguntas más importantes son:
  - Podemos definir algoritmos de inferencia para estos lenguajes?
  - Cuan eficientes son?
  - Cuáles son los límites de expresividad de estos lenguajes?
- La tarea **no** es fácil, porque los distintos operadores del lenguaje pueden **interactuar** de formas inesperadas. E.g., agregar el operador **S** transforma el problema de satisfiabilidad del lenguaje en EXPTIME-complete!

The diagram illustrates the complexity classes of various languages. It features four overlapping ellipses: a large outer one labeled 'EXPTIME', a middle one labeled 'PSPACE-complete', a smaller one labeled 'NP-complete', and a small inner circle labeled 'Tractable'. Arrows point from the 'Tractable' circle to 'NP-complete', from 'NP-complete' to 'PSPACE-complete', and from 'PSPACE-complete' to 'EXPTIME'. Inside the 'EXPTIME' ellipse, there are several colored dots (green, purple, blue) representing different languages. Some dots are also inside the 'PSPACE-complete' ellipse, and one is inside the 'NP-complete' ellipse. The text 'Lenguajes' is written near the dots.

- : Lógica Computacional y Demostración Automática

## Posibles Aplicaciones

- ▶ Los lenguajes de la familia de las lógicas modales pueden usarse en **áreas muy diversas**:
  - ▶ Verificación de Software y Hardware.
  - ▶ Representación de Conocimientos.
  - ▶ Criptografía.
  - ▶ Inteligencia Artificial.
  - ▶ Filosofía.
  - ▶ Lingüística Computacional.
  - ▶ Epistemología.
  - ▶ ...
- ▶ **Porque?** Muchas cosas pueden ser representadas como grafos (i.e., estructuras relacionales). Y como vimos, los lenguajes modales fueron **desarrollados especialmente** para razonar sobre grafos y describir sus propiedades.

· Lógica Computacional y Demostración Automática

INRIA Nancy Grand Est

- : Lógica Computacional y Demostración Automática INRIA Nancy Grand Est

## Lógicas Híbridas

- ▶ Vamos a presentar un tipo particular de lenguajes modales llamados **lenguajes híbridos**.
- ▶ Intuitivamente, los lenguajes híbridos son lenguajes modales que incluyen alguna **noción de igualdad**.

The diagram illustrates a hybrid modal logic formula and its semantics. A circle labeled  $S (pink \wedge X)$  has two arrows pointing to a vertical rectangle. The rectangle is divided into two horizontal sections: the top section is labeled  $X$  and contains a pink dot; the bottom section is labeled  $X$  and contains a blue dot. Below the rectangle, the formula  $S (pink \wedge X) \wedge S (blue \wedge X) \rightarrow S (pink \wedge blue)$  is written.

: Lógica Computacional y Demostración Automática

INRIA Nancy Grand Est

- 
- Diagram illustrating the decomposition of a disjunctive query into two conjunctive queries:
- Left side (Query):  $S (pink \wedge X)$
- Right side (Decomposition):
- Top section:  $X$  (containing a pink circle)
  - Bottom section:  $X$  (containing a blue circle)
- Bottom text:  $S (pink \wedge X) \wedge S (blue \wedge X) \Rightarrow S (pink \wedge blue)$

INRIA Nancy Grand Est

## El Sabor de un Clásico...

- ▶ Todo muy moderno, muy superado, pero yo prefiero la lógica clásica.
- ▶ Aunque no parezca, **estamos haciendo Lógica Clásica!!!**
- ▶ Los lenguajes que estuvimos discutiendo son **fragmentos** del lenguaje de primer (o segundo) orden. Lo único que hicimos fue elegir 'el pedacito' que necesitábamos para una aplicación dada.
- ▶ Esta es exactamente la forma en que vemos hoy por hoy a los lenguajes modales, como una forma de investigar fragmentos particularmente interesantes de los lenguajes clásicos.
- ▶ Donde **"interesantes"** significa
  - ▶ Decidibles, expresivos, de "baja" complejidad, modulares,
  - ...

