Prolog

Filosofía del paradigma

- La mayoría de los lenguajes de programación se basan en la teoría lógica de primer orden, aunque también incorporan algunos comportamientos de orden superior, en este sentido, destacan los lenguajes funcionales ya que se basan en el Calculo Lambda, es la única teoría lógica de orden superior.
  - Modelar problemas por medio de la abstracción, utilizando un sistema de lógica formal que permite llegar a una conclusión por medio de hechos y reglas.
  - Aplicación de reglas de la lógica para inferir conclusiones a partir de datos.

¿Qué es la Programación Lógica?

- Paradigma de programación basado en la lógica de primer orden. La programación lógica estudia el uso de la lógica para el planteamiento de problemas y el control sobre las reglas de inferencia para alcanzar la solución automática.
- La programación lógica, junto con la funcional, forma parte de lo que se conoce como Programación Declarativa, es decir la programación consiste en indicar como resolver un problema mediante sentencias.

Características del Paradigma

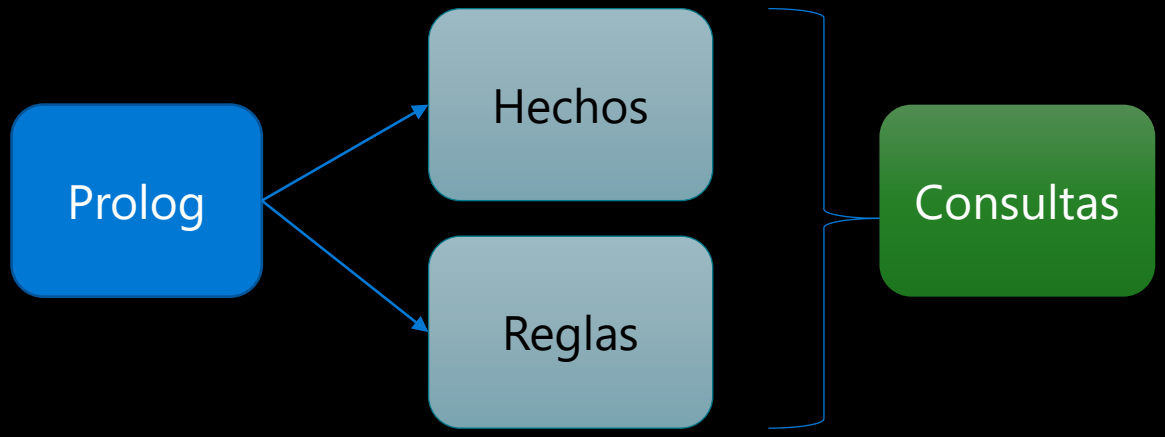
- Basada en la lógica
  - La programación lógica se basa en la lógica matemática, por lo que se utilizan símbolos lógicos como "and", "or", "not" para describir las relaciones lógicas entre los hechos y las reglas.
- Declarativa
  - La programación lógica es un paradigma declarativo, lo que significa que el programador define el resultado deseado, pero no las instrucciones específicas para alcanzarlo. El lenguaje de programación traduce la especificación del problema en una serie de operaciones y algoritmos que se utilizan para obtener el resultado deseado.
- Basada en reglas
  - En la programación lógica, el programador define las reglas que describen las relaciones lógicas entre los hechos y los objetos, y luego le solicita al programa que encuentre una solución lógica a un problema en particular.

Características del Paradigma

- Inferencia
  - La programación lógica se utiliza para la inferencia de datos, que es el proceso de deducir nuevos conocimientos a partir de los datos existentes mediante la aplicación de reglas lógicas.
- Resolución de problemas
  - La programación lógica se utiliza para la resolución de problemas en áreas como la inteligencia artificial, la representación del conocimiento, la inferencia de datos, la planificación y la resolución de problemas en general.
- No determinista
  - La programación lógica es no determinista, lo que significa que el orden de las operaciones y las soluciones que se encuentran no están determinadas de antemano. El programa debe buscar entre todas las posibles soluciones para encontrar la más adecuada.
- Recursiva
  - La programación lógica se basa en la recursividad, lo que significa que una regla puede invocarse a sí misma para resolver un problema de forma repetida hasta que se alcance la solución final.

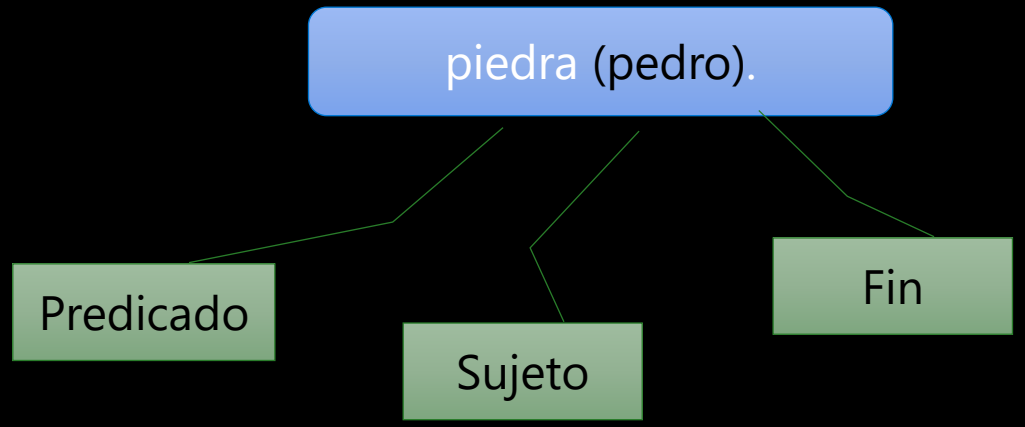
PROgraming in **log**ic

- Prolog es un lenguaje declarativo basado en Reglas y Hechos de lógica, cuya información es retribuido en forma de consultas.
- Originado en Europa a principios de los 70’s por Alain Colmerauer .
- Para realizar los programas, se debe pensar declarativamente.



Introduccion

- HECHOS
  - mario es maestro
  - luisa es un programador
- Notación En Prolog
  - maestro (mario).
  - programador(luisa).



- Para definir un hecho en Prolog, deberá tomar en cuenta que nuestra oración (hecho) debe llevar el formato predicado(sujeto).

Y veamos como se relaciona con la Lógica el Prolog

- La lógica es una ciencia formal que estudia la estructura o formas del pensamiento humano (como proposiciones, conceptos y razonamientos) para establecer leyes y principios válidos para obtener criterios de verdad.

Lógica de orden cero o lógica proposicional

- También llamada ***lógica de enunciados***: toma como elemento básico las frases declarativas simples o proposiciones. Su estructura está dada por:

$$
\begin{aligned}
\textit{Sentencia} &\longrightarrow \textit{Sentencia Atómica} \mid \textit{Sentencia Compleja} \\
\textit{Sentencia Atómica} &\longrightarrow \textbf{Verdadero} \mid \textbf{Falso} \mid \textit{Símbolo Proposicional} \\
\textit{Símbolo Proposicional} &\longrightarrow \textbf{P} \mid \textbf{Q} \mid \textbf{R} \mid \ldots \\
\textit{Sentencia Compleja} &\longrightarrow \neg \textit{Sentencia} \\
&\mid (\textit{Sentencia} \wedge \textit{Sentencia}) \\
&\mid (\textit{Sentencia} \vee \textit{Sentencia}) \\
&\mid (\textit{Sentencia} \Rightarrow \textit{Sentencia}) \\
&\mid (\textit{Sentencia} \Leftrightarrow \textit{Sentencia})
\end{aligned}
$$

Lógica de primer orden o lógica de predicados

- También llamada ***lógica predictiva***: es un sistema deductivo basado en un lenguaje lógico matemático formal. Su estructura esta dada por:

Sentencia	⟶	*Sentencia Atómica* | *(Sentencia Conectiva Sentencia)* | *Cuantificador Variable … Sentencia* | ¬*Sentencia*
Sentencia Atómica	⟶	*Predicado (Término...)* | *Término = Término*
Término	⟶	*Función(Término)* | *Constante* | *Variable*
Conectiva	⟶	^ |v|⇒| ⇔
Cuantificador	⟶	¬*Sentencia*
Variable	⟶	*a* | *x* | *s* | …
Predicado	⟶	*TieneColor* | *EstáLloviendo* | ...
Función	⟶	*Hombre* | *Humano* | *Mujer* | ...

Cláusula de Horn

- Secuencia de literales que contiene a lo sumo uno de sus literales positivos (disyunción de literales). Esto es un ejemplo de una cláusula de Horn, y abajo se indica una fórmula como esta también puede reescribirse de forma equivalente como una implicación:

$$\neg p \lor \neg q \lor \cdots \lor \neg t \lor u$$
$$(p \land q \land \cdots \land t) \rightarrow u$$

antecedente → consecuente "Si es verdad el antecedente, entonces es verdad el consecuente"

$$\neg mujer(A) \lor \neg padre(B, A) \lor hija(A, B)$$
$$(mujer(A) \land padre(B, A)) \rightarrow hija(A, B)$$

Y Prolog...

$$(mujer(A) \land padre(B, A)) \rightarrow hija(A, B)$$

"A es hija de B si A es mujer y B es padre de A"

```
hija(A,B) :- mujer(A), padre(B,A)
```

Estructuras básicas

- Constantes, Variables y Functores.
- Hechos, reglas con cabeza y cola
- Consultas.
- El control se basa en dos conceptos: la unificación y el backtracking.

Estructuras básicas

- SINTAXIS
  - Las variables deben escribirse con Mayúsculas.
    - Con el carácter _, se establecen las variables anónimas.
      - ?-likes(_,juan).
  - Las constantes se escriben con minúsculas.
  - Las afirmaciones se terminan con . (punto).
  - No se pueden dejar espacios entre los nombres de las constantes, para ello utilice el guion bajo (_)
  - Los comentarios empiezan con%.
- OPERADORES
  - Conjunción , (coma)
  - Regla o Condición :-
  - Fin de la condición .

Estructuras básicas

- HECHOS: representan relaciones entre objetos.
  - nombre_relacion(arg1, arg2, …., argN)
  - Ejemplo:
    - tiene(auto, motor) .
    - capital(francia, paris).
    - humano(pablo).
  - Un hecho con un solo argumento, en general es una propiedad.
  - Se pueden introducir hechos con variables como axiomas, por ejemplo:
    - suma(0, X, X).
      - En ellos, las variables se consideran cuantificadas universalmente.
        - Es decir, ∀x suma(0, x, x).
- CONJUNCIÓN , (coma):
  - gusta(maría,X), gusta(Juan,X).

Estructuras básicas

- REGLA O CONDICIÓN :-
  - Las reglas son usadas cuando se quiere señalar que un hecho depende de un grupo de otros hechos. Así el condicional "Si" expresa una regla.
    - "Yo uso una paraguas si está lloviendo"
    - "Juan compra vino si este es menos caro que la cerveza"
  - Así se pueden dar definiciones:
    - X es un pájaro, si:
      - X es un animal, y
      - X tiene alas
    - X es una hermana de Y, si:
      - X es mujer, y
      - X y Y tienen los mismos padres.

Estructuras básicas

- REGLA O CONDICIÓN :-
  - En consiste en una cabeza y un cuerpo, así:
    - CABEZA :- CUERPO
    - :- es el condicional "Si"
    - gusta(juan,X):- gusta(X,cerveza).
    - gusta(juan,X):- gusta(X,cerveza),gusta(X,pizza).
    - "Juan gusta de comer cualquier pizza y acompañarla con cerveza":
      - gusta(juan,X):- pizza(X),gusta(X,cerveza).
  - Las reglas deben terminar **siempre** con un punto (Fin de la condición).

Estructuras básicas

- CONSULTAS: Son preguntas a la base de conocimiento.
  - Ejemplo: ?- es_alumno_curso(pablo, pIII).
    - true
  - Ejemplo: ?- es_alumno_curso(paola, pIII).
    - false
  - Consultas con variables.
    - Ejemplo: ?- es_alumno_curso(pablo, X).
      - X=pIII
    - Ejemplo: ?- es_alumno_curso(paola, X).
      - X= false

Unificación

- Gracias a la unificación, cada objetivo determina un subconjunto de cláusulas susceptibles de ser ejecutadas.
- Cada una de ellas se denomina punto de elección.
- Dos términos T y U son unificables, si y solo si, existe una o varias sustituciones que, aplicados simultáneamente a T y a U, haga a estos dos términos "idénticos".
- Si no existe el conjunto de sustituciones, los dos términos no son unificables.
- Dado los términos:
  - T = p(X, f(a,b), g(Z,F))
  - U = p(t(e), W, g(3,C))
  - X = t(e)W = f(a,b)
  - Z=3 y F=C

Unificación

- padre(jose, maria).
- padre(roberto, romina).
- padre(andres, marcelo).

- ?- padre(roberto, X).
- X=romina

- padre(roberto, X) **unifica con padre(roberto, romina)**
- y la resolución es **X=romina**

Unifiquemos

- f(a)
- f(b)
- g(a)
- g(b)
- h(b)
- k(X):- f(X), g(X), h(X)

- **? k(Y)**

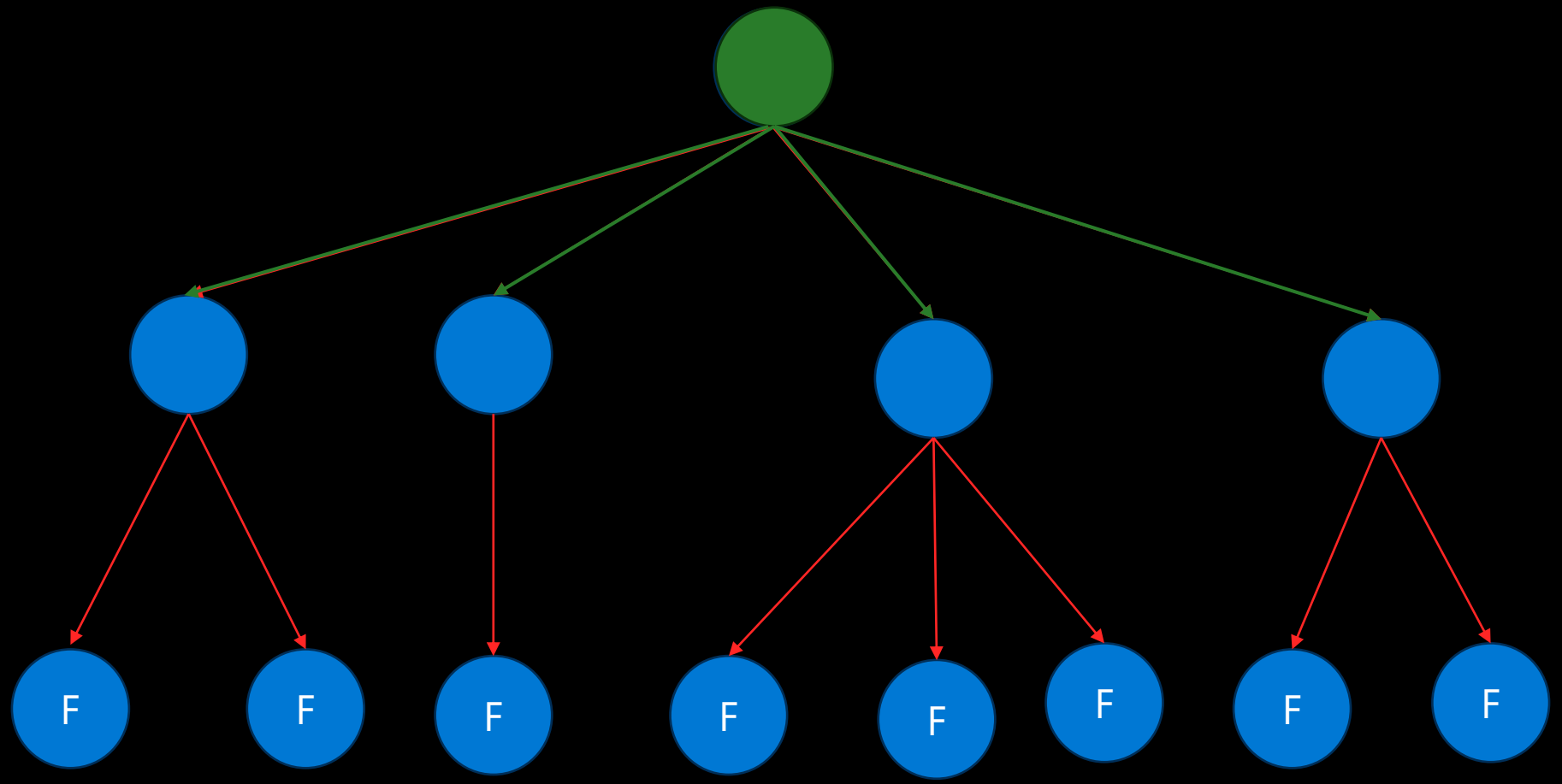
Backtracking

- Prolog selecciona el primer punto de elección y sigue ejecutando el programa hasta determinar si el objetivo es verdadero o falso.
- En caso de ser falso entra en juego el backtracking, que consiste en deshacer todo lo ejecutado situando el programa en el mismo estado en el que estaba justo antes de llegar al punto de elección.
- Entonces se toma el siguiente punto de elección que estaba pendiente y se repite de nuevo el proceso. Todos los objetivos terminan su ejecución bien en éxito ("verdadero"), bien en fracaso ("falso").

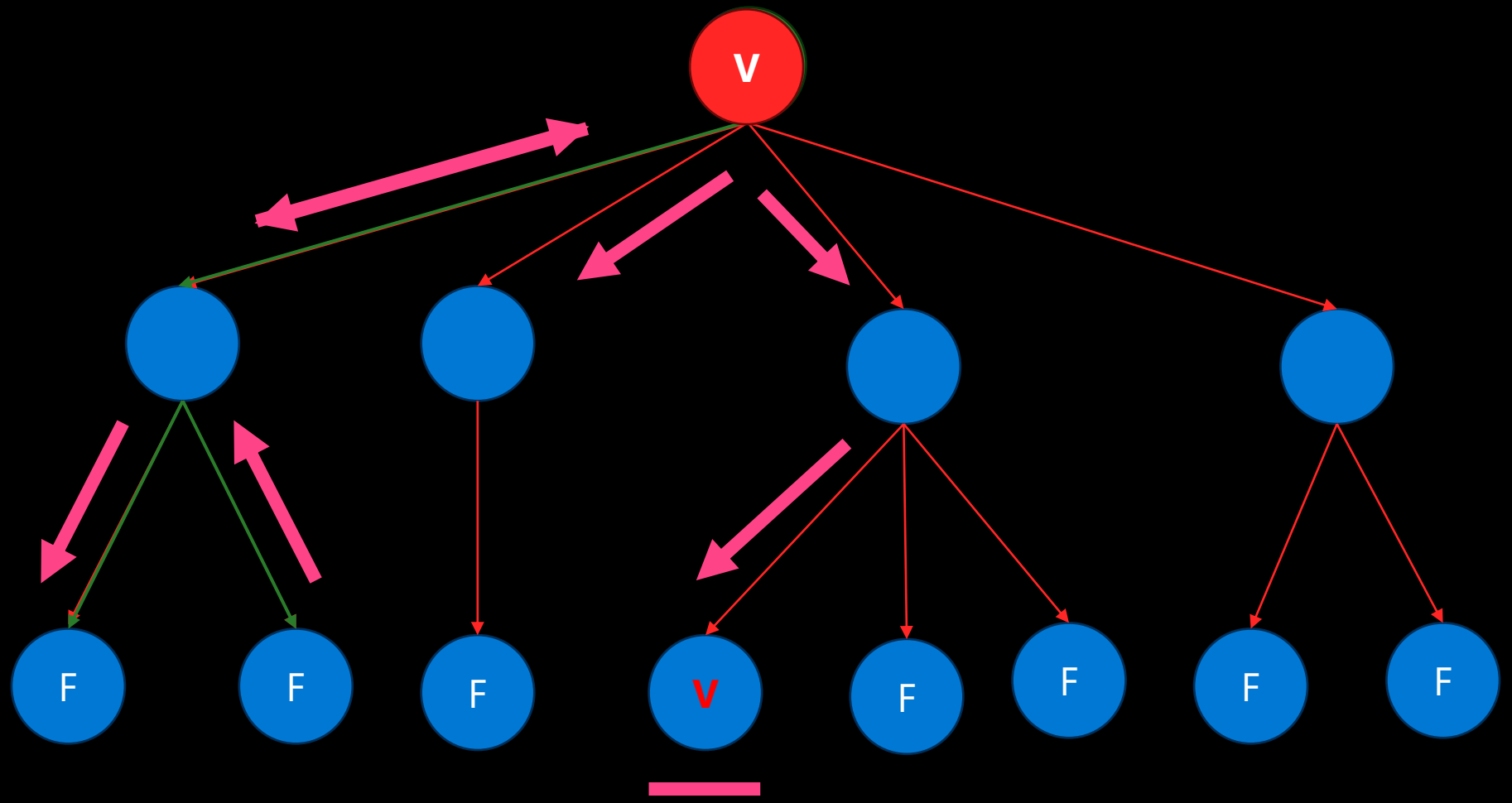
Backtracking

- Los programas en Prolog se componen de cláusulas de Horn que constituyen reglas del tipo "modus ponendo ponens”
  - "Si es verdad el antecedente, entonces es verdad el consecuente".
- No obstante, la forma de escribir las cláusulas de Horn es al contrario de lo habitual.
- Primero se escribe el consecuente y luego el antecedente.
- El antecedente puede ser una conjunción de condiciones que se denomina secuencia de objetivos.
- Cada objetivo se separa con una coma y puede considerarse similar a una instrucción o llamada a procedimiento de los lenguajes imperativos.
- En Prolog no existen instrucciones de control.
- Su ejecución se basa en dos conceptos: la unificación y el backtracking.

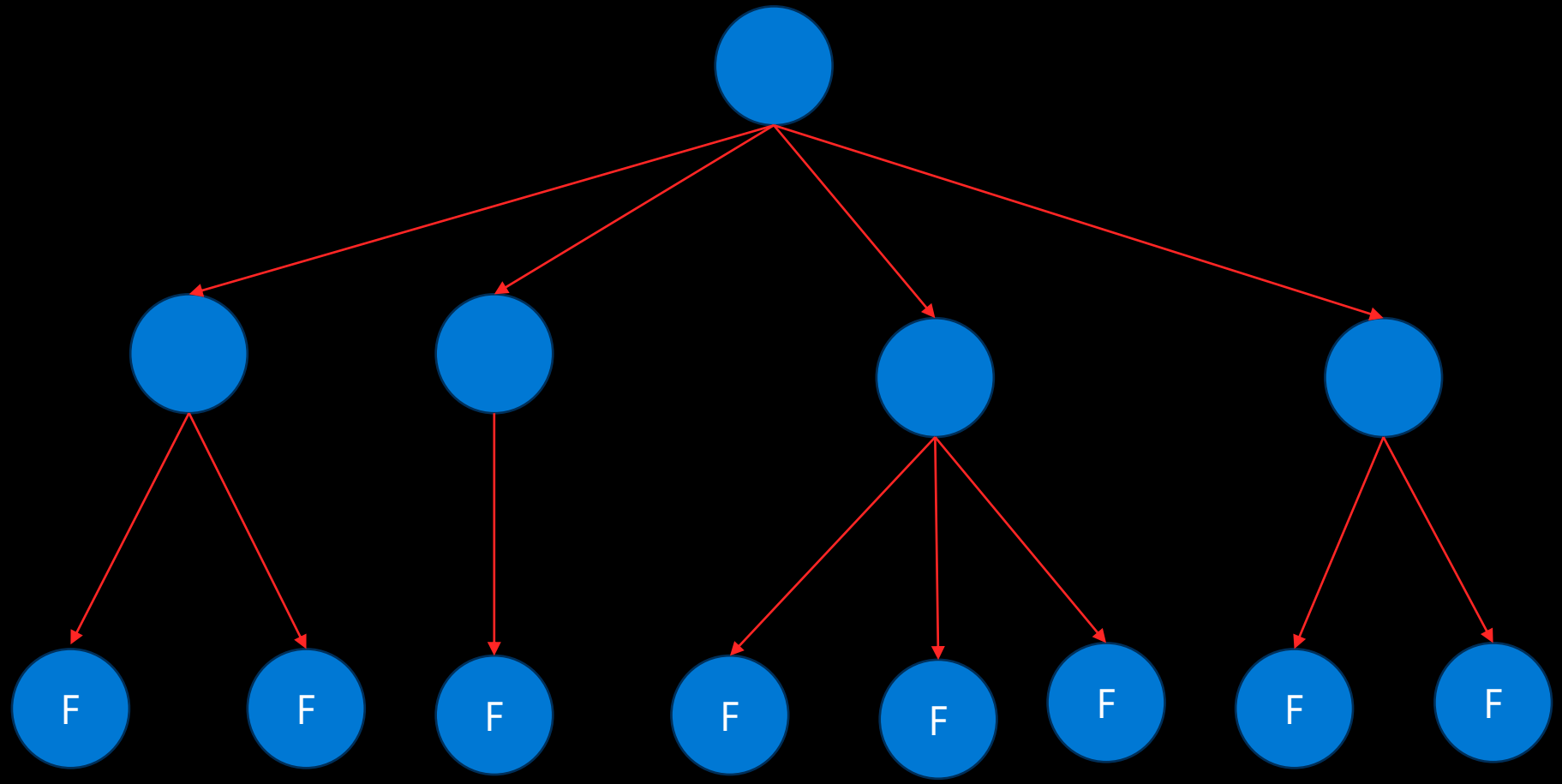
Backtracking en funcionamiento si todo es F



Backtracking en funcionamiento si hay V



Backtracking en funcionamiento



Ejemplo

- Ahora basándonos en lo ya aprendido ejecutaremos nuestro primer ejemplo en el editor SWI-Prolog on-line (https://swish.swi-prolog.org/).
- Para ello teclee lo siguiente:
  - maestro(mario).
  - programador(luisa).
  - piedra(pedro).
- Teclee las consultas, ejemplo:
  - ?- maestro(mario).
- Interprete Prolog y área de consulta
  - ¿mario es maestro?

Comandos básicos

- ? halt.
  - Sale del entorno Prolog.
- ? listing.
  - Muestra todo los predicados de la base de conocimiento.
- ? trace, CONSULTA
  - Activamos el modo traza o debug.

Ejemplo con conjuncion

- Ahora utilizaremos variables y la conjunción para realizar consultas.
- Para ello teclee lo siguiente:
  - dog(fido).
  - dog(rover).
  - dog(henry).
  - cat(felix).
  - cat(michael).
  - cat(jane).
  - animal(X):-dog(X).

? dog(fido).
? dog(jane).
? animal(fido).
? dog(X).
? animal(felix).
? listing(dog).
? cat(X),dog(Y).

Mas ejemplos

- wizard(ron).
- hasWand(ron).
- hasBroom(ron).
- quidditchPlayer(harry).
- wizard(X) :- hasBroom(X),hasWand(X).
- hasBroom(X):- quidditchPlayer(X).

- Como responde a:
  - wizard(ron).
  - witch(ron).
  - wizard(hermione).
  - witch(hermione).
  - wizard(harry).
- A los que responde FALSE, ¿Cómo los “arreglo”?

Seguimos

- hermano(pablo, daniela).
- hermano(pablo, lucia).
- hermano(juan, lucas).
- hermano(lucas, miguel).
- hermano(pablo, martin).
- hermano(miguel, edith).

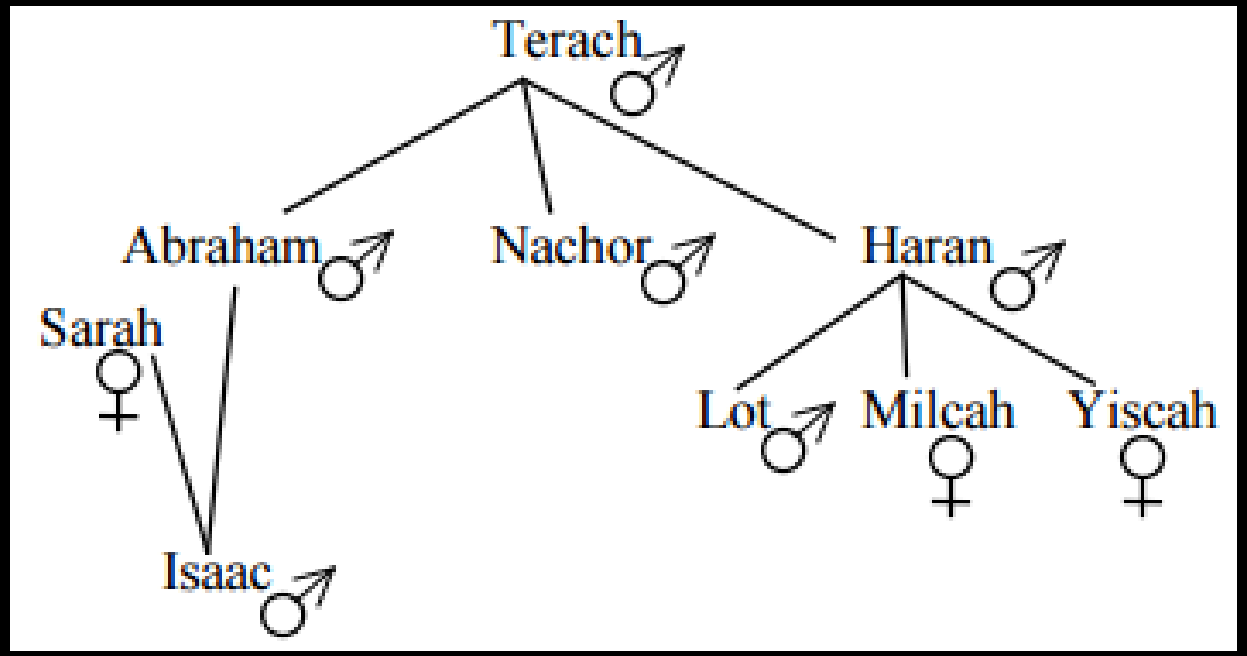
- Consultas:
  - ?- hermano(pablo, martin).
  - ?- hermano(pablo, X).
  - ?- hermano(X, Y).
  - ?- hermano(pablo, daniela), hermano(pablo,lucia).
  - ¿Lucas y Edith tienen el mismo hermano?
    - ?- hermano(lucas, X), hermano(X,edith).

Seguimos

- Hechos:
  - Las aves vuelan.
  - Los pingüinos no vuelan.
  - Coti es un ave.
  - Terry es un perro.
  - Pity es un ave.
- Reglas:
  - Una mascota vuela si es un ave y no es un pingüino.
- Consultas:
  - ¿Coti vuela?
  - ¿qué mascotas vuelan?

Y ahora…

- ¿¿¿Como hacemos un árbol genealógico, y como hacemos las consultas????



La familia…

- Una regla que define la relación “ser hijo” a partir de las relaciones dadas podría ser:
  - es_hijo(X,Y) :- es_padre(Y,X), es_hombre(X).
  - que se leería de la forma: “para cualesquiera X e Y, X es hijo de Y si Y es padre de X y X es hombre"
  - Ya que se corresponde con la fórmula lógica: ∀x ∀y ((es_padre(y,x) ∧ es_hombre(x)) → es_hijo(x,y)).
- De igual forma se definirían otras relaciones mediante reglas:
  - es_hija(X,Y) :- es_padre(Y,X), es_mujer(X).
  - es_abuelo(X,Z) :- ….
    es_padre(X,U), es_padre(U,Z).
  - O sea ∀x ∀z (∃u (es_padre(x,u) ∧ es_padre(u,z)) → es_abuelo(x,z)).

Sin usar el entorno ¿Qué hace?

- % Hechos:
  - es_español(manolo).
  - es_italiano(marco).
  - es_colombiano(marcelo).

- % Reglas:
  - es_europeo(A) :- es_español(A).
  - es_europeo(A) :- es_italiano(A).
  - es_americano(A) :- es_colombiano(A).
  - es_terricola(A) :- es_europeo(A).
  - es_terricola(A) :- es_americano(A).
  - son_del_mismo_continente(A,B) :- es_europeo(A), es_europeo(B).
  - son_del_mismo_continente(A,B):-es_americano(A), es_americano(B).

A programar

- Implementen una regla que diga “cuando llueve, me resfrío”. Fíjense que para que el intérprete conteste true a la pregunta “yo me resfrío”, el antecedente de la implicación tiene que ser cierto.

Ejercitemos

- Representar los siguientes enunciados como hechos:
  - Juan es el hermano de Julia.
  - José es valiente.
  - Pedro esta leyendo Hamlet.
  - Shakespeare es el autor de Hamlet.
  - Mozart compuso La Flauta Mágica.
  - Pablo visitó a Norma desde Holanda.
  - Mirta es alