Lógica de Primer Orden (FOL)

Inteligencia Artificial e Ingeniería del Conocimiento

Constantino Antonio García Martínez

Universidad San Pablo Ceu

Contenidos

Revisión de Lógica de Primer Orden (FOL)

Lógica de Primer Orden: Sintaxis

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\ \textit{Sentence}\ ) \\
&\mid \neg\, \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier Variable}, \ldots \textit{Sentence} \\
\\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{Variable} \\
\\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\\
\text{OPERATOR PRECEDENCE} &: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Revisión de Lógica de Primer Orden (FOL)

Lógica de Primer Orden: Una Base de Conocimiento

Ejemplo: Sistema de Recomendación de Películas

Motivación
¿Cómo modelaría un sistema de streaming el conocimiento sobre películas, usuarios y preferencias para hacer recomendaciones?

Queremos representar:

- Películas específicas: *Inception*, *Interstellar*, *The Matrix*
- Personas: Ana, Carlos, María
- Géneros: Ciencia Ficción, Drama, Acción
- Relaciones: "A Ana le gusta Inception", "Matrix es de ciencia ficción"
- Información estructurada: Director de una película, año de estreno

Objetivo: Modelar este dominio usando la sintaxis de Lógica de Primer Orden

Modelando el Conocimiento en Lógica de Primer Orden

Símbolos de Constante (*Constant*):

Representan objetos específicos del dominio.

- Películas: *Inception*, *Interstellar*, *Matrix*, *Parasite*, *Arrival*
- Personas: *Ana*, *Carlos*, *Maria*
- Géneros: *SciFi*, *Drama*, *Thriller*
- Directores: *Nolan*, *Wachowski*, *Villeneuve*

Exercise: Símbolos de Predicado

Escribe símbolos de predicado apropiados para representar las siguientes relaciones. Indica la aridad (número de argumentos) y el significado de cada argumento:

- "A una persona le gusta una película"
- "Una película pertenece a un género"
- "Una película fue dirigida por un director"
- "Una persona ha visto una película"
- "Una película tiene alta valoración"

Exercise: Símbolos de Función

Escribe símbolos de función apropiados para obtener información estructurada.

Indica la aridad y qué devuelve cada función:

- "Obtener el director de una película"
- "Obtener el año de estreno de una película"
- "Obtener la valoración numérica de una película"

Sentencias Atómicas y Complejas

Exercise: Sentencias Atómicas

Usando los predicados y términos definidos, escribe sentencias atómicas para representar los siguientes hechos:

- "A Ana le gusta Inception"
- "Matrix es de ciencia ficción"
- "Interstellar fue dirigida por Nolan"
- "El director de Inception es Nolan" (usando igualdad)
- "Parasite se estrenó en 2019" (usando igualdad)

Sentencias Atómicas y Complejas

Exercise: Sentencias Complejas

Escribe sentencias complejas para representar:

- "A Ana le gusta Inception e Inception es de ciencia ficción"
- "Si una película m ha sido vista por un usuario u, entonces le gusta o no le gusta" (tautología)
- "Si una película p tiene alta valoración y es de ciencia ficción, entonces le gusta a Carlos"

Exercise: Cuantificador Universal

Escribe sentencias cuantificadas universalmente para expresar:

- "Todas las películas dirigidas por Nolan tienen alta valoración"
- "Si a Ana le gusta una película, es porque la ha visto"

Exercise: Cuantificador Existencial

Escribe sentencias cuantificadas existencialmente para expresar:

- "Existe al menos una película de ciencia ficción que le gusta a Ana"
- "A María le gusta al menos una película de Villeneuve"

Base de Conocimiento Completa: Algunas Reglas

Exercise: Reglas generales
Escribe las siguientes reglas generales:

1. Si a alguien le gusta una película, la ha visto
2. Si te gusta una película de un género, existen otras del mismo género
3. Todas las películas de Nolan tienen alta valoración
4. Si te gusta una película altamente valorada de un género, hay otras peliculas de ese género altamente valoradas

Base de Conocimiento Completa: Algunos Hechos

Finalmente, después de escribir reglas en nuestra KB, la alimentamos con **Hechos específicos** (ground atomic sentences):

Example: Hechos específicos

$$\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$$

$$\mathit{DirectedBy}(\mathit{Interstellar}, \mathit{Nolan})$$

$$\mathit{DirectedBy}(\mathit{Matrix}, \mathit{Wachowski})$$

$$\mathit{HasGenre}(\mathit{Inception}, \mathit{SciFi}) \land \mathit{HasGenre}(\mathit{Inception}, \mathit{Thriller})$$

$$\mathit{HasGenre}(\mathit{Interstellar}, \mathit{SciFi}) \land \mathit{HasGenre}(\mathit{Interstellar}, \mathit{Drama})$$

$$\mathit{Likes}(\mathit{Ana}, \mathit{Inception})$$

$$\mathit{Likes}(\mathit{Ana}, \mathit{Interstellar})$$

$$\mathit{Likes}(\mathit{Carlos}, \mathit{Matrix})$$

$$\mathit{WatchedBy}(\mathit{Inception}, \mathit{Ana})$$

$$\mathit{HighlyRated}(\mathit{Inception}) \land \mathit{HighlyRated}(\mathit{Interstellar})$$

Semántica de Base de Datos

Definición: Semántica de Base de Datos
La **semántica de base de datos** (database semantics) es una convención de modelado donde asumimos que:

1. Cada hecho atómico que **no está explícitamente** en la KB es **falso**
2. Solo lo que está afirmado (o se puede inferir) es verdadero
3. Esto se conoce como la **"Closed World Assumption"** (CWA)

En nuestro ejemplo:

- Si $Likes(Maria, Matrix)$ NO está en la KB $\Rightarrow$ asumimos que es **falso**
- Si $HasGenre(Arrival, Drama)$ NO está en la KB $\Rightarrow$ asumimos que es **falso**
- Solo conocemos lo que hemos almacenado explícitamente

Esto contrasta con la **"Open World Assumption"** (OWA) donde la ausencia de información significa "desconocido", no necesariamente "falso".

Usando Lógica de Primer Orden: ASK, TELL y ASKVARS

Operaciones Básicas: TELL y ASK

Al igual que en lógica proposicional, tenemos dos operaciones fundamentales:

TELL
Añade una nueva sentencia a la base de conocimiento.

Ejemplos:

- $TELL(KB, Likes(Carlos, Arrival))$
- $TELL(KB, \forall x\ HasGenre(x, SciFi) \Rightarrow HighlyRated(x))$
- $TELL(KB, DirectedBy(Arrival, Villeneuve))$

ASK
Pregunta si una sentencia se puede inferir de la KB. Devuelve `True` o `False`.

Ejemplos:

- $ASK(KB, Likes(Ana, Inception)) \rightarrow$ `True`
- $ASK(KB, HighlyRated(Inception)) \rightarrow$ `True`
- $ASK(KB, Likes(Maria, Matrix)) \rightarrow$ `False` (con CWA)

ASKVARS: Consultas con Variables

ASKVARS
En lógica de primer orden, a menudo queremos saber **qué objetos** satisfacen una consulta, no solo si es verdadera o falsa.

$ASKVARS(KB, \text{consulta con variables})$ devuelve una lista de **sustituciones** (bindings) que hacen verdadera la consulta.

Example: ¿Qué películas le gustan a Ana?

$$ASKVARS(KB, Likes(Ana, x))$$

Respuesta: lista de sustituciones

$$\{x/Inception\}, \{x/Interstellar\}$$

Esto significa: $x = Inception$ y $x = Interstellar$ satisfacen la consulta.

Substitución o Binding
Una **sustitución** o **binding** es un mapeo de variables a términos. **Notación**: $\{variable/term\}$

Listas de Sustitución (Binding Lists)

Example: ¿Qué películas dirigidas por Nolan están en la KB?

$$ASKVARS(KB, DirectedBy(m, Nolan))$$

Respuesta:

$$\{m/Inception\}, \{m/Interstellar\}$$

Example: ¿Qué persona-película cumplen que a la persona le gusta la película y es de ciencia ficción?

$$ASKVARS(KB, Likes(u, m) \land HasGenre(m, SciFi))$$

Respuesta (conjunto de sustituciones):

$$\{u/Ana, m/Inception\}$$

$$\{u/Ana, m/Interstellar\}$$

$$\{u/Carlos, m/Matrix\}$$

Cada línea es una **sustitución completa** que satisface la consulta.

Aplicación: Sistema de Recomendación

¿Cómo usar esto para recomendar películas?

Exercise: Recomienda películas de ciencia ficción que Carlos no ha visto

Exercise: ¿Qué películas le gustan a usuarios con gustos similares a Ana?

Esta es la base de los sistemas de filtrado colaborativo!

¿Cómo implementar estas operaciones?

Siguiente paso
¿Cómo implementar TELL, ASK y ASKVARS?

$\rightarrow$ Necesitamos algoritmos de **inferencia** y **unificación**

Inferencia en Lógica de Primer Orden

El Problema de la Inferencia en Lógica de Primer Orden

Hemos visto cómo **representar** conocimiento en FOL, pero ¿cómo **razonamos** con él?

Recordatorio de Lógica Proposicional:

- Teníamos algoritmos como DPLL, resolución, model checking
- Decidible: siempre terminan con una respuesta (True/False)
- Espacio de búsqueda finito (posibles modelos)

Pregunta Central
¿Podemos aplicar las mismas técnicas a FOL?

Spoiler: No directamente. FOL es **mucho más complejo**.

El Problema de la Inferencia en Lógica de Primer Orden

Recordatorio: Queremos determinar si una consulta se puede inferir de la KB.

Base de Conocimiento simple (dominio familiar):

- **Constante**: *John*
- **Hecho**: $Human(John)$
- ...
- **Regla**: $\forall x\ Human(x) \Rightarrow Human(Mother(x))$

Query: ¿Qué podemos deducir de esta KB?

