Tema 1: Introducción

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

¿Qué es el razonamiento automático?

- **L. Wos: *What is Automated Reasoning?* (Journal of Automated Reasoning, Vol. 1 (1985) pp. 6–9)**
  - **Automated reasoning is concerned with the study of using the computer to assist in that part of problem solving requiring reasoning. Some questions arisinq during the study concern the representation of knowledge, the rules for deriving new knowledge frow that which is given, and strategies for controling the rules. Other questions concern the implementation of the resulting theory and concern various applications for which the corresponding software can be used. Theory, implementation, and application play equally vital and interconnecting roles for automated reasoning in its attempt to reach one of its primary goals – the goal of providing an automated reasoning assistant.**

¿Qué es el razonamiento automático?

- **Campos de aplicación del R.A.**
  - **Verificación de sistemas computacionales**
  - **Programación lógica**
  - **Programación lógica inductiva**
  - **Síntesis de programas**
  - **Inteligencia artificial**
  - **Representación del conocimiento**
  - **Sistemas expertos**
  - **Planificación**
  - **Robótica**
  - **Razonamiento de los agentes inteligentes**
  - **Procesamiento del lenguaje natural**
  - **Lógica computacional**
  - **Demostración y descubrimiento de teoremas matemáticos**

Procesamiento del conocimiento

- **Ejemplo de conocimiento:**
  1. ***El bloque rojo está sobre el bloque verde***
  2. ***El bloque verde está encima (aunque no necesariamente sobre) el bloque azul***
  3. ***El bloque verde no está sobre el bloque azul***
  4. ***El bloque amarillo está sobre el bloque verde o sobre el bloque azul***
  5. ***El bloque azul está encima de otro bloque***
  6. ***El bloque negro está sobre la mesa***
  7. ***El boque rojo está libre***

- **Tipos de conocimiento:**
  - **Conocimiento completo: 1**
  - **Restricciones: 2**
  - **Conocimiento negativo: 3**
  - **Conocimiento indefinido: 4**
  - **Conocimiento existencial: 5**
- **Problema: *X está sobre Y***

Procesamiento del conocimiento

- **Consecuencias:**
  - ***El bloque rojo está sobre el bloque verde***
  - ***El bloque verde está sobre el bloque amarillo***
  - ***El bloque amarillo está sobre el bloque azul***
  - ***El bloque azul está sobre el bloque negro***
  - ***El bloque negro está sobre la mesa***
- **Demostración**

```
* 1 (rojo sobre verde)
  2 (verde encima azul)
  3 NO (verde sobre azul)
  4 (amarillo sobre verde) O (amarillo sobre azul)
  5 (azul sobre rojo) O (azul sobre verde) O
    (azul sobre amarillo) O (azul sobre negro)
* 6 (negro sobre mesa)
  7 (libre rojo)
* 8 (amarillo sobre azul)                          [1,4]
  9 (verde sobre rojo) O (verde sobre amarillo)
    O (verde sobre negro) O (verde sobre mesa)     [3]
 10 (verde sobre amarillo) O (verde sobre negro)
    O (verde sobre mesa)                           [9,1]
 11 (verde sobre amarillo) O (verde sobre negro)   [10,2]
*12 (verde sobre amarillo)                         [11,6,2]
*13 (azul sobre negro)                             [5,7,1,12]
```

Patrones de razonamiento

- **Ejemplo 1**
  - **Todos los españoles son europeos**
  - **Todos los andaluces son españoles**
  - **Por tanto, todos los andaluces son europeos**
- **Ejemplo 2**
  - **Todos los hombres son mortales**
  - **Todos los griegos son hombres**
  - **Por tanto, todos los griegos son mortales**
- **Patrón**
  - **Todos los Y son Z**
  - **Todos los X son Y**
  - **Por tanto, todos los X son Z**
- **Notas**
  - **Forma y contenido**
  - **Silogismos de Aristóteles (siglo IV a.n.e.)**
  - **Premisas y conclusión**
  - **Patrones correctos e incorrectos**

Patrones incorrectos

- **Patrón incorrecto**
  - **Todos los X son Y**
  - **Algunos Y son Z**
  - **Por tanto, algunos X son Z**
- **Ejemplo 1 (incorrecto)**
  - **Todos los andaluces son españoles**
  - **Algunos españoles son vascos**
  - **Por tanto, algunos andaluces son vascos**
- **Ejemplo 2 (correcto)**
  - **Todos los andaluces son españoles**
  - **Algunos españoles son madridistas**
  - **Por tanto, algunos andaluces son madridistas**
- **Problemas**
  - **Decidir si un patrón dado es correcto**
  - **Calcular cuántos patrones correctos existen**
  - **Determinar los patrones correctos**

Razonamiento con OTTER

- Base de conocimiento
  - Base de reglas:
    * **R1: Si el animal tiene pelos es mamífero.**
    * **R2: Si el animal da leche es mamífero.**
    * **R3: Si el animal es un mamífero y tiene pezuñas es ungulado.**
    * **R4: Si el animal es un mamífero y rumia es ungulado.**
    * **R5: Si el animal es un ungulado y tiene cuello largo es una jirafa.**
    * **R6: Si el animal es un ungulado y tiene rayas negras es una cebra.**

  - Base de hechos:
    * **H1: El animal tiene pelos.**
    * **H2: El animal tiene pezuñas.**
    * **H3: El animal tiene rayas negras.**

  - Consecuencia
    * **El animal es una cebra.**

Razonamiento con OTTER

- **Solución con OTTER**
  - **Representación en OTTER** (`animales.in`)

```
formula_list(sos).
tiene_pelos | da_leche -> es_mamifero.
es_mamifero & (tiene_pezuñas | rumia) -> es_ungulado.
es_ungulado & tiene_cuello_largo -> es_jirafa.
es_ungulado & tiene_rayas_negras -> es_cebra.

tiene_pelos & tiene_pezuñas & tiene_rayas_negras.

-es_cebra.
end_of_list.

set(binary_res).
```

Razonamiento con OTTER

- **Solución con OTTER**

```
> otter <animales.in
-------> sos clausifies to:
list(sos).
1 [] -tiene_pelos | es_mamifero.
2 [] -da_leche | es_mamifero.
3 [] -es_mamifero | -tiene_pezuñas | es_ungulado.
4 [] -es_mamifero | -rumia | es_ungulado.
5 [] -es_ungulado | -tiene_cuello_largo | es_jirafa.
6 [] -es_ungulado | -tiene_rayas_negras | es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezuñas.
9 [] tiene_rayas_negras.
10 [] -es_cebra.
end_of_list.
set(binary_res).
   dependent: set(factor).
   dependent: set(unit_deletion).

======= end of input processing =======
```

Razonamiento con OTTER

```
========== start of search ==========

given clause #1: (wt=1) 7 [] tiene_pelos.

given clause #2: (wt=1) 8 [] tiene_pezuñas.

given clause #3: (wt=1) 9 [] tiene_rayas_negras.

given clause #4: (wt=1) 10 [] -es_cebra.

given clause #5: (wt=2) 1 [] -tiene_pelos | es_mamifero.
** KEPT (pick-wt=1): 11 [binary,1.1,7.1] es_mamifero.
11 back subsumes 2.
11 back subsumes 1.

given clause #6: (wt=1) 11 [binary,1.1,7.1] es_mamifero.

given clause #7: (wt=3) 3 [] -es_mamifero
                             | -tiene_pezuñas
                             | es_ungulado.
** KEPT (pick-wt=1): 12 [binary,3.1,11.1,unit_del,8]
                       es_ungulado.
12 back subsumes 4.
12 back subsumes 3.

given clause #8: (wt=1) 12 [binary,3.1,11.1,unit_del,8]
                           es_ungulado.

given clause #9: (wt=3) 6 [] -es_ungulado
                             | -tiene_rayas_negras
                             | es_cebra.
** KEPT (pick-wt=0): 13 [binary,6.1,12.1,unit_del,9,10]
                       $F.
```

Razonamiento con OTTER

```
Length of proof is 2.  Level of proof is 2.

---------------- PROOF ----------------

1 [] -tiene_pelos | es_mamifero.
3 [] -es_mamifero | -tiene_pezuñas | es_ungulado.
6 [] -es_ungulado | -tiene_rayas_negras | es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezuñas.
9 [] tiene_rayas_negras.
10 [] -es_cebra.
11 [binary,1.1,7.1] es_mamifero.
12 [binary,3.1,11.1,unit_del,8] es_ungulado.
13 [binary,6.1,12.1,unit_del,9,10] $F.

------------ end of proof -------------
```

Razonamiento no deductivo

- **Inducción**
  - **Al unir 2 puntos de una circunferencia el círculo se divide en 2 partes**
  - **Al unir 3 puntos de una circunferencia el círculo se divide en 4 partes**
  - **Al unir 4 puntos de una circunferencia el círculo se divide en 8 partes**
  - **Al unir 5 puntos de una circunferencia el círculo se divide en 16 partes**
  - **Por tanto, al unir 6 puntos de una circunferencia el círculo se divide en 32 partes**
- **Abducción**
  - **Cuando llueve, el suelo está mojado**
  - **El suelo está mojado**
  - **Por tanto, ha llovido**

PLI con Progol

- **Ejemplos positivos**

```
p([]).          p([0]).         p([0,0]).
p([1,1]).       p([0,0,0]).     p([0,1,1]).
p([1,0,1]).     p([1,1,0]).     p([0,0,0,0]).
p([0,0,1,1]).   p([0,1,0,1]).   p([1,0,0,1]).
p([0,1,1,0]).   p([1,0,1,0]).   p([1,1,0,0]).
p([1,1,1,1]).
```

- **Ejemplos negativos**

```
:- p([1]).          :- p([0,1]).
:- p([1,0]).        :- p([0,0,1]).
:- p([0,1,0]).      :- p([1,0,0]).
:- p([1,1,1]).      :- p([0,0,0,1]).
:- p([0,0,1,0]).    :- p([0,1,0,0]).
:- p([1,0,0,0]).    :- p([0,1,1,1]).
:- p([1,0,1,1]).    :- p([1,1,0,1]).
:- p([1,1,1,0]).
```

- **Modos**

```
:- modeh(1,p(+lista_binaria))?
:- modeb(1,+constant = #constant)?
:- modeb(1,+lista_binaria = [-binario|-lista_binaria])?
:- modeb(1,p(+lista_binaria))?
:- modeb(1,not(p(+lista_binaria)))?
```

- **Tipos**

```
lista_binaria([]).
lista_binaria([X|Y]) :- binario(X), lista_binaria(Y).

binario(0).
binario(1).
```

PLI Progol

- **Sesión**

```
> progol ej1
CProgol Version 4.4
...
p([]).
p([0|A]) :- p(A).
p([1|A]) :- not(p(A)).
```

Arboles de decisión con Progol

Ejemplo	Acción	Autor	Tema	Longitud	Sitio
e1	saltar	conocido	nuevo	largo	casa
e2	leer	desconocido	nuevo	corto	trabajo
e3	saltar	desconocido	viejo	largo	trabajo
e4	saltar	conocido	viejo	largo	casa
e5	leer	conocido	nuevo	corto	casa
e6	saltar	conocido	viejo	largo	trabajo
e7	saltar	desconocido	viejo	corto	trabajo
e8	leer	desconocido	nuevo	corto	trabajo
e9	saltar	conocido	viejo	largo	casa
e10	saltar	conocido	nuevo	largo	trabajo
e11	saltar	desconocido	viejo	corto	casa
e12	saltar	conocido	nuevo	largo	trabajo
e13	leer	conocido	viejo	corto	casa
e14	leer	conocido	nuevo	corto	trabajo
e15	leer	conocido	nuevo	corto	casa
e16	leer	conocido	viejo	corto	trabajo
e17	leer	conocido	nuevo	corto	casa
e18	leer	desconocido	nuevo	corto	trabajo

Arboles de decisión en Progol

- **Representación** `softbot.pl`
  - **Parámetros**

```
:- set(r,1000)?
:- set(posonly)?
```

  - **Modos**

```
:- modeh(1,accion(+ejemplo,#t_accion))?
:- modeb(1,autor(+ejemplo,#t_autor))?
:- modeb(1,tema(+ejemplo,#t_tema))?
:- modeb(1,longitud(+ejemplo,#t_longitud))?
:- modeb(1,sitio(+ejemplo,#t_sitio))?
```

  - **Tipos**

```
ejemplo(e1).  ...  ejemplo(e18).
t_accion(saltar).  t_accion(leer).
t_autor(conocido). t_autor(desconocido).
t_tema(nuevo).     t_tema(viejo).
t_longitud(largo). t_longitud(corto).
t_sitio(casa).     t_sitio(trabajo).
```

  - **Conocimiento base**

```
autor(e1,conocido).    ... autor(e18,desconocido).
tema(e1,nuevo).        ... tema(e18, nuevo).
longitud(e1,largo).    ... longitud(e18,corto).
sitio(e1,casa).        ... sitio(e18, trabajo).
```

  - **Ejemplos positivos**

```
accion(e1,saltar).     ... accion(e18,leer).
```

Arboles de decisión en Progol

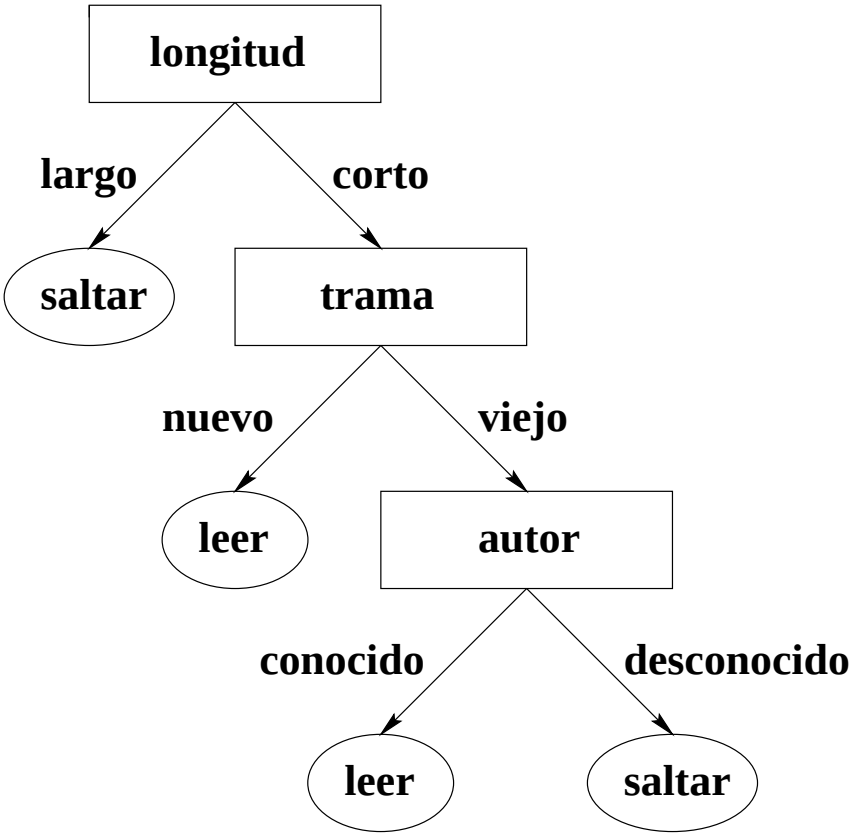
- **Restricciones**

```
:- hypothesis(Cabeza,Cuerpo,_),
   accion(A,C),
   Cuerpo,
   Cabeza = accion(A,B),
   B \= C.
```

- **Sesión**

```
> progol ej2
CProgol Version 4.4
...
accion(A,saltar) :-
  longitud(A,largo).
accion(A,leer) :-
  tema(A,nuevo),
  longitud(A,corto).
accion(A,saltar) :-
  autor(A,desconocido),
  tema(A,viejo).
accion(A,leer) :-
  autor(A,conocido),
  longitud(A,corto).
```

Arboles de decisión en Progol



Bibliografía

- Bundy, A. *The Computer Modelling of Mathematical Reasoning* (Academic Press, 1983)
  - Cap. 1 "Introduction"
- Genesereth, M.R. *Computational Logic* (27 March 2000)
  - Cap. 1: "Introduction"
- Muggleton, S. y Firth, J. *CProgol4.4: a tutorial introduction* (En "Inductive Logic Programming and Knowledge Discovery in Databases". Springer, 2000)
- Nilsson, N.J. *Inteligencia artificial (Una nueva síntesis)* (McGraw–Hill, 2000)
  - Cap. 1 "Introducción"
- Russell, S. y Norvig, P. *Inteligencia artificial (un enfoque moderno)* (Prentice Hall Hispanoamericana, 1996)
  - Cap. 6 "Agentes que razonan de manera lógica"