



# Tecnologías de Programación

Paradigma Lógico – ProLog

Introducción



# Programación Declarativa

- Programación Imperativa
  - Programar es: codificar el COMO
    - Los programas son recetas a ejecutar
    - No se describe el problema, sino la solución
  - Paradigmas secuencial, estructurado, etc.
- Programación Declarativa
  - Programar es: describir el QUE
    - Los programas expresan conocimiento del dominio
    - Los lenguajes utilizan las expresiones, no las ejecutan
  - Paradigmas funcional y lógico



# Breve Reseña Histórica

- 1870 Charles S. Peirce realiza una descripción de una notación relativa a la lógica como una ampliación de los conceptos del álgebra de Boole para posibles cálculos lógicos.
- 1889 Giuseppe Peano publica “Principios de aritmética , nueva exposición del método”, donde se aplica por primera vez, una sistematización de las matemáticas con una notación funcional.
- 1896 Charles S. Peirce publica “Una teoría de inferencia probable”, donde propone una notación gráfica para las matemáticas llamada grafos existenciales, que él la llamó “la lógica del futuro”.
- 1951 Alfred Horn publica “Sobre sentencias las cuales son verdaderas de la unión directa de las álgebras”, en la cual presenta un modelo lógico para el tratamiento de oraciones del lenguaje natural.
- 1972 En la Universidad de Marsella, Alain Colmerauer y un grupo de investigadores presentan el lenguaje PROLOG (que utiliza el método de Horn) como una herramienta para resolver ciertos problemas en el área de la inteligencia artificial, originalmente vinculados al tratamiento computacional del lenguaje natural. Más tarde en la Universidad de Edimburgo se perfecciona el lenguaje y se comienza a escribir el compilador.
- 1974 Robert Kowalski publica “Predicados lógicos como un lenguaje de programación”, donde crea el Paradigma Lógico como un paradigma de programación.
- 1979 Robert Kowalski publica “Lógica para la resolución de problemas y junto con la Universidad de Edimburgo escriben un nuevo compilador para el Prolog, adaptándolo al paradigma lógico planteando la ecuación Lógica + control + estructuras de datos = programas.
- 1986 Borland Internacional presenta el primer compilador comercial Prolog para PC.



# Lógica de Primer Orden

- El lógico, es un paradigma de programación basado en la lógica de primer orden.
- La lógica de predicados o de primer orden (LPO, L1) es una generalización de la lógica de proposiciones (LP, L0). Introduce nuevos elementos que permiten especificar la estructura interna de los elementos y las relaciones entre los mismos.
- Esta nueva lógica tendría que permitir una descripción más fina de la realidad, pudiendo distinguir los objetos o términos (por ejemplo, los hombres) de sus propiedades o predicados (por ejemplo, la propiedad de ser mortales).



# Lógica de Primer Orden

- La lógica proposicional puede ser no apropiada para expresar ciertos tipos de conocimiento. Por ejemplo:
  - Juan es padre de Tomás
- En lógica proposicional solo se puede dar a esta frase un valor de verdad (V o F).
- En lógica de primer orden, tenemos herramientas para expresar que Juan y Tomás son elementos de un dominio, y que existe una relación entre dichos elementos (relación padre/hijo)



# Lógica de Primer Orden

- La LP también puede no ser apropiada para modelar cierto tipo de razonamiento
  - Propositiones:
    - Juan ayuda a todas las personas que gustan de la lógica
    - Martín es persona
    - A Martín le gusta la lógica
  - Conclusión:
    - Juan ayuda a Martín
- El razonamiento es correcto, pero es muy difícil o imposible de especificar sin las herramientas brindadas por la LPO





# Lógica de Primer Orden

- Resumiendo, la LP no permite referirse en forma sencilla a elementos de un dominio.
- Más aún, si los elementos del dominio son infinitos, simplemente no puede expresar conocimiento acerca de todos los individuos (no tenemos cuantificadores).
- La LP tampoco es capaz de representar propiedades de objetos.
- La lógica de primer orden (LPO) soluciona estos problemas en este sentido:
  - Permite hacer cuantificación sobre los objetos de un dominio
    - Todos los caballos son animales.
    - Algunos manzanas son rojas.
  - Permite representar propiedades a través de relaciones y funciones.



# Lógica de Primer Orden

- Supongamos las siguientes declaraciones:
  - a) Toda madre ama a sus hijos
  - b) María es madre y Juan es hijo de María
- Aplicando algo de razonamiento podemos inferir que:
  - **c) María ama a Juan**





# Lógica de Primer Orden

- Las sentencias en lenguaje natural expresadas en “a” y “b” describen un universo de personas y relaciones entre las mismas.
- El ejemplo refleja la idea principal de la programación lógica:
  - Describir universos de objetos y relaciones sobre los mismos, y aplicar un sistema de programación para inferir conclusiones como la “c”.



# Lógica de Primer Orden

## Formalización de Sentencias

- Se debe definir en forma precisa las sentencias que conformarán el programa
- Lenguaje formal
  - Elimina ambigüedad
  - Factible de ser interpretado por un computador
- Conceptos a tener en cuenta
  - Alfabeto
  - Sintaxis
  - Semántica



# Lógica de Primer Orden

## Alfabeto

- Brinda un conjunto de símbolos con los que trabajar.
- Elementos básicos del alfabeto:
  - **Constantes**: representan objetos concretos del dominio. (\*)
  - **Variables**: representan objetos no específicos del dominio. (\*)
  - **Predicados**: todo predicado tiene asociado un número **n** denominado aridad, y que se expresa como “predicado/**n**”.
    - Predicados monádicos,  $n = 1$ : representan propiedades de objetos.
    - Predicados poliádicos,  $n > 1$ : representan relaciones entre objetos.
  - **Cuantificadores y conectores lógicos**:  $\forall, \exists, \wedge, \vee, \neg, \rightarrow, \leftrightarrow$
  - **Funtores**: algunas relaciones son funcionales. (\*)
    - Una persona tiene exactamente otra persona que es su padre.
    - Un punto en el plano se define con dos valores.
  - **(\*)Términos**: Un término es una expresión lógica que se refiere a un objeto. Por lo tanto, son términos las constantes, las variables y los funtores. (Ejemplo: punto(2, 3))



# Ejercicios 1 (10 min.)

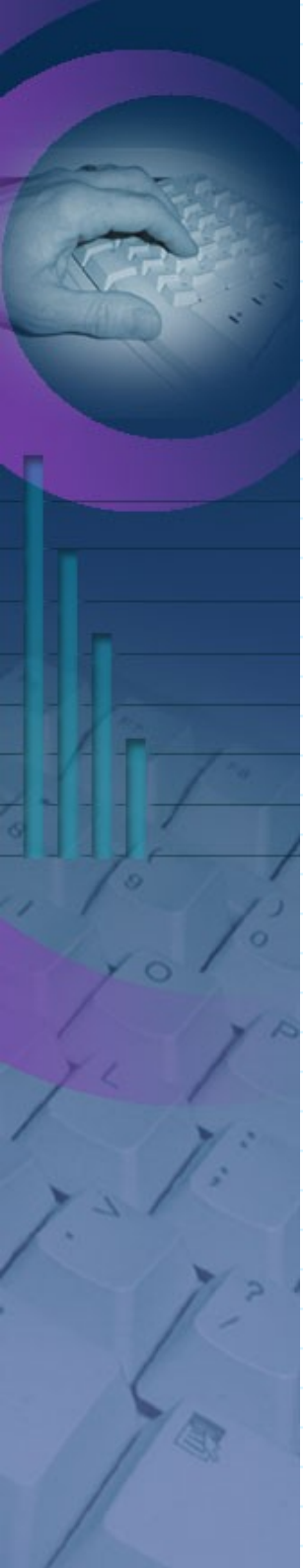
- Escriba las tablas de verdad de las siguientes proposiciones:
  - $A \wedge B$
  - $A \vee B$
  - $A \vee \neg B$
  - $B \rightarrow A$
- Reescriba la siguiente proposición de tal forma que no queden signos de negación
  - $A \vee \neg B \vee \neg C \vee \neg D$



# Lógica de Primer Orden

## Sintaxis

- En lenguaje natural solo cierta combinación de palabras conforman frases con significado. La contraparte en lógica de predicado son las fórmulas bien formadas (wff)
- Sea  $T$  conjunto de términos del alfabeto  $A$ , el conjunto  $F$  de wff es el menor conjunto tal que:
  - Si  $p/n$  es un predicado en  $A$  y  $t_1, \dots, t_n \in T$ , entonces  $p(t_1, \dots, t_n) \in F$ 
    - Éste tipo de fórmula es llamada fórmula atómica
  - Si  $G$  y  $H \in F$ , entonces también pertenecen:
    - $(\neg G)$ ,  $(G \wedge H)$ ,  $(G \vee H)$ ,  $(G \leftarrow H)$ ,  $(G \leftrightarrow H)$
  - Si  $G \in F$  y  $X$  es una variable en  $A$ ,
    - entonces  $(\forall X G) \in F$  y  $(\exists X G) \in F$



# Lógica de Primer Orden

## Ejemplo de Formalización

- a) Todas las madres aman a sus hijos
- b) maría es madre y juan es hijo de maría
- c) maría ama a juan

Las sentencias podrían formalizarse de la siguiente forma:

- a)  $\forall X ( \forall Y ( ( \text{madre}( X ) \wedge \text{hijo\_de}( Y, X ) ) \rightarrow \text{ama}( X, Y ) ) )$
- b)  $\text{madre}(\text{maría}) \wedge \text{hijo\_de}(\text{juan}, \text{maría})$
- c)  $\text{ama}(\text{maría}, \text{juan})$





# Lógica de Primer Orden

## Semántica

- Las fórmulas hacen referencia a algún universo y en ese marco pueden ser V o F
- El significado de una fórmula se define en relación a un dominio
- La semántica establece las conexiones entre wffs y alguna estructura (formadas por objetos y relaciones entre los mismos) para definir el significado de las fórmulas



# Lógica de Primer Orden

## Interpretación y Valoración

- Una interpretación  $\mathfrak{I}$  de un alfabeto  $A$  es un dominio no vacío  $D$  y un mapeo que los asocia de la siguiente forma:
  - Cada constante  $c \in A$  con un elemento  $c' \in D$
  - Cada functor  $f \in A$  con una función  $f': D^n \rightarrow D$
  - Cada predicado  $p \in A$  con una relación  $p' \subseteq D^n$
- Una Valoración  $\varphi$  es una función que asigna objetos de una interpretación a variables del lenguaje



# Lógica de Primer Orden

## Modelo – Consecuencia Lógica

- Dado un conjunto de fórmulas  $P$ , se dice que una interpretación  $\mathfrak{I}$  es un modelo de  $P$  si y solo si cada fórmula de  $P$  es verdadera en  $\mathfrak{I}$ .
- ¿que otras fórmulas además de las de  $P$  son verdaderas en los modelos de  $P$ ?
- Dado un conjunto de fórmulas  $P$ , una fórmula  $F$  es consecuencia lógica de  $P$  si y solo si  $F$  es Verdadero en cada modelo de  $P$ .
- **Importante:** Al proponer una nueva fórmula, la máquina no conoce la interpretación sobre la que se trabaja, pero como asume que es modelo del conjunto de fórmulas, solo debe demostrar que la nueva fórmula es una consecuencia lógica, y como tal debe ser verdadera en cualquier modelo.



# Lógica de Primer Orden

## Cláusulas definidas

- Una Cláusula es una fórmula de la forma:
  - $\forall(L_0 \vee \dots \vee L_n)$  donde cada  $L_i$  es una fórmula atómica positiva (literal positivo) o la negación de una fórmula atómica (literal negativo)
  - Todas las variables que ocurran en cualquiera de las fórmulas  $L_i$  son cuantificadas (implícitamente) en forma universal.
- Cláusula Definida
  - Una cláusula definida es una cláusula con un solo literal positivo
  - Es decir:  $\forall(L_0 \vee \neg L_1 \dots \vee \neg L_n)$



# Lógica de Primer Orden

## Cláusulas Definidas

- Entonces, llamaremos cláusulas definidas, a las fórmulas con el siguiente formado (sintaxis):
  - $A_0 \vee \neg A_1 \vee \dots \vee \neg A_n$
- O equivalentemente
  - $A_0 \leftarrow A_1 \wedge \dots \wedge A_n$
  - donde  $A_0, \dots, A_n$  son formulas atómicas, y todas las variables que ocurran en una fórmula son (implícitamente) cuantificadas en forma universal.
  - $A_0$  es llamado cabecera de la cláusula y  $(A_1 \wedge \dots \wedge A_n)$  cuerpo de la misma.
  - Un caso particular se da cuando  $n = 0$ , entonces la cláusula no tiene cuerpo y solo queda definida la cabecera.



# Campos de Aplicación

- Fundamentalmente en:

- Sistemas expertos

- Sistemas que imitan el comportamiento de un experto humano

- Procesamiento del lenguaje natural

- Dividir el lenguaje en partes y relaciones, y tratar de comprender su significado

- Y en general en cualquier campo en donde la naturaleza del problema se pueda expresar como un cuerpo de predicados lógicos





# Ventajas y Desventajas

- Ventajas

- Sencillez, potencia y elegancia.
- Cercanía a las especificaciones del problema realizada con lenguajes formales.
- Metodología rigurosa de especificación.
- Facilidad en la implementación de estructuras complejas.

- Desventajas

- Poco eficientes.
- Forma no habitual de encarar el problema/código.



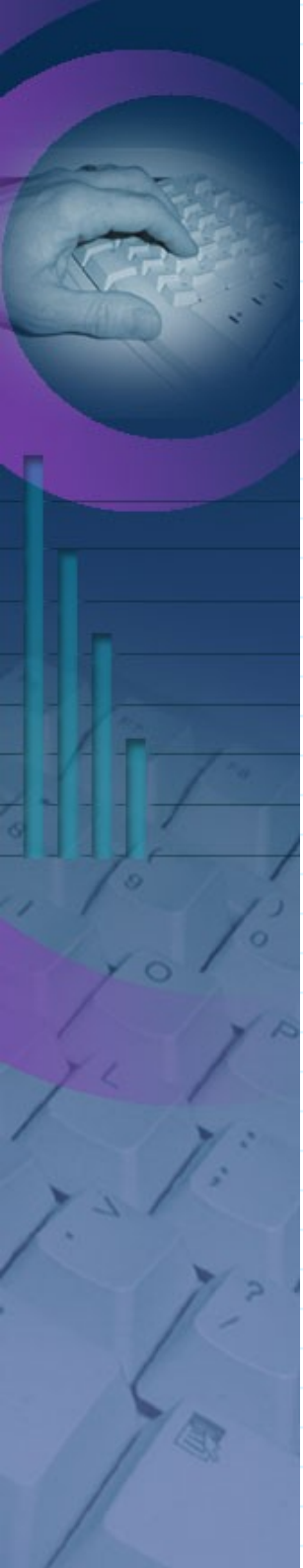
# Programación Lógica

- Solo existen las expresiones lógicas !!!  
entonces:
  - Programa:
    - Conocimientos acerca del problema.
    - Expresado como conjunto de axiomas y reglas lógicas.
  - Como se invoca:
    - Proveyendo un nuevo axioma o regla lógica.
  - En que consiste la ejecución:
    - Tratar de probar el nuevo axioma o regla dada en la invocación.



# Prolog

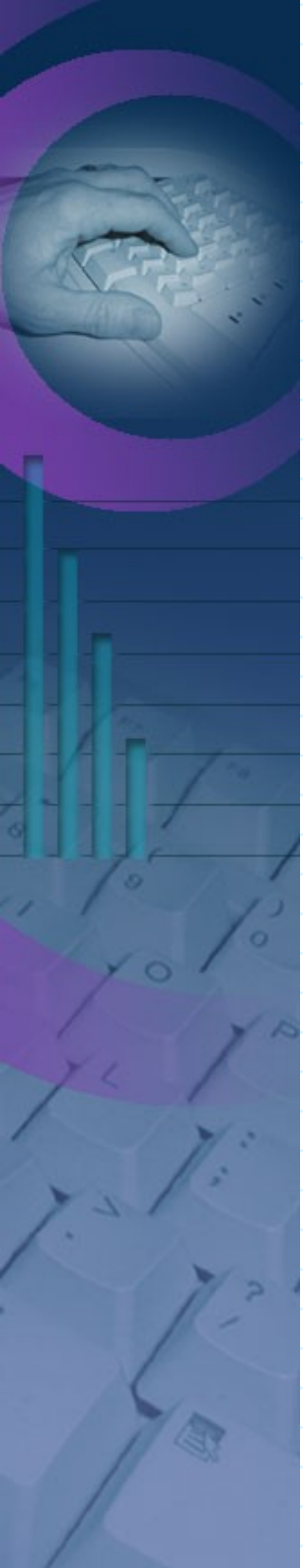
- Lenguaje de programación que respeta el paradigma lógico.
- Muy útil para resolver problemas que implican objetos y relaciones entre los mismos.
- Sintaxis simple que permitirá:
  - Declarar **hechos** sobre objetos y relaciones.
  - Hacer **preguntas** sobre objetos y relaciones.
  - Definir **reglas** sobre objetos y relaciones.



# Prolog

## Cláusulas

- Las cláusulas en Prolog respetan el formato clausular visto anteriormente, consisten en un cuerpo y una cabeza.
- Se deduce entonces que tenemos tres tipos de cláusulas:
  - Hechos: cláusulas sin cuerpo.
  - Reglas: cláusulas con cabeza y cuerpo.
  - Preguntas: cláusulas sin cabeza.



# Prolog

## Hechos

- Los hechos expresan axiomas (se consideran verdades).
- Ej.: Para expresar que Juan es hijo de María, se puede escribir:
  - `progenitor(maría, juan).`
- “progenitor” es el nombre de la relación, “maría” y “juan” son los argumentos.
- La semántica del hecho es dada por el programador.



## Ejercicios 2 (5 min.)

- Escriba los siguientes hechos en prolog:
  - 1) Juan tiene un libro
  - 2) Ana ama a Raúl
  - 3) El oro es valioso
  - 4) El platino es mas valioso que el oro
  - 5) Germán es hijo de Juan y María
  - 6) El sueldo de Carlos es de \$5200





# Prolog

## Preguntas

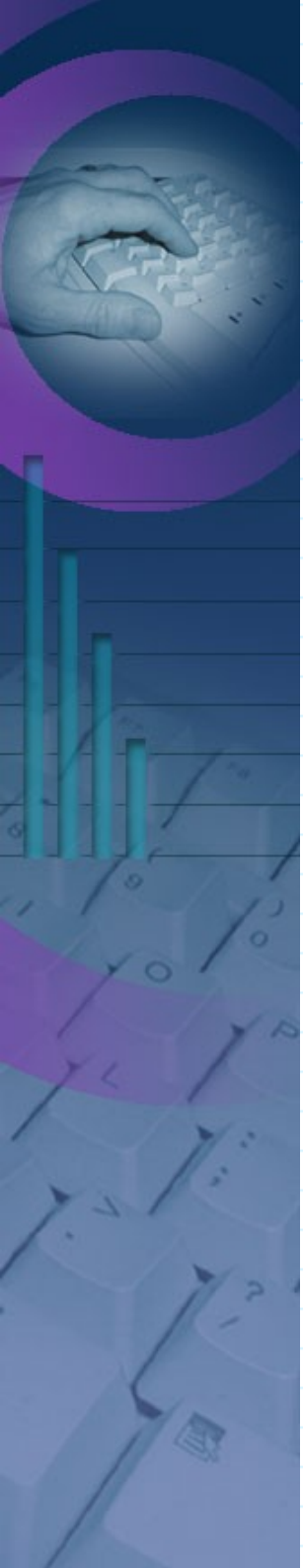
- Sobre un conjunto de hechos se pueden realizar una serie de preguntas.
- Ej.: Para preguntar si Juan tiene un libro escribiríamos:
  - `tiene(juan, libro).`      % notar que es la misma sintaxis que el hecho
- Prolog tratará de unificar nuestra pregunta con su cuerpo de conocimiento, buscando si existe una declaración con el mismo nombre y la misma cantidad de parámetros (aridad).



# Prolog

## Preguntas

- Contestará “Si” (o true) si pudo unificar y “No” (o fail) si no pudo.
- El “No” no implica que el hecho sea falso, sino que no se pudo probar su veracidad con el conocimiento almacenado en la base de datos.
- Como realizamos preguntas mas interesantes, por ejemplo ¿que tiene Juan?
  - `tiene(juan, X).`    % notar que el segundo argumento comienza con mayúscula



# Prolog

## Variables

- En Prolog las variables comienzan con letra mayúscula o con el signo “\_” seguidas de algo más.
- El signo “\_”, cuando se encuentra solo, es un caso especial de variable denominada “Variable anónima”.
- Dos apariciones de la variable anónima (\_\_) no hacen referencia a la misma variable.
- Cuando una pregunta tiene variables libres, se dice que tiene tantos grados de libertad como variables libres contenga.
- Mediante el proceso de unificación, las variables libres van quedando ligadas a términos particulares. Cuando esto se produce, la variable queda ligada al término en cuestión por el resto de la ejecución de la rama que se está evaluando.



# Prolog

## Preguntas

- Cuando aparecen variables en la pregunta, PROLOG recorre la base de datos hasta encontrar el primer hecho que coincide con el nombre de la relación y su aridad, y con los argumentos que no son variables. Luego ve si puede unificar la/s variable/s, y cada alternativa de unificación que encuentre se transformará en una respuesta a la pregunta.
- El mecanismo por el cual realiza este proceso se denomina Backtracking automático.



## Ejercicios 3 (15 min.)

- Dada las declaraciones de los ejercicios 1, cree las preguntas dando un grado de libertad a las mismas
  - 1) Que tiene Juan?
  - 2) Quien ama a Raúl?
  - 3) Que es valioso?
  - 4) Que es más valioso que el oro?
  - 5) Quien es el hijo de Juan y María?
  - 6) Cual es el sueldo de Carlos?
  - 7) Quienes ganan \$2300



# Prolog

## Preguntas

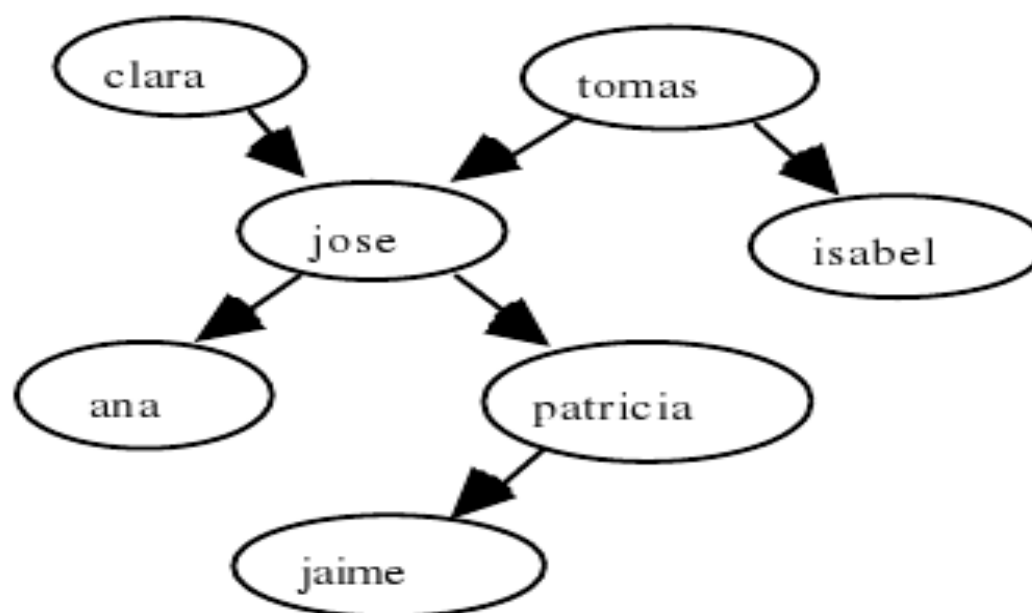
- Se pueden formular preguntas mas complejas a partir de la conjunción de dos o más preguntas simples.
- Ej.: quienes de los empleados de Walmart cobra \$3500.
  - `trabaja_en(X, walmart), sueldo(X, 3500).`
- La coma (“,”) actúa como conjunción de objetivos. Para obtener una respuesta se deberá encontrar respuesta a ambas preguntas.
- La variable que aparece de ésta forma pasa a la segunda parte ya ligada (deja de ser una variable libre).
- El ámbito de las variables alcanza solo a la expresión en donde se encuentren. Las variables de distintas expresiones no están relacionadas.



# Prolog

## Preguntas

- Un ejemplo algo más complejo, dado el siguiente esquema de relaciones de parentesco:



para preguntar si Clara es bisabuela de Jaime, deberíamos escribir algo así:

- `progenitor(clara, X), progenitor(X, Y), progenitor(Y, jaime).`



# Prolog

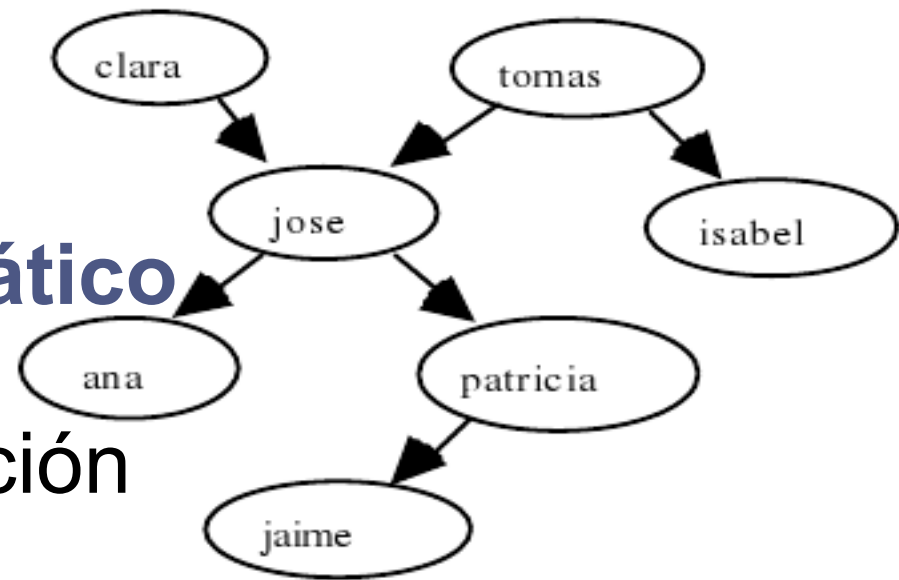
## Backtracking Automático

- Es el proceso por el cual al encontrar más de una alternativa con las cuales unificar, se deja un punto de elección por cada una. Cada punto de elección representa un camino alternativo.
- Cada vez que se evalúa un punto de elección, se elimina el mismo y se comienza a recorrer el camino.
- Cuando se llega al final del camino seleccionado, se devuelve el resultado y se vuelve hasta encontrar otro punto de elección a evaluar.
- Este proceso se repite  $N$  veces hasta que ya no queden puntos de elección a evaluar.
- Mediante esta técnica, Prolog recorre todas las posibles alternativas a evaluar.

# Prolog

## Backtracking Automático

- Esquema de deducción



?-progenitor(clara,X), progenitor(X,Y), progenitor(Y,jaime).

X=jose

progenitor(jose,Y), progenitor(Y,jaime).

Y = ana

Y = patricia

progenitor(jose,ana), **progenitor(ana,jaime)**

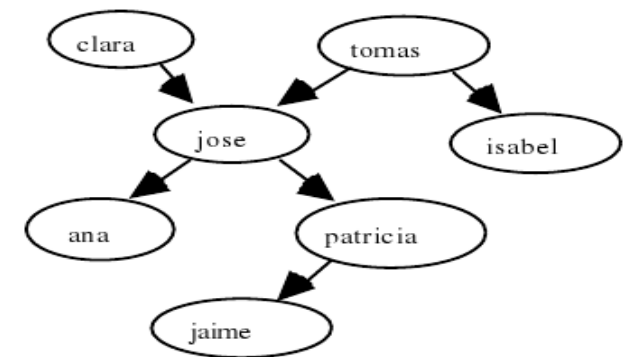
progenitor(jose,patricia), progenitor(patricia,jaime)

No

Si

# Ejercicios 4 (15 min.)

- 1) Escriba el cuerpo de conocimiento en forma de hechos que represente el árbol genealógico de la figura mediante el predicado progenitor/2
- 2) Realice las siguientes preguntas:
  - a) Quienes son los progenitores de Jose?
  - b) Quienes son hijos de Tomás?
  - c) Quien es abuelo de Patricia?
  - d) Quien es bisabuelo de Jaime?
  - e) Con quien ha tenido hijos Clara?





# Prolog

## Reglas

- Las reglas son declaraciones en Prolog usadas para representar conocimiento inferido de otras reglas y/o hechos.
- Las reglas tienen una cabeza y un cuerpo.
- Ej.: podemos inferir la relación abuelo como:
  - `abuelo(X, Y) :- progenitor(X, Z), progenitor(Z, Y).`
- El símbolo “:-” separa la cabeza y el cuerpo de la regla, para que se cumpla la cabeza se debe cumplir el cuerpo.



# Prolog

## Reglas Recursivas

- Como haríamos si quisiéramos declarar una regla “ancestro/2” que determine si una persona es ancestro de otra?
  - `ancestro(X,Y):-progenitor(X,Y).`
  - `ancestro(X,Y):-progenitor(X,X1), progenitor(X1,Y).`
  - `ancestro(X,Y):-progenitor(X,X1), progenitor(X1,X2), progenitor(X2,Y).`
  - ...
- La definición de varias reglas con el mismo nombre y aridad equivale en Prolog a la disyunción lógica (OR lógico)
- Lo que estaríamos declarando arriba es que una persona es ancestro si es progenitora, o es progenitora de la progenitora, o es.....

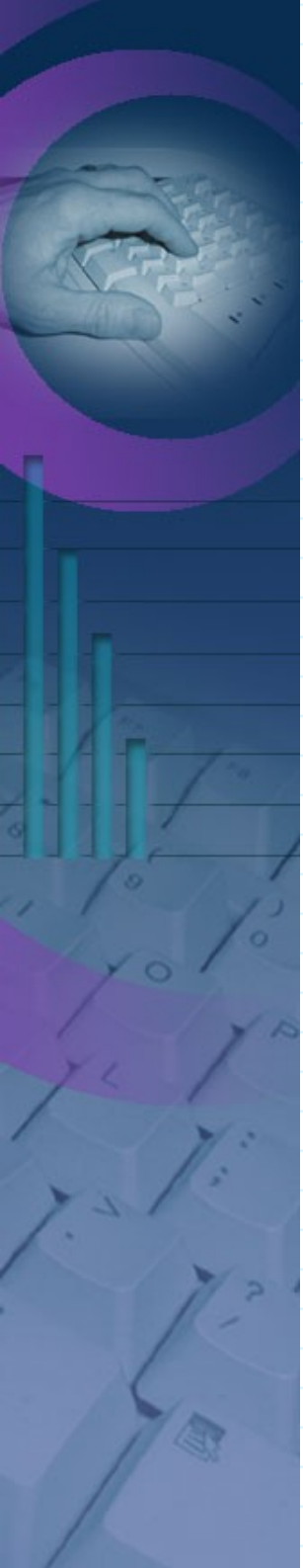




# Prolog

## Reglas Recursivas

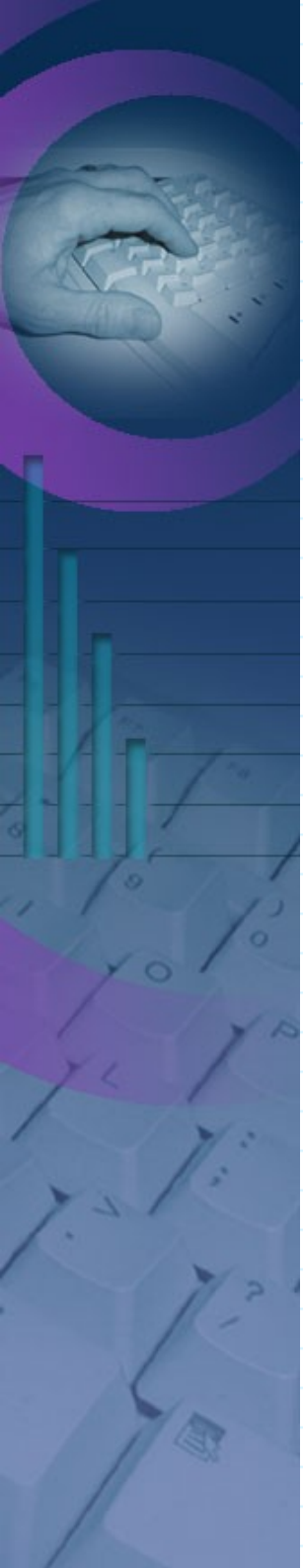
- La definición del predicado ancestro/2 como se implementó está acotada a la cantidad de niveles que definamos.
- El mismo conocimiento podría ser inferido a través de una regla recursiva:
  - `ancestro(X,Y):-progenitor(X,Y).`
  - `ancestro(X,Y):-progenitor(X, Z), ancestro(Z,Y).`



# Prolog

## Sintaxis

- Los Objetos o Términos en Prolog puede ser “objetos simples” o “estructuras”
- Las constantes serán átomos o números. Los átomos comienzan con una letra minúscula. Los números pueden ser reales o enteros, sin una definición explícita de tipos
- Las variables comienzan con mayúscula o subrayado



# Prolog

## Sintaxis

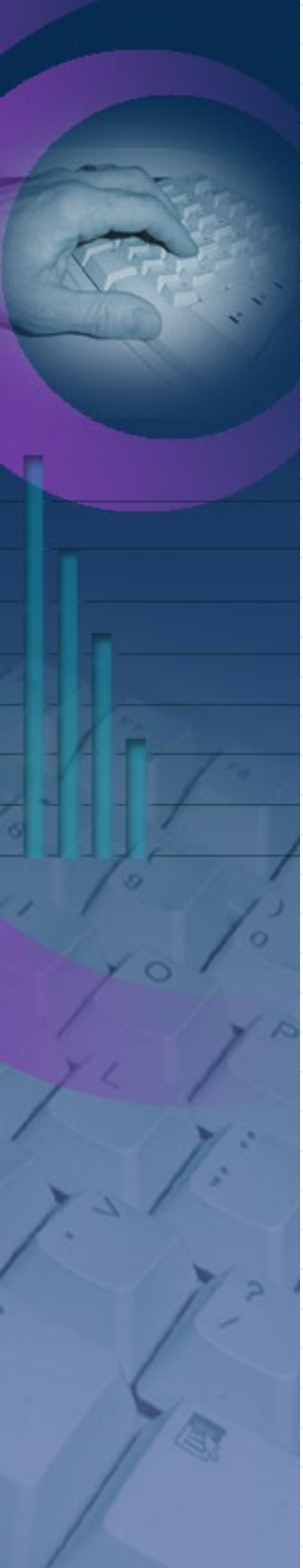
- Las variable anónimas son aquellas cuyos nombres consisten solo en el carácter subrayado (“\_”) y se usan cuando no es importante el nombre de la variable.

- Ej.:

`tiene_hijo(X) :- progenitor(X, Y).`

La variable “Y” no es tenida en cuenta en la definición, por lo que conviene cambiarla por la variable anónima:

`tiene_hijo(X) :- progenitor(X, _).`



# Prolog

## Sintaxis

- La sintaxis para representar estructuras es al misma que para los hechos. Los funtores de las estructuras son los nombres de los hechos, los componentes de las estructuras son los argumentos de los hechos.
- Ej.:
  - punto(X, Y).
  - segmento(punto(X1, Y1), punto(X2, Y2)).
  - triangulo(punto(X1, Y1), punto(X2, Y2), punto(X3, Y3)).



# Ejercicios 5 (25 min.)

1) Tiene éxito la unificación? Cual es el resultado de la instanciación de las variables?

- $\text{triangulo}(\text{punto}(-1,0), P2, P3) = \text{triangulo}(P1, \text{punto}(1,0), \text{punto}(0,Y))$ .

2) Representar cualquier segmento vertical con  $X = 5$  dada la siguiente definición:

- $\text{segmento}(\text{punto}(X1, Y1), \text{punto}(X2, Y2))$ .

3) Si representamos un cuadrilátero con el predicado  $\text{cuadrilatero}(\text{punto}(X1, Y1), \text{punto}(X2, Y2), \text{punto}(X3, Y3), \text{punto}(X4, Y4))$ , donde los argumentos son vértices ordenados en forma horaria, definir la relación:

- $\text{rectangulo}(R)$ .

que será verdadero sólo cuando  $R$  sea un rectángulo con sus aristas en posición horizontal o vertical.



# Prolog

## Significado Declarativo y Procedural

- En un lenguaje declarativo puro el orden en que se realicen las declaraciones no tiene importancia
- Si bien Prolog es un lenguaje declarativo, tiene un componente procedural dado por el orden en que se realizan las unificaciones
- Al momento de escribir un programa, se debe pensar en el aspecto declarativo del lenguaje, pero sin descuidar el aspecto procedural, ya que el mismo puede incidir en el resultado



# Ejercicios 6 (30 min.)

- Construir el árbol de resolución para la pregunta:

?-predecesor(clara,patricia).

teniendo en cuenta las siguientes 4 definiciones de predecesor:

a)  $\text{predecesor}(X,Y):-\text{progenitor}(X,Y).$

$\text{predecesor}(X,Y):-\text{progenitor}(X,Z), \text{predecesor}(Z,Y).$

b)  $\text{predecesor}(X,Y):-\text{progenitor}(X,Z), \text{predecesor}(Z,Y).$

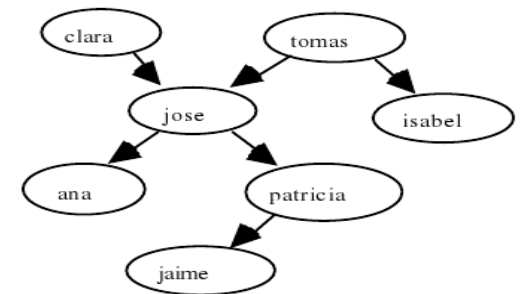
$\text{predecesor}(X,Y):-\text{progenitor}(X,Y).$

c)  $\text{predecesor}(X,Y):-\text{progenitor}(X,Y).$

$\text{predecesor}(X,Y):-\text{predecesor}(Z,Y), \text{progenitor}(X,Z).$

d)  $\text{predecesor}(X,Y):-\text{predecesor}(Z,Y), \text{progenitor}(X,Z).$

$\text{predecesor}(X,Y):-\text{progenitor}(X,Y).$





# Referencias

- SWI-Prolog es una implementación open source del lenguaje de programación Prolog, y es la implementación seleccionada para llevar adelante la cátedra.
- Corre sobre Linux, Windows, y Macintosh.
- Ha estado bajo continuo desarrollo desde el año 1987
  - <http://www.swi-prolog.org/>
- Como entorno de desarrollo utilizaremos JprologEditor, un entorno sencillo y multiplataforma escrito en Java, que funciona como front-end de SWI-Prolog
  - <http://www.trix.homepage.t-online.de/JPrologEditor/>