· : Lógica Computacional y Demostración Automática

INRIA Nancy Grand Est

- : Lógica Computacional y Demostración Automática INRIA Nancy Grand Est

## Sintaxis

- El lenguaje híbrido que vamos a usar se define sobre la signatura  $S = (\text{PROP}, \text{REL}, \text{NOM})$  donde
  - $\text{PROP} = \{p, q, r, \dots\}$  es el conjunto de **simbolos de proposición**
  - $\text{REL} = \{r_1, r_2, r_3, \dots\}$  es el conjunto de **simbolos de relacion**
  - $\text{NOM} = \{i, j, k, \dots\}$  es el conjunto de **simbolos de constantes** (los llamamos **nominales**)
  - $\text{VAR} = \{x, y, z, \dots\}$  es el conjunto de **variables**
- Las formulas se definen entonces como

$$\varphi := a \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \langle r \rangle\varphi \mid \langle r \rangle^-\varphi \mid E\varphi \mid @_i\varphi \mid \downarrow x.\varphi$$

donde  $a \in \text{NOM} \cup \text{PROP} \cup \text{VAR}$ ,  $r \in \text{REL}$ ,  $i \in \text{NOM}$  y  $x \in \text{VAR}$ . (La formula  $[r]\varphi$  se define como  $\neg\langle r \rangle\neg\varphi$  y  $A\varphi$  se define como  $\neg E\neg\varphi$ )

## Semantica

- Un modelo  $\mathcal{M}$  es de la forma  $\langle M, I \rangle$  donde  $M$  es no vacío, y
  - $I(r)$  es una relacion binaria sobre  $M$  para  $r \in \text{REL}$ ,
  - $I(p)$  es un subconjunto de  $M$  para  $p \in \text{PROP}$ ,
  - $I(i)$  es un elemento de  $M$  para  $i \in \text{NOM}$
- Una asignación  $g$  para  $\mathcal{M}$  es una funcion  $g : \text{VAR} \rightarrow M$ .

- Definimos

$\mathcal{M}, g, m \models i$	sii	$m = I(i), i \in \text{NOM}$
$\mathcal{M}, g, m \models p$	sii	$m \in I(p), p \in \text{PROP}$
$\mathcal{M}, g, m \models \neg\varphi$	sii	$\mathcal{M}, g, m \not\models \varphi$
$\mathcal{M}, g, m \models \varphi \wedge \psi$	sii	$\mathcal{M}, g, m \models \varphi$ y $\mathcal{M}, g, m \models \psi$
$\mathcal{M}, g, m \models \langle r \rangle\varphi$	sii	$\exists m' \in M \text{ t.q. } (m, m') \in I(r) \text{ \& } \mathcal{M}, g, m' \models \varphi$
$\mathcal{M}, g, m \models \langle r \rangle^-\varphi$	sii	$\exists m' \in M \text{ t.q. } (m', m) \in I(r) \text{ \& } \mathcal{M}, g, m' \models \varphi$
$\mathcal{M}, g, m \models E\varphi$	sii	$\exists m' \in M \text{ t.q. } \mathcal{M}, g, m' \models \varphi$
$\mathcal{M}, g, m \models @_i\varphi$	sii	$\mathcal{M}, g, I(i) \models \varphi$
$\mathcal{M}, g, m \models \downarrow x.\varphi$	sii	$\mathcal{M}, g[x \leftarrow m], m \models \varphi$

## Que podemos escribir?

- Cuán expresivo es el lenguaje  $\mathcal{L}(\text{NOM}, @, \langle r \rangle, \langle r \rangle^-, E, \downarrow)$
- Empecemos con menos (y para los que saben...)
- Cual es el poder expresivo de  $\mathcal{L}(\langle r \rangle, \langle r \rangle^-)$ 
  - Es el lenguaje **temporal** básico (futuro y pasado)
  - SAT es decidable (PSPACE-complete).
  - El fragmento **guarded** de LPO<sup>2</sup>.
- Cual es el poder expresivo de  $\mathcal{L}(\text{NOM}, \langle r \rangle, \langle r \rangle^-)$ 
  - Es el lenguaje **temporal híbrido** básico
  - SAT es decidable (EXPTIME-complete).
  - El fragmento **guarded** de LPO<sup>2</sup> con constantes.
- Cual es el poder expresivo de  $\mathcal{L}(\langle r \rangle, \downarrow, E)$ 
  - Es tan expresivo como LPO!!!
  - SAT es indecible.
- Cual es el poder expresivo de  $\mathcal{L}(\langle r \rangle, \downarrow)$ 
  - Es menos expresivo que LPO.
  - SAT es todavia indecible.

## Model Checking

- El problema de **model checking global** es el siguiente:
  - Dado un modelog  $\mathcal{M}$
  - y una formula  $\alpha$ ,
  - retornar todos los estados de  $\mathcal{M}$  en los que  $\alpha$  es verdadero.
- Definimos el **truth set** de una fórmula  $\alpha$  respecto de un modelo  $\mathcal{M} = \langle W, R, V \rangle$  como:
 
$$\text{Truth}(\mathcal{M}, \alpha) = \{w \in W \mid \mathcal{M}, w \models \alpha\}$$
- Un **model checker** es un programa que dado  $\mathcal{M}$  y  $\alpha$  retorna  $\text{Truth}(\mathcal{M}, \alpha)$ .

## El Model Checker MCLite

- MCLite es un model checker para el lenguaje  $\mathcal{L}(\text{NOM}, \langle r \rangle, \langle r \rangle^-, @, E)$ .
- Por simplicidad, trabajaremos con lógicas monomodales (sólo  $\langle r \rangle$ ) pero la extensión a lógicas multimodales es simple.
- El algoritmo usa una estrategia bottom-up: examina las subfórmulas de la fórmula input  $\alpha$  en forma incremental, hasta que finalmente el obtenemos el truth set de  $\alpha$ .
- Si una subfórmula  $\beta$  de  $\alpha$  es verdadera en un estado  $w$ , el model checker marca  $w$  con  $\beta$ .

## Tipos de Datos y Funciones Auxiliares

- Sea  $\mathcal{M} = \langle W, R, V \rangle$  un modelo y  $\alpha$  la fórmula que queremos chequear.
- Sea  $\text{sub}(\alpha)$  el conjunto de subfórmulas de  $\alpha$ .
- El model checker mantendrá una bit table  $L$  de tamaño  $|\text{sub}(\alpha)| \times |W|$  tal que, para cada subfórmula  $\beta$  de  $\alpha$  y cada  $w \in W$ ,
 
$$L(\beta, w) = 1 \text{ if } \mathcal{M}, w \models \beta \text{ y } L(\beta, w) = 0 \text{ if } \mathcal{M}, w \not\models \beta$$
- Ademas, sea  $L(\beta) = \{w \in W \mid L(\beta, w) = 1\}$
- Al terminar la corrida tendremos que  $\text{Truth}(\mathcal{M}, \alpha) = L(\alpha)$
- Finalmente dado  $w \in W$ , sea  $R(w)$  el conjunto de  $R$ -sucesores de  $w$  y  $R^-(w)$  el conjunto de  $R$ -predecesores de  $w$ .

## MCLite( $M, \alpha$ )

```

1: for  $\beta \in \text{sub}(\alpha)$ ,  $w \in W$  do:  $L(\beta, w) \leftarrow 0$ 
2: for  $i \leftarrow 1$  to  $|\alpha|$  do:
3:   for  $\beta \in \text{sub}(\alpha)$  such that  $|\beta| = i$  do:
4:     case of  $\beta$ 
5:       •  $\beta \in \text{ATOM}$  :  $\forall w \in W$ , if  $w \in V(\beta)$  then  $L(\beta, w) \leftarrow 1$ 
6:       •  $\beta = \beta_1 \wedge \beta_2$  :  $\forall w \in W$ , if  $L(\beta_1, w) = L(\beta_2, w) = 1$  then  $L(\beta, w) \leftarrow 1$ 
7:       •  $\beta = \neg\beta_1$  :  $\forall w \in W$ , if  $L(\beta_1, w) = 0$  then  $L(\beta, w) \leftarrow 1$ 
8:       •  $\beta = \langle r \rangle\beta_1$  :  $\text{MC}_{\langle r \rangle}(M, \beta_1)$ 
9:       •  $\beta = \langle r \rangle^-\beta_1$  :  $\text{MC}_{\langle r \rangle^-}(M, \beta_1)$ 
10:      •  $\beta = E\beta_1$  :  $\text{MC}_E(M, \beta_1)$ 
11:      •  $\beta = @_i\beta_1$  :  $\text{MC}_{@}(M, i, \beta_1)$ 
12: return  $L(\alpha)$ 

```

### $\text{MC}_{\langle r \rangle}(M, \alpha)$

```

1: for  $w \in L(\alpha)$  do
2:   for  $v \in R^{-1}(w)$  do
3:      $L(\langle r \rangle\alpha, v) \leftarrow 1$ 

```

### $\text{MC}_{\langle r \rangle^-}(M, \alpha)$

```

1: for  $w \in L(\alpha)$  do
2:   for  $v \in R(w)$  do
3:      $L(\langle r \rangle^-\alpha, v) \leftarrow 1$ 

```

### $\text{MC}_E(M, \alpha)$

```

1: for  $w \in L(\alpha)$  do
2:   for  $w \in W$  do
3:      $L(E\alpha, w) \leftarrow 1$ 

```

### $\text{MC}_{@}(M, i, \alpha)$

```

1: let  $\{v\} = V(i)$ 
2: if  $L(\alpha, v) = 1$  then
3:   for  $w \in M$  do
4:      $L(@_i\alpha, w) \leftarrow 1$ 

```

## Worst-case Complexity de MCLite

- Dado  $f, g : N \rightarrow N$ , recordemos que

$$f(n) = O(g(n))$$

significa que hay una constante  $c > 0$  y un número natural  $n_0 \geq 1$  tal que

$$f(n) \leq c \times g(n)$$

para todo  $n \geq n_0$ .

- Por ejemplo,  $2n + 1 = O(n)$ .

## Contamos...

- Sea  $n = |W|$ ,  $m = |R|$ , y  $k$  el tamaño de  $\alpha$ .
- Notar que  $|sub(\alpha)| = O(k)$ . Además, chequear y cambiar  $L(\beta, w)$  es  $O(1)$ . Igual que  $w \in V(p)$  si  $V$  es un bit table.
- Entonces, inicializar  $L$  toma  $O(k \times n)$ . El loop principal itera  $O(k)$  veces. La complejidad de cada iteración depende de  $\beta$ .
- In particular,
  - if  $\beta \in ATOM$ , then the check of  $\beta$  takes  $O(n)$ ;
  - if  $\beta = \beta_1 \wedge \beta_2$ , then the check of  $\beta$  takes  $O(n)$ ;
  - if  $\beta = \neg\beta_1$ , then the check of  $\beta$  takes  $O(n)$ ;
  - if  $\beta = E\beta_1$ , then the check of  $\beta$  takes  $O(2n) = O(n)$ ;
  - if  $\beta = @_i\beta_1$ , then the check of  $\beta$  takes  $O(2n) = O(n)$ ;
  - if  $\beta = \langle r \rangle \beta_1$ , then the check of  $\beta$  takes  $O(n + m)$ ;
  - if  $\beta = \langle r \rangle^- \beta_1$ , then the check of  $\beta$  takes  $O(n + m)$ ;
- Es decir, la complejidad de MCLite es:  $O(k \times (n + m))$ .

## El problema con $\downarrow$

- Cuando agregamos  $\downarrow$  no podemos usar una estrategia bottom-up.
- Por que? Consideremos las formulas  $\langle r \rangle \alpha$  y  $\downarrow x. \langle r \rangle \beta(x)$ , donde  $\beta(x)$  es una formula donde aparece la variable  $x$ .
  - En el primer caso, podemos chequear  $\alpha$ , etiquetar el modelo, y luego chequear  $\langle r \rangle \alpha$  como hicimos en MCLite.
  - En el segundo caso, no podemos chequear primero  $\beta(x)$ , porque no sabemos a qué elemento esta linkeado  $x$ .

## MCFull( $M, g, \alpha$ )

```

1: for  $\beta \in sub(\alpha)$ ,  $w \in W$  do:  $L(\beta, w) \leftarrow 0$ 
2: case of  $\alpha$ :
3:   •  $\alpha \in ATOM$  :  $\forall w \in W$ , if  $w \in V(\alpha)$  then  $L(\alpha, w) \leftarrow 1$ 
4:   •  $\alpha \in VAR$  :  $L(\alpha, g(\alpha)) \leftarrow 1$ 
5:   •  $\alpha = \alpha_1 \wedge \alpha_2$  : MCFull( $M, g, \alpha_1$ ); MCFull( $M, g, \alpha_2$ )
6:   •  $\alpha = \neg\alpha_1$  : MCFull( $M, g, \alpha_1$ )
7:   •  $\alpha = @_i\alpha_1$  : MCFull( $M, g, \alpha_1$ )
8:   •  $\alpha = \langle r \rangle \alpha_1$  : MCFull( $M, g, \alpha_1$ ); MC $\langle r \rangle$ ( $M, \alpha_1$ )
9:   •  $\alpha = \langle r \rangle^- \alpha_1$  : MCFull( $M, g, \alpha_1$ ); MC $\langle r \rangle^-$ ( $M, \alpha_1$ )
10:  •  $\alpha = E\alpha_1$  : MCFull( $M, g, \alpha_1$ ); MC $E$ ( $M, t, g, \alpha_1$ ) //almost
11:  •  $\alpha = \downarrow x. \alpha_1$  : Check $\downarrow$ ( $M, g, x, \alpha_1$ )
12:  •  $\alpha = \downarrow x. \neg \alpha_1$  : Check $\downarrow$ ( $M, g, x, \alpha_1$ )
13:  •  $\alpha = \downarrow x. @_i \alpha_1$  : Check $\downarrow$ ( $M, g, x, \alpha_1$ )
14:  •  $\alpha = \downarrow x. \langle r \rangle \alpha_1$  : Check $\downarrow$ ( $M, g, x, \alpha_1$ )
15:  •  $\alpha = \downarrow x. \langle r \rangle^- \alpha_1$  : Check $\downarrow$ ( $M, g, x, \alpha_1$ )
16: return  $L(\alpha)$ 
    
```

## Check <sub>$\downarrow$</sub> ( $M, g, x, \alpha$ )

```

1: for  $w \in W$  do:
2:    $g(x) \leftarrow w$ 
3:   MCFull( $M, g, \alpha$ )
4:   if  $L(\alpha, w) = 1$  then
5:     Clear( $L, x$ )
6:    $L(\downarrow x. \alpha, w) \leftarrow 1$ 
7: else
8:   Clear( $L, x$ )
    
```

La función Clear( $L, x$ ) pone a 0 el valor de  $L(\alpha, w)$  para todo  $w \in W$  y toda fórmula  $\alpha$  con  $x$  libre.

## Complejidad de MCFull

- Supongamos que  $\alpha$  no contiene el operador  $\downarrow$ . Sea  $\alpha = \tau\alpha_1$ , donde  $\tau$  es el operador principal de  $\alpha$ .
- Sea  $C(\alpha)$  el costo de MCFull en  $\alpha$  y sea  $C_\tau(\alpha)$  el costo de  $MC_\tau$  en  $\alpha$ . Entonces,

$$C(\tau\alpha_1) = C(\alpha_1) + C_\tau(\alpha)$$

- Por lo que, en el peor caso, el costo de MCFull en  $\alpha$  es  $O(k \times (n + m))$  como para MCLite.
- Por ejemplo, si  $\alpha = \langle r \rangle \langle r \rangle \langle r \rangle p$ , entonces

$$\begin{aligned}
 C(\alpha) &= C(\langle r \rangle \langle r \rangle \langle r \rangle p) + (n + m) = \\
 &= C(\langle r \rangle \langle r \rangle p) + 2(n + m) = \\
 &= C(p) + 3(n + m) = n + 3(n + m)
 \end{aligned}$$

## Complejidad de MCFull

- Supongamos ahora que  $\alpha$  contiene  $\downarrow$ .
- Sea  $d$  el grado de anidamiento de  $\downarrow$  en  $\alpha$ . Por ejemplo  $\downarrow x. \langle r \rangle x$  tiene grado 1, y  $\downarrow x. \langle r \rangle \downarrow y. @_x y$  tiene grado 2.
- La función Check <sub>$\downarrow$</sub> ( $M, g, x, \beta$ ) corre en  $n \times C(\beta)$ .
- Por lo que, en el peor caso, el costo de MCFull en  $\alpha$  es  $O(k \times (n + m) \times n^d)$ .
- Por ejemplo, si  $\alpha = \downarrow x. \langle r \rangle \downarrow y. @_x y$ , entonces

$$\begin{aligned}
 C(\alpha) &= n \times C(\langle r \rangle \downarrow y. @_x y) = \\
 &= n \times (n + m + C(\downarrow y. @_x y)) = \\
 &= n \times (n + m + n \times C(@_x y)) = \\
 &= n \times (n + m + n \times (n + C(y))) = \\
 &= n \times (n + m + n \times (n + 1)) = O(n^3)
 \end{aligned}$$

## Complejidad de MCFull

- Es decir, la complejidad temporal de MCFull es exponencial en el grado de anidamiento de  $\downarrow$  que, en general, es proporcional al tamaño de  $\alpha$ .
- Como el tamaño del stack de recursion de MCFull está limitado por el tamaño de  $\alpha$ , la complejidad espacial de MCFull es polynomial.
- Es decir, model checking para  $\mathcal{L}(\text{NOM}, \langle r \rangle, \langle r \rangle^-, @, E, \downarrow)$  esté en PSPACE.
- Nos podemos preguntar: es este el mejor algoritmo?

## Lower bounds

- ▶ Sabemos que  $\mathcal{L}(\text{NOM}, \langle r \rangle, \langle r \rangle^-, @, E, \downarrow)$  tiene la misma expresividad que LPO.
- ▶ Es conocido que el problema de model checking para LPO es PSPACE-complete, por lo tanto también para  $\mathcal{L}(\text{NOM}, \langle r \rangle, \langle r \rangle^-, @, E, \downarrow)$ . Pero podemos demostrar un resultado mas fuerte.
- ▶ Llamemos, **fragmento no decorado de  $L$**  al fragmento de  $L$  que no usa ni símbolos de proposición ni nominales  
**Theorem:** El problema de model checking para el fragmento no decorado de  $\mathcal{L}(\downarrow)$  es PSPACE-complete.

## El model checker `mcheck`

- ▶ Es un model checker **de juguete**.
- ▶ Warning: no intentar usarlo para nada **serio**.
- ▶ Si estan interesados en model checkers para lógicas híbridas hay otras alternativas pero el input format de `mcheck` es mas simple.