Sustituciones Infinitas

Intento ingenuo: Probar todas las posibles sustituciones de x en la regla
$\forall x\ Human(x) \Rightarrow Human(Mother(x))$

Sustituciones posibles:

$\{x/John\}$ $Human(John) \Rightarrow Human(Mother(John))$

$\{x/Mother(John)\}$ $Human(Mother(John)) \Rightarrow Human(Mother($

$\{x/Mother(Mother(John))\}$ $\cdots$

$\{x/Mother(Mother(Mother(John)))\}$ $\cdots$

$\vdots$

Problema: ¡Lista Infinita de Sustituciones!
No podemos enumerar todas las posibles instanciaciones de las variables.

Necesitamos un enfoque más inteligente que trabaje **simbólicamente** con las variables.

Semidecidibilidad de la Lógica de Primer Orden

El problema de las sustituciones infinitas tiene consecuencias profundas:

Teorema: FOL es Semidecidible

- Si una sentencia α es **consecuencia lógica** de la KB, un algoritmo de inferencia completo eventualmente lo **demostrará** (lo encontrará)
- Si α **no es** consecuencia lógica de la KB, el algoritmo podría **no terminar nunca**

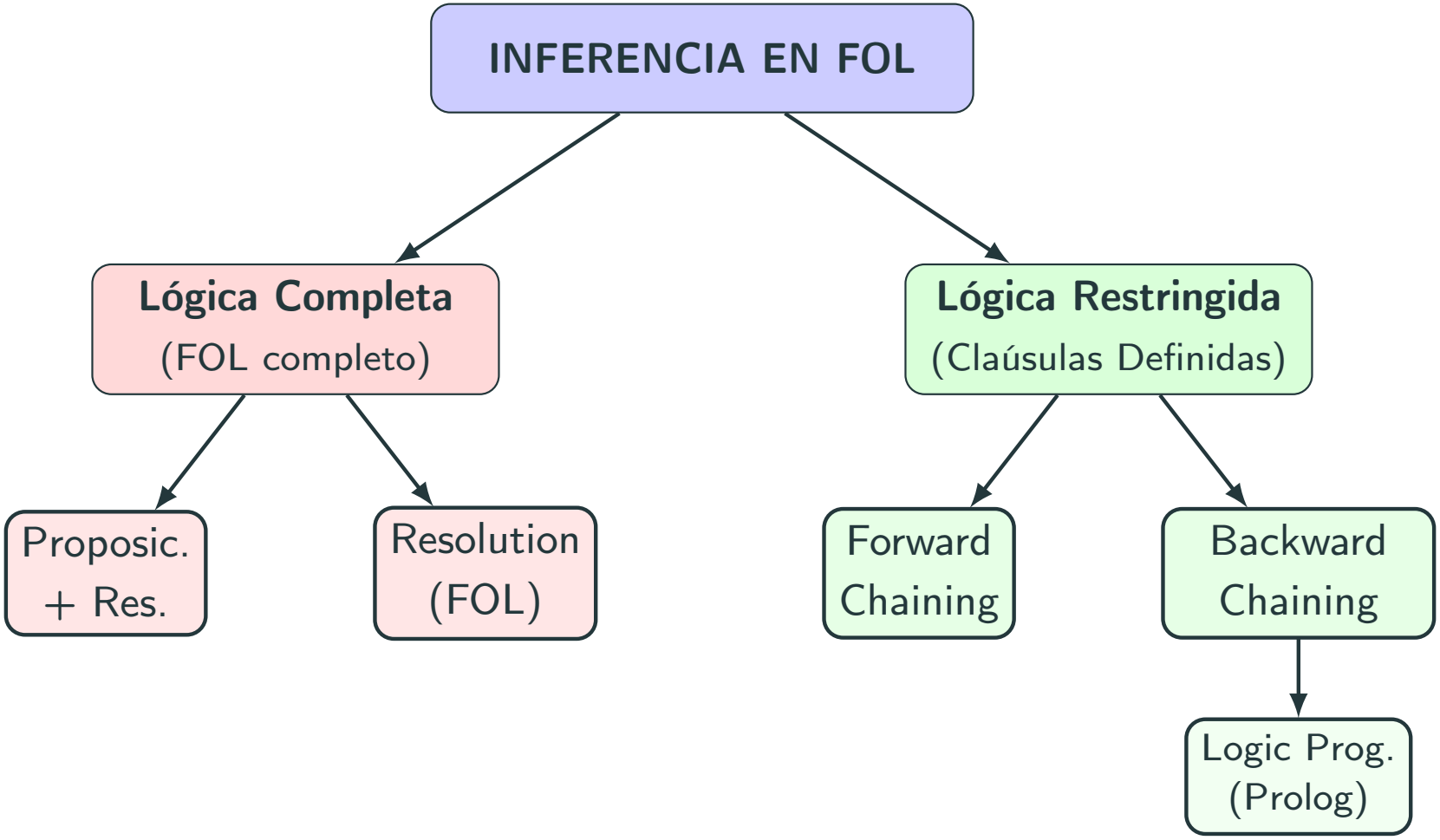
Contraste con Lógica Proposicional:

- Lógica Proposicional es **decidible**: siempre termina (Sí/No)
- FOL es **semidecidible**: solo garantiza respuesta positiva

Implicación práctica: Necesitamos estrategias inteligentes que:

- Eviten enumeración exhaustiva
- Sean eficientes en casos prácticos comunes
- Tal vez sacrifiquen expresividad por decidibilidad

Panorama de Técnicas de Inferencia en FOL



Lógica Completa:
+ Expresividad total
+ Completo
- Costoso/potencialmente infinito

Lógica Restringida:
- Expresividad limitada
+ Completo para el fragmento
+ Eficiente en práctica

Sustituciones y Unificación

Sustituciones y Unificación

Sustituciones

Definición de Sustitución

Sustitución
Una **sustitución** θ es un conjunto finito de pares *variable/término*:

$$\theta = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$$

donde cada v_i es una variable y cada t_i es un término.

Notación: $\{x/John, y/Mother(z)\}$ significa:

- La variable x se sustituye por la constante *John*
- La variable y se sustituye por el término $Mother(z)$

Example: Sustituciones

- $\theta_1 = \{x/Ana\}$ - sustituye x por *Ana*
- $\theta_2 = \{p/Inception, d/Nolan\}$ - sustituye p por *Inception* y d por *Nolan*
- $\theta_3 = \{m/Director(Inception)\}$ - sustituye m por el término funcional

Aplicando Sustituciones

Notación
Sea $SUBST(\theta, \alpha)$ el resultado de aplicar la sustitución θ a la sentencia α.

Example: Sustituciones
Sea $\theta = \{x/Ana, y/Inception\}$

- $SUBST(\theta, Likes(x, y)) = Likes(Ana, Inception)$
- $SUBST(\theta, Likes(x, Matrix)) = Likes(Ana, Matrix)$ (solo x se sustituye)
- $SUBST(\theta, Likes(Carlos, y)) = Likes(Carlos, Inception)$ (solo y se sustituye)

Exercise: Aplicando Sustituciones
Dadas las siguientes sustituciones, calcula $SUBST(\theta, \alpha)$:

1. $\theta_1 = \{x/John, y/Mary\}$
   $\alpha_1 = Human(x) \Rightarrow Human(Mother(x))$
2. $\theta_3 = \{m/Matrix, u/Carlos, g/SciFi\}$
   $\alpha_3 = Likes(u, m) \wedge HasGenre(m, g)$

Sustituciones con Cuantificadores

Aplicar sustituciones a sentencias con cuantificadores requiere atención especial.

Cuantificador Universal ($\forall$): Solo sustituimos variables **libres**

- Sea $\theta = \{x/Ana\}$ y $\alpha = \forall x\ Likes(x, Inception)$
- $SUBST(\theta, \alpha) = \forall x\ Likes(x, Inception)$ (¡x está ligada!)
- Pero si $\alpha = \forall y\ Likes(x, y)$, entonces $SUBST(\theta, \alpha) = \forall y\ Likes(Ana, y)$

Cuantificador Existencial ($\exists$): Skolemización

- Para eliminar $\exists$, introducimos una **constante de Skolem**
- $\exists x\ Likes(Ana, x)$ se convierte en $Likes(Ana, SK_1)$
- SK_1 es una constante nueva que "testifica" la existencia
- **Intuición**: "Si existe algo, démosle un nombre específico"

Example: Skolemización
$\exists m\ DirectedBy(m, Nolan) \land HighlyRated(m)$
Skolemización: $DirectedBy(SK_{NolanMovie}, Nolan) \land HighlyRated(SK_{NolanMovie})$

Sustituciones con Cuantificadores: Ejercicio

Exercise: Sustituciones con Cuantificadores

Para cada sentencia, indica si la sustitución afecta a la expresión y por qué:

1. $\theta = \{x/John\}$, $\alpha = \forall x\ Human(x) \Rightarrow Human(Mother(x))$
   ¿Cuál es $SUBST(\theta, \alpha)$?
2. $\theta = \{y/SciFi\}$, $\alpha = \forall x\ HasGenre(x, y) \Rightarrow Likes(Carlos, x)$
   ¿Cuál es $SUBST(\theta, \alpha)$?
3. $\alpha = \exists p\ Likes(Ana, p) \wedge HasGenre(p, SciFi)$
   Aplica Skolemización. ¿Cuál es la sentencia resultante?
4. $\alpha = \forall u\ \exists m\ Likes(u, m) \wedge DirectedBy(m, Nolan)$
   Aplica Skolemización al cuantificador existencial.

Sustituciones y Unificación

Unificación

El Problema de Unificación

Ejemplo intuitivo: Considera la siguiente KB y query:

KB:

- $\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$
- $\forall x\ \mathit{DirectedBy}(x, \mathit{Nolan}) \Rightarrow \mathit{HighlyRated}(x)$

Query: $\mathit{ASKVARS}(\mathit{HighlyRated}(m))$ - ¿Qué películas tienen alta valoración?

Pregunta: ¿Cómo respondiste mentalmente esta pregunta?

El Problema de Unificación

Ejemplo intuitivo: Considera la siguiente KB y query:

KB:

- $\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$
- $\forall x\ \mathit{DirectedBy}(x, \mathit{Nolan}) \Rightarrow \mathit{HighlyRated}(x)$

Query: $\mathit{ASKVARS}(\mathit{HighlyRated}(m))$ - ¿Qué películas tienen alta valoración?

Pregunta: ¿Cómo respondiste mentalmente esta pregunta?

Lo que hiciste: ¡**Unificaste** $\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$ con $\mathit{DirectedBy}(x, \mathit{Nolan})$!

Encontraste que $x = \mathit{Inception}$ hace que ambas expresiones coincidan.

El Problema de Unificación

Motivación: Para aplicar reglas de inferencia en FOL, necesitamos hacer coincidir patrones con variables.

Problema de Unificación
Dadas dos expresiones p y q, encontrar una sustitución θ tal que:

$$SUBST(\theta, p) = SUBST(\theta, q)$$

Notación: $UNIFY(p, q) = \theta$ donde $SUBST(\theta, p) = SUBST(\theta, q)$

Si tal θ existe, decimos que p y q son **unificables**, y θ es un **unificador**.

Example: Unificación

1. $UNIFY(Likes(x, Inception), Likes(Ana, Inception)) = \{x/Ana\}$
   Resultado: $SUBST(\{x/Ana\}, Likes(x, Inception)) = Likes(Ana, Inception)$
2. $UNIFY(DirectedBy(p, Nolan), DirectedBy(Interstellar, d)) = \{p/Interstellar, d/Nolan\}$
   Resultado: $DirectedBy(Interstellar, Nolan)$
3. $UNIFY(Likes(x, y), Likes(Ana, Matrix)) = \{x/Ana, y/Matrix\}$
   Resultado: $Likes(Ana, Matrix)$

Ejemplos de Unificación

Exercise: Casos de Unificación

Para cada par de expresiones, determina $UNIFY(p, q)$ si existe, o explica por qué no son unificables:

1. $UNIFY(Human(x), Human(John))$
2. $UNIFY(Likes(Ana, y), Likes(x, Matrix))$
3. $UNIFY(Likes(x, x), Likes(Ana, Matrix))$
4. $UNIFY(Director(Inception), Nolan)$
5. $UNIFY(Mother(x), Mother(Mother(John)))$
6. $UNIFY(Human(x), Human(Mother(x)))$ (**occur-check**)
7. $UNIFY(Knows(John, x), Knows(x, Elizabeth))$ (**standardizing-apart**)

Unificación y Modus Ponens: Ejemplo

Base de Conocimiento:

- **Hecho 1**: $DirectedBy(Inception, Nolan)$
- **Hecho 2**: $HighlyRated(Inception)$
- **Regla**: $\forall m, d\ (DirectedBy(m, d) \land HighlyRated(m)) \Rightarrow \exists u\ Likes(u, m)$

Example: Pasos para aplicar Modus Ponens

Unificación y Modus Ponens: Ejercicio

Exercise: Aplicando Modus Ponens con Unificación
Para cada caso, aplica Modus Ponens usando *UNIFY*:

1. **KB**: $Likes(Ana, Inception)$
   Regla: $\forall x, m\ Likes(x, m) \Rightarrow WatchedBy(m, x)$
   ¿Qué podemos concluir?
2. **KB**: $HasGenre(Matrix, SciFi)$
   Regla: $\forall p, g\ HasGenre(p, g) \land HighlyRated(p) \Rightarrow Likes(Carlos, p)$
   ¿Podemos concluir algo? ¿Por qué?
3. **KB**: $DirectedBy(Interstellar, Nolan)$, $HighlyRated(Interstellar)$
   Regla: $\forall m, d\ DirectedBy(m, d) \land HighlyRated(m) \Rightarrow \exists u\ Likes(u, m)$
   ¿Qué podemos concluir?

Restricción a Claúsulas Definidas

FOL completo es semidecidible, pero podemos restringir la lógica para obtener algoritmos decidibles y eficientes.

Recordatorio: En lógica proposicional vimos FC y BC para **Horn Clauses**.

En Lógica de Primer Orden, las Horn Clauses se clasifican en:

- **Claúsulas Definidas** (exactamente 1 literal positivo):
  - Hechos y reglas
  - Usadas en la KB
  - FC y BC trabajan con estas
- **Claúsula objetivo** (0 literales positivos):
  - Representan queries/objetivos
  - $P_1 \wedge ... \wedge P_n \Rightarrow \mathit{False}$
  - BC las usa internamente

En adelante: Usaremos “Claúsulas Definidas"para referirnos a las cláusulas de la KB en FOL.

Restricción a Claúsulas Definidas

Cláusulas Definidas

- **Atómica**: Un literal positivo único (hecho), o
- **Implicación**: Cuyo antecedente es una **conjunción de literales positivos** y cuyo consecuente es un **literal positivo único**

$P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$ donde $P_1, \ldots, P_n, Q$ son literales positivos.

Restricciones:

- No se permiten cuantificadores existenciales $\exists$
- Los cuantificadores universales $\forall$ se dejan implícitos

Restricción a Claúsulas Definidas

Example: Claúsulas Definidas

- $DirectedBy(Inception, Nolan)$
- $DirectedBy(x, Nolan) \Rightarrow HighlyRated(x)$ (1 literal en antecedente)
- $Quality(x) \wedge HasGenre(x, y) \Rightarrow Recommended(x, y)$ (2 literales)

Si ves una variable x en una Claúsula Definida, significa que hay un $\forall x$ implícito.

AntiEjemplos

- $Likes(Ana, Inception) \vee Likes(Ana, Matrix)$ (disyunción)
- $\neg Likes(Carlos, p) \Rightarrow Dislikes(Carlos, p)$ (literal negativo)

¿Por Qué Restringir a Claúsulas Definidas?

Propiedades de Claúsulas Definidas

- **Decidibles**: Los algoritmos de inferencia siempre terminan
- **Completos**: Si algo es consecuencia lógica, lo encontraremos
- **Eficientes**: Algoritmos polinomiales en muchos casos prácticos
- **Suficientemente expresivos**: Muchas aplicaciones reales se pueden modelar

Contraste con FOL completo:

	FOL Completo	**Claúsulas Definidas**
Expresividad	Total	Restringida
Decidibilidad	Semidecidible	**Decidible**
Completitud	Completo	**Completo**
Eficiencia	Potencialmente infinito	Eficiente

Siguiente paso: Algoritmos de inferencia para Claúsulas Definidas:

- Forward Chaining (data-driven)
- Backward Chaining (goal-driven)

Sustituciones y Unificación

Forward Chaining

Forward Chaining: Intuición

Forward Chaining es un algoritmo de inferencia **guiado por datos** (data-driven).

Idea Principal

- Comenzar con los **hechos conocidos** en la KB
- Aplicar reglas cuyas **premisas están satisfechas**
- Añadir las **conclusiones** derivadas a la KB
- Repetir hasta que no se puedan derivar más hechos

En FOL usa unificación

Propiedades:

- **Sound** (correcto): Solo deriva consecuencias lógicas
- **Complete** (completo): Encuentra todas las consecuencias para Claúsulas Definidas
- **Decidible**: Siempre termina para Claúsulas Definidas

Forward Chaining: Ejemplo Simple

Example: Forward Chaining Simple

Dada la siguiente KB, deriva todos los hechos posibles usando Forward Chaining:

KB:

1. $\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$
2. $\mathit{HighlyRated}(\mathit{Inception})$
3. $\forall x\ \mathit{DirectedBy}(x, \mathit{Nolan}) \Rightarrow \mathit{Quality}(x)$
4. $\forall x\ \mathit{HighlyRated}(x) \Rightarrow \mathit{Recommended}(x)$

Forward Chaining: Ejemplo Complejo

Example: Forward Chaining Complejo

Dada la siguiente KB, deriva todos los hechos posibles usando Forward Chaining:

KB:

1. $DirectedBy(Inception, Nolan)$
2. $DirectedBy(Interstellar, Nolan)$
3. $HasGenre(Inception, SciFi)$
4. $HasGenre(Interstellar, SciFi)$
5. $Likes(Ana, Inception)$
6. $\forall x\ DirectedBy(x, Nolan) \Rightarrow HighlyRated(x)$
7. $\forall x\ HighlyRated(x) \Rightarrow Quality(x)$
8. $\forall x, u\ (Likes(u, x) \land Quality(x)) \Rightarrow WatchedBy(x, u)$
9. $\forall x, y\ (Quality(x) \land HasGenre(x, y)) \Rightarrow Recommended(x, y)$

Sustituciones y Unificación

Backward Chaining

Backward Chaining: Intuición

Backward Chaining es un algoritmo de inferencia **guiado por objetivos** (goal-driven).

Idea Principal

- Comenzar con una **query** (objetivo a probar)
- Buscar reglas cuya **conclusión unifique** con el objetivo
- Establecer las **premisas** de esas reglas como **nuevos subobjetivos**
- Repetir recursivamente hasta encontrar hechos en la KB

En FOL usa unificación

Propiedades:

- **Sound** (correcto): Solo prueba consecuencias lógicas válidas
- **Complete** (completo): Encuentra prueba si existe (para Claúsulas Definidas)
- **Decidible**: Termina para Claúsulas Definidas (con detección de ciclos)
- Más **eficiente** que FC cuando solo queremos probar una query específica

Backward Chaining: Ejemplo Simple

Example: Backward Chaining Simple
Dada la siguiente KB, usa Backward Chaining para probar la query:

KB:

1. $\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$
2. $\mathit{HighlyRated}(\mathit{Inception})$
3. $\forall x\ \mathit{DirectedBy}(x, \mathit{Nolan}) \Rightarrow \mathit{Quality}(x)$
4. $\forall x, y\ (\mathit{Quality}(x) \land \mathit{HighlyRated}(x)) \Rightarrow \mathit{Recommended}(x)$

Query: $\mathit{Recommended}(\mathit{Inception})$?

Backward Chaining: Ejemplo Complejo

Example: Backward Chaining Complejo
Dada la siguiente KB, usa Backward Chaining para probar la query:

KB:

1. $\mathit{DirectedBy}(\mathit{Inception}, \mathit{Nolan})$
2. $\mathit{DirectedBy}(\mathit{Interstellar}, \mathit{Nolan})$
3. $\mathit{HasGenre}(\mathit{Inception}, \mathit{SciFi})$
4. $\mathit{HighlyRated}(\mathit{Inception})$
5. $\mathit{Likes}(\mathit{Ana}, \mathit{Inception})$
6. $\forall x\ \mathit{DirectedBy}(x, \mathit{Nolan}) \Rightarrow \mathit{Quality}(x)$
7. $\forall x\ \mathit{HighlyRated}(x) \Rightarrow \mathit{Quality}(x)$
8. $\forall x, u\ (\mathit{Likes}(u, x) \land \mathit{Quality}(x)) \Rightarrow \mathit{WatchedBy}(x, u)$
9. $\forall x, y\ (\mathit{Quality}(x) \land \mathit{HasGenre}(x, y)) \Rightarrow \mathit{MustWatch}(x, y)$

Query: $\mathit{MustWatch}(\mathit{Inception}, \mathit{SciFi})$?

Comparación: Forward vs Backward Chaining

Exercise: Forward vs Backward Chaining
Completa la siguiente tabla comparativa:

Aspecto	Forward Chaining	Backward Chaining
Estrategia		
Punto de inicio		
Dirección		
Eficiencia		
Uso típico		
Complejidad temporal		
Completitud		
Decidibilidad		

Sustituciones y Unificación

Logic Programming y Prolog

Logic Programming: De la Teoría a la Práctica

Logic Programming es un paradigma donde los programas son bases de conocimiento lógicas.

Prolog (Programming in Logic)
Prolog es un lenguaje de programación que implementa Backward Chaining para KBs con Claúsulas Definidas.

Relación con lo que hemos visto:

- KB = Programa Prolog (hechos + reglas)
- Query = Objetivo a probar
- Motor de inferencia = Backward Chaining con depth-first search
- Unificación automática de variables

Example: Los Simpsons en Prolog

En Lógica de Primer Orden: **Hechos**:

- $Parent(Homer, Bart)$, $Parent(Homer, Lisa)$, $Parent(Homer, Maggie)$
- $Parent(Marge, Bart)$, $Parent(Marge, Lisa)$, $Parent(Marge, Maggie)$
- $Male(Homer)$, $Male(Bart)$
- $Female(Marge)$, $Female(Lisa)$, $Female(Maggie)$

Reglas:

- $\forall x, y\ Parent(x, y) \Rightarrow Ancestor(x, y)$
- $\forall x, y\ (Parent(x, y) \wedge Male(x)) \Rightarrow Father(x, y)$
- $\forall x, y\ (Parent(x, y) \wedge Female(x)) \Rightarrow Mother(x, y)$

Prolog: Creando tu Primera KB

Exercise: Crear una Base de Conocimiento en Prolog

Información a modelar:

- Leonardo DiCaprio actuó en Inception
- Leonardo DiCaprio actuó en Titanic
- Matthew McConaughey actuó en Interstellar
- Si un actor actuó en una película dirigida por Nolan, entonces es un actor versátil
- Inception fue dirigida por Nolan
- Interstellar fue dirigida por Nolan

Tareas:

1. Escribe los hechos y reglas necesarios en sintaxis Prolog
2. Resuelve las siguientes queries:
   - ¿Leonardo DiCaprio es un actor versátil?
   - ¿Qué actores son versátiles?
   - ¿En qué películas actuó Matthew McConaughey?

Diferencias entre Prolog y Lógica de Primer Orden

Prolog NO es FOL puro. Tiene varias diferencias importantes: **1. Database Semantics (Closed World Assumption)**:

- Lo que no está en la KB (o no se puede probar) es **falso**
- Ejemplo: Si `likes(ana, matrix)` no está ni se puede derivar $\rightarrow$ es falso

2. Negación por Fallo (\+):

```
not_watched(User, Movie) :- \+ watched(User, Movie).
```

- `\+ goal` significa "no se puede probar goal"
- CWA en acción: ausencia de prueba = falso
- Ejemplo: `?- \+ watched(carlos, inception).` $\rightarrow$ Yes (si no hay evidencia)

Diferencias entre Prolog y FOL (cont.)

3. Sin Occur-Check:

- Prolog omite occur-check por eficiencia
- Puede producir inferencias insólidas (raramente un problema)
- Ejemplo: `X = f(X)` puede ünificar"(incorrectamente)

4. Built-in Aritméticos:

- Evaluación por código, no por inferencia lógica

```
?- X is 4 + 3.
X = 7              % Funciona: evalúa la expresión

?- 5 is X + Y.
Error              % Falla: no resuelve ecuaciones
```

Diferencias entre Prolog y FOL (cont.)

5. Side Effects (`assert/retract`):

- Modificar la KB durante la ejecución
- No tienen equivalente en lógica pura

```
?- assert(new_fact(abc)).      % Añade hecho a la KB
?- retract(old_fact(xyz)).     % Elimina hecho de la KB
```

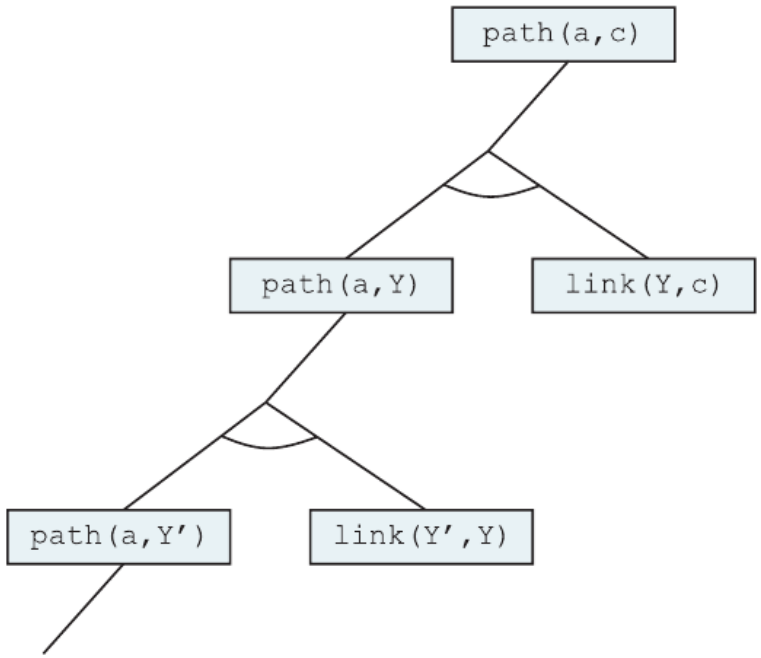
El Talón de Aquiles de Prolog: Orden de las Reglas

Prolog usa depth-first search sin detección de ciclos infinitos.

Ejemplo: Definir el concepto de "camino" en un grafo

Versión que NO TERMINA:

```
path(X, Z) :- path(X, Y), link(Y, Z).  % Caso recursivo primero
path(X, Z) :- link(X, Z).              % Caso base segundo
```

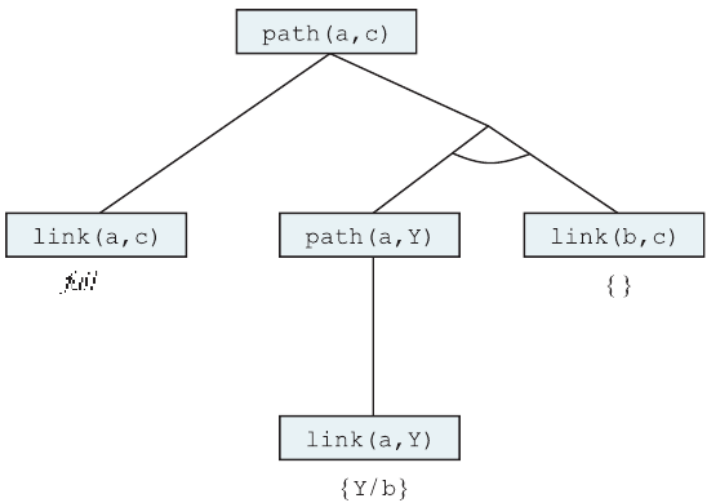


(b)

El Talón de Aquiles de Prolog: La Solución

Versión que SÍ TERMINA (mismo código, distinto orden):

```
path(X, Z) :- link(X, Z).              % Caso base primero
path(X, Z) :- path(X, Y), link(Y, Z).  % Caso recursivo segundo
```



Query `?- path(a, c).` ahora **termina correctamente**:
Primero intenta el caso base, luego el recursivo.

Lección
El orden de las reglas y objetivos importa en Prolog!

Alternativa a Prolog: Datalog

Datalog
Datalog es un subconjunto de Prolog sin términos funcionales (solo predicados y constantes).

Restricción: No se permiten functors como `father(john)` como argumento.

Ventajas de Datalog:

- **Siempre termina**: Decidible (número finito de hechos ground posibles)
- **Eficiente**: Algoritmos de evaluación muy optimizados
- **Usado en práctica**: Bases de datos modernas, sistemas de análisis

Ejemplos de uso:

- Neo4j Cypher (queries en grafos)
- Datomic (base de datos temporal)
- Soufflé (análisis estático de programas)
- LogicBlox (análisis de datos empresariales)