

Inteligencia Artificial

Agentes Lógicos



Universidad
Rey Juan Carlos

Lógica de Primer Orden

- La lógica proposicional es demasiado pequeña para representar el conocimiento de entornos complejos de una manera concisa.
- La lógica de primer orden es suficientemente expresiva para representar una buena parte de nuestro conocimiento de sentido común.

- La naturaleza de los lenguajes de representación
 - Lenguajes de Programación



1. Carecen de un mecanismo *general* de obtener hechos de otros hechos
2. Carecen de la expresividad para manejar información parcial

- La lógica proposicional es declarativo.
 1. Separación entre conocimiento e inferencia
 2. Maneja información parcial ($\wedge \vee$)
 3. Composicionalidad
- Le falta expresividad

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \quad \text{vs. "Las casillas al lado de pozos tienen brisa"}$$

- La lógica proposicional es demasiado pequeña para representar el conocimiento de entornos complejos de una manera concisa.
- La **lógica de primer orden** es suficientemente expresiva para representar una buena parte de nuestro conocimiento de sentido común.

- Los lenguajes naturales



1. Son muy expresivos
2. Son medio de comunicación, además de representación
3. Contexto y ambigüedad

- Lógica de Primer Orden combina la lógica proposicional con las partes del lenguaje natural que no tiene inconvenientes, construyéndose alrededor de objetos y relaciones

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

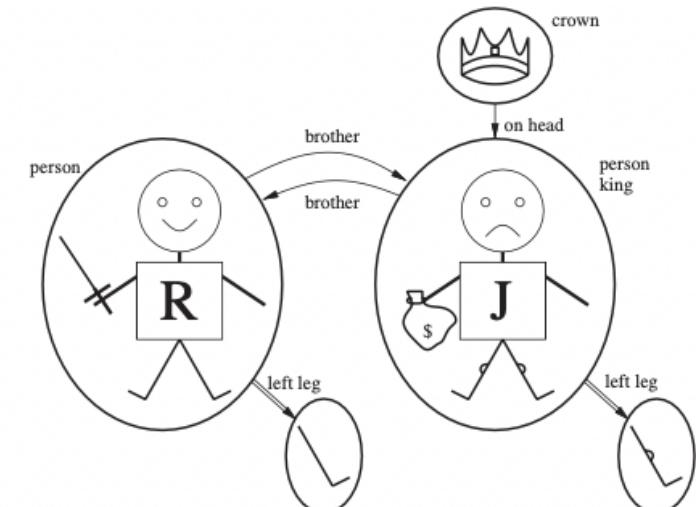
Sintaxis y Semántica de la Lógica de Primer Orden

Modelos para la Lógica de Primer Orden

- Cada modelo vincula el vocabulario de las sentencias lógicas con elementos del mundo posible, de modo que se pueda determinar la verdad de cualquier oración.
- El **dominio** de un modelo es el conjunto de **objetos** o **elementos de dominio** que contiene.
 - Se requiere que no esté vacío.
 - Lo que importa es *cuántos objetos* hay en cada modelo en particular, no cuáles son.

Modelo con 5 objetos

- Los objetos se pueden relacionar de varias maneras:
 - **Relación binaria** de hermandad. Conjunto (set) de <tupla>:
 ↪ {<Richard the Lionheart, King John>, <King John, Richard the Lionheart>}
 - **Relación binaria “crown on head”**. <tupla>:
 ↪ <the crown, King John>
 - **Relación unarias**, o propiedades:
 - person
 - king
 - **Funciones**:
 - Relación que, dado un objeto, se relaciona con un solo objeto



- Se requieren **funciones totales** (debe haber un valor para cada tupla de entrada). ¿La corona tiene pierna izquierda?

Símbolos e interpretación

- Los elementos sintácticos básicos de la lógica de primer orden son los símbolos que representan objetos, relaciones y funciones:

- **Símbolos constantes** - para los objetos
Richard, John
- **Símbolos de predicado** - para las relaciones
Brother, OnHead, Gerson, King, Crown
- **Símbolos de funciones** - para las funciones
LeftLeg
- Cada predicado o función viene con una **aridad** que fija el número de argumentos

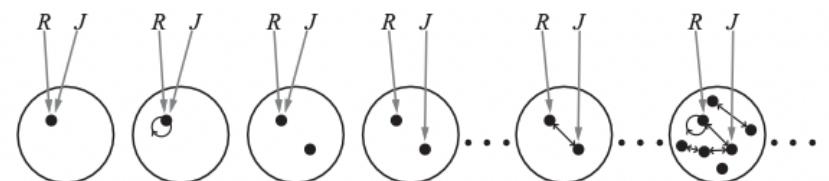
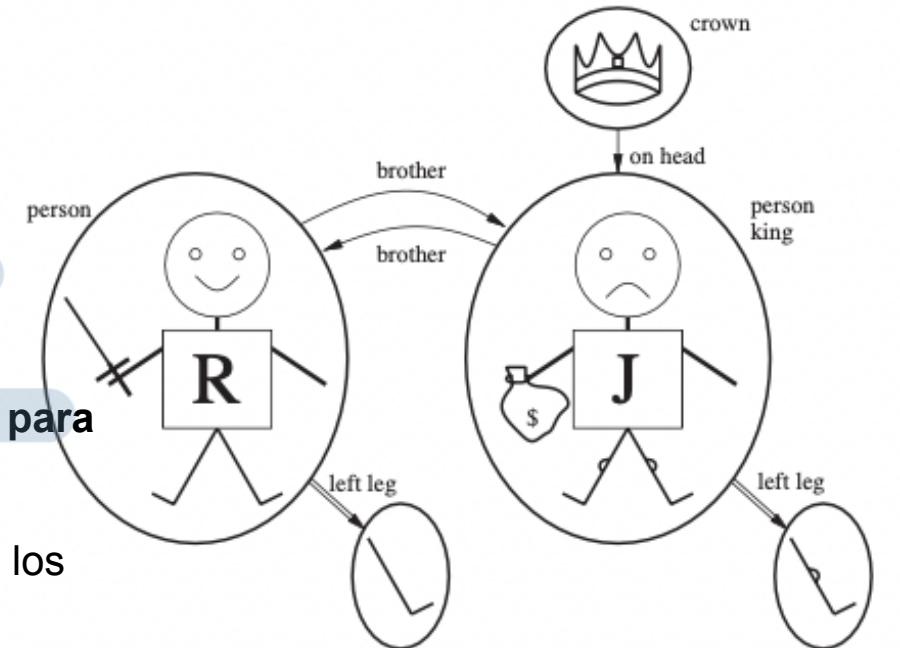
- Cada modelo debe proporcionar la información requerida para determinar si una sentencia dada es verdadera o falsa.

- Cada modelo incluye una **interpretación (I)** que especifica exactamente a qué objetos, relaciones y funciones se refieren los símbolos de constante, predicado y función.

- *Richard y John*
- *Brothers* es la relación de hermandad
- *OnHead* es la relación "on head"
- Person, King y Crown son los conjuntos de objetos de cada tipo
- *LeftLeg* es la función "left leg"

First Order Logic

- Un modelo en FOL consiste en un conjunto de objetos y una interpretación que mapea **símbolos constantes a objetos**, símbolos de predicado a relaciones en esos objetos y símbolos de función a funciones en esos objetos
- El número de posibles modelos puede ser prácticamente infinito en Lógica de Primer Orden



Símbolos e interpretación

- Un **término** es una expresión lógica que se refiere a un objeto.

- Los símbolos constantes son términos
- No todos los símbolos deben tener nombre. Por eso hay funciones
- Una función es un término complejo, no una llamada a una subrutina con un valor de retorno. λ – expressions

LeftLeg(John)

- Una **sentencia atómica expresa hechos**

- Formada por un símbolo de predicado y opcionalmente una lista de términos entre paréntesis

*Brother(Richard, John)
Married(Father(Richard), Mother(John))*

- Una oración atómica es **verdadera** en un modelo dado si la relación a la que hace referencia el símbolo de predicado se cumple entre los objetos a los que se refieren los argumentos.

- Una **sentencia compleja** usa conectores lógicos para construirse. Igual que en lógica proposicional.

\neg Brother (LeftLeg (Richard), John)
Brother (Richard, John) \wedge Brother (John, Richard)
King (Richard) \vee King (John)
 \neg King(Richard) \Rightarrow King(John)

```

Sentence → AtomicSentence | ComplexSentence
AtomicSentence → Predicate | Predicate(Term,...) | Term = Term
ComplexSentence → ( Sentence ) | [ Sentence ]
|  $\neg$  Sentence
| Sentence  $\wedge$  Sentence
| Sentence  $\vee$  Sentence
| Sentence  $\Rightarrow$  Sentence
| Sentence  $\Leftrightarrow$  Sentence
| Quantifier Variable,... Sentence

```

```

Term → Function( Term,...)
| Constant
| Variable

```

```

Quantifier →  $\forall$  |  $\exists$ 
Constant → A |  $X_1$  | John | ...
Variable → a | x | s | ...
Predicate → True | False | After | Loves | Raining | ...
Function → Mother | LeftLeg | ...

```

OPERATOR PRECEDENCE : $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$



Símbolos e interpretación

- Los **cuantificadores** permiten expresar propiedad de colecciones enteras de objetos, en lugar de enumerar los objetos por nombre. El símbolo x es llamado variable y se escribe en minúsculas.

- **Universales**

$$\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$$

x → Richard the Lionheart,
x → King John,
x → Richard's left leg,
x → John's left leg,
x → the crown.

Richard the Lionheart is a king ⇒ Richard the Lionheart is a person.

King John is a king ⇒ King John is a person.

Richard's left leg is a king ⇒ Richard's left leg is a person.

John's left leg is a king ⇒ John's left leg is a person.

The crown is a king ⇒ the crown is a person.

Símbolos e interpretación

- Los **cuantificadores** permiten expresar propiedad de colecciones enteras de objetos, en lugar de enumerar los objetos por nombre.
 - **Existenciales**

$$\exists x \text{ Crown}(x) \wedge \text{OnHead}(x, \text{John})$$

$x \rightarrow$ King John,
 $x \rightarrow$ Richard's left leg,
 $x \rightarrow$ John's left leg,
 $x \rightarrow$ the crown.

King John is a crown \wedge King John is on John's head;
Richard the Lionheart is a crown \wedge Richard the Lionheart is on John's head;
Richard's left leg is a crown \wedge Richard's left leg is on John's head;
John's left leg is a crown \wedge John's left leg is on John's head;
The crown is a crown \wedge the crown is on John's head.

Símbolos e interpretación

- Los **cuantificadores** permiten expresar propiedad de colecciones enteras de objetos, en lugar de enumerar los objetos por nombre.
 - **Cuantificadores anidados**

$\forall x \forall y \text{ Brother}(x,y) \Rightarrow \text{Sibling}(x,y)$

$\forall x, y \text{ Sibling}(x,y) \Leftrightarrow \text{Sibling}(y,x)$

En el caso de utilizar diferentes cuantificadores el orden es muy importante:

“Todo el mundo quiere a alguien”

$\forall x \exists y \text{ Loves}(x,y)$

“Hay alguien que es querido por todo el mundo”

$\exists y \forall x \text{ Loves}(x,y)$

Símbolos e interpretación

- Los **cuantificadores** permiten expresar propiedad de colecciones enteras de objetos, en lugar de enumerar los objetos por nombre.
 - **Conexión entre cuantificadores:** Conectados a través de la negación del otro.

$\forall x \neg \text{Likes}(x, \text{Parsnips})$ es equivalente a $\neg \exists x \text{ likes}(x, \text{Parsnips})$

$\forall x \text{ likes}(x, \text{IceCream})$ es equivalente a $\neg \exists x \neg \text{likes}(x, \text{IceCream})$

$$\begin{array}{ll}
 \forall x \neg P \equiv \neg \exists x P & \neg(P \vee Q) \equiv \neg P \wedge \neg Q \\
 \neg \forall x P \equiv \exists x \neg P & \neg(P \wedge Q) \equiv \neg P \vee \neg Q \\
 \forall x P \equiv \neg \exists x \neg P & P \wedge Q \equiv \neg(\neg P \vee \neg Q) \\
 \exists x P \equiv \neg \forall x \neg P & P \vee Q \equiv \neg(\neg P \wedge \neg Q)
 \end{array}$$

- **Igualdad**

- Podemos usar el símbolo de igualdad para significar que dos términos se refieren al mismo objeto
- ¿Cuál se refiere a que Richard tiene al menos dos hermanos?

$\exists x, y \text{ Brother}(x, \text{Richard}) \wedge \text{Brother}(y, \text{Richard}) \wedge \neg(x=y)$

$\exists x, y \text{ Brother}(x, \text{Richard}) \wedge \text{Brother}(y, \text{Richard})$

Símbolos e interpretación

- Los **cuantificadores** permiten expresar propiedad de colecciones enteras de objetos, en lugar de enumerar los objetos por nombre.
 - **Semántica de base de datos** - una semántica alternativa a la semántica estándar

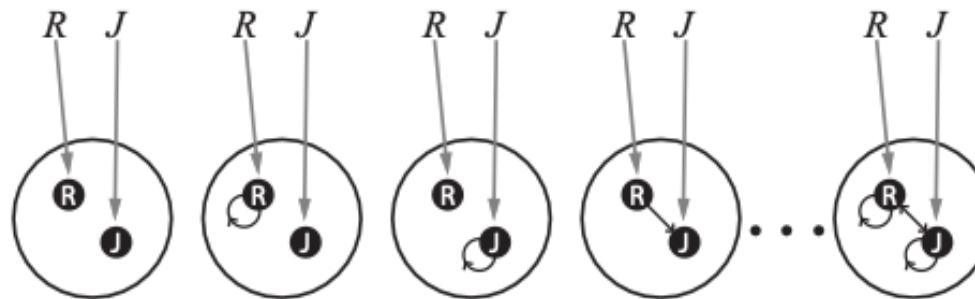
Los hermanos de Richard son John y Geoffrey

Brother (John , Richard) \wedge Brother (Geoffrey , Richard) ? Válida bajo asunción de semántica de base de datos

Brother (John , Richard) \wedge Brother (Geoffrey , Richard) \wedge John/= Geoffrey
 $\wedge \forall x \text{ Brother}(x,\text{Richard}) \Rightarrow (x=\text{John} \vee x=\text{Geoffrey})$

¿Qué se hace en las bases de datos para simplificar?

- **Suposición de nombres únicos:** cada símbolo constante se refiere a un objeto distinto
- **Suposición de mundo cerrado:** las sentencias atómicas que no se sabe que son verdaderas son falsas
- **Cierre de dominio:** cada modelo no contiene más elementos de dominio que los nombrados por los símbolos constantes



Usando Lógica de Primer Orden

Afirmaciones y consultas en lógica de primer orden

- Las sentencias se añaden a la base de conocimiento mediante **aserciones** usando TELL

$\text{TELL}(KB, \text{King}(\text{John})) .$

$\text{TELL}(KB, \text{Person}(\text{Richard})) .$

$\text{TELL}(KB, \forall x \text{ King}(x) \Rightarrow \text{Person}(x)) .$

- Preguntamos a la base de conocimiento **queries** o **goals** mediante **aserciones** usando TELL

$\text{ASK}(KB, \text{King}(\text{John}))$

$\text{ASK}(KB, \exists x \text{ Person}(x))$

$\text{ASKVARS}(KB, \text{Person}(x))$ devuelve $\{x / \text{John}\}$ y $\{x / \text{Richard}\}$

**Substitución o
lista vinculante**

El dominio del parentesco

- El primer dominio de ejemplo permite tener:
 - **hechos** como: "*Elizabeth es la madre de Charles*" y "*Charles es el padre de William*"
 - **reglas** como: "*La abuela de uno es la madre de uno de los padres*"
 - Los **objetos** son personas
 - 2 predicados unarios: *Hombre* y *Mujer*
 - Predicados binarios (relaciones): *Padre*, *Hermano*, *Hermano*, *Hermana*, *Niño*, *Hija*, *Hijo*, *Cónyuge*, *Esposa*, *Esposo*, *Abuelo*, *Nieto*, *Primo*, *Tía* y *Tío*
 - 2 funciones: *Madre* y *Padre* ← porque cada persona tiene exactamente uno de estos

$$\forall m, c \text{ } Mother(c) = m \Leftrightarrow Female(m) \wedge Parent(m, c)$$

$\forall w, h \ Husband(h, w) \Leftrightarrow Male(h) \wedge Spouse(h, w)$

$$\forall x \ Male(x) \Leftrightarrow \neg Female(x)$$

$$\forall p, c \text{ } Parent(p, c) \Leftrightarrow Child(c, p)$$

$$\forall g, c \text{ } Grandparent(g, c) \Leftrightarrow \exists p \text{ } Parent(g, p) \wedge Parent(p, c)$$

$$\forall x, y \ Sibling(x, y) \Leftrightarrow x \neq y \wedge \exists p \ Parent(p, x) \wedge Parent(p, y)$$

Axiomas

$$\forall x, y \ Sibling(x, y) \Leftrightarrow Sibling(y, x)$$

Teorema

están implicados
por los axiomas



El dominio del parentesco

- No es necesario definir todos los axiomas. *Person* no está definido

$$\forall x \ Person(x) \Leftrightarrow \dots$$

- Aunque se puede definir mediante la especificación parcial de propiedades

$$\forall x \ Person(x) \Rightarrow \dots$$

$$\forall x \ \dots \Rightarrow Person(x)$$

$$\forall m, c \ Mother(c) = m \Leftrightarrow Female(m) \wedge Parent(m, c)$$

$$\forall w, h \ Husband(h, w) \Leftrightarrow Male(h) \wedge Spouse(h, w)$$

$$\forall x \ Male(x) \Leftrightarrow \neg Female(x)$$

$$\forall p, c \ Parent(p, c) \Leftrightarrow Child(c, p)$$

$$\forall g, c \ Grandparent(g, c) \Leftrightarrow \exists p \ Parent(g, p) \wedge Parent(p, c)$$

$$\forall x, y \ Sibling(x, y) \Leftrightarrow x \neq y \wedge \exists p \ Parent(p, x) \wedge Parent(p, y)$$

Axiomas

$$\forall x, y \ Sibling(x, y) \Leftrightarrow Sibling(y, x)$$

Teorema
están implicados
por los axiomas

Ejercicio

¿El hecho $\neg Spouse(George, Laura)$ se deriva de los hechos $Jim \neq George$ y $Spouse(Jim, Laura)$? Si es así, demuéstralos; si no, proporciona axiomas adicionales según sea necesario. ¿Qué sucede si usamos *Spouse* como un símbolo de función unario en lugar de un predicado binario?

El dominio de los números naturales

- Los **objetos** son los números naturales
- 1 predicado unario: $NatNum$
- 1 funciones: S sucesor
- 1 símbolo constante: 0

$$NatNum(0) .$$

$$\forall n \ NatNum(n) \Rightarrow NatNum(S(n)) \\ 0, S(0), S(S(0))$$

$$\forall n \ 0 \neq S(n)$$

$$\forall m, n \ m \neq n \Rightarrow S(m) \neq S(n)$$

Axiomas

- Definición de suma

$$\forall m \ NatNum(m) \Rightarrow + (0, m) = m .$$

$$\forall m, n \ NatNum(m) \wedge NatNum(n) \Rightarrow + (S(m), n) = S(+ (m, n))$$

$$\forall m, n \ NatNum(m) \wedge NatNum(n) \Rightarrow (m + 1) + n = (m + n) + 1$$

El mundo de Wumpus

- Percepciones es el predicado binario *Percept*:

$$\text{Percept}([\text{Stench}, \text{Breeze}, \text{Glitter}, \text{None}, \text{None}], 5)$$

$$\forall t, s, g, m, c \text{ } \text{Percept}([s, \text{Breeze}, g, m, c], t) \Rightarrow \text{Breeze}(t)$$

$$\forall t, s, b, m, c \text{ } \text{Percept}([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{Glitter}(t)$$

- Acciones

$$\text{Turn(Right)}, \text{Turn(Left)}, \text{Forward}, \text{Shoot}, \text{Grab}, \text{Climb}$$

$$\text{ASKVARS}(\exists a \text{ } \text{BestAction}(a, 5))$$

$$\forall t \text{ } \text{Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab}, t) \quad \{a/\text{Grab}\}$$

$$\forall x, y, a, b \text{ } \text{Adjacent}([x, y], [a, b]) \Leftrightarrow$$

$$(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1))$$

- Predicado unario *Pit* que se evalúa a verdadero en las casillas con hoyos

- *Wumpus* constante o predicado unario

$$\begin{array}{ll} \text{At}(\text{Agent}, s, t) & \forall x, s_1, s_2, t \text{ } \text{At}(x, s_1, t) \wedge \text{At}(x, s_2, t) \Rightarrow s_1 = s_2 \\ \forall t \text{At}(\text{Wumpus}, [2, 2], t). & \end{array}$$

$$\forall s, t \text{ } \text{At}(\text{Agent}, s, t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$$

$$\forall s \text{ } \text{Breezy}(s) \Leftrightarrow \exists r \text{ } \text{Adjacent}(r, s) \wedge \text{Pit}(r)$$

$$\forall t \text{ } \text{HaveArrow}(t + 1) \Leftrightarrow (\text{HaveArrow}(t) \wedge \neg \text{Action}(\text{Shoot}, t))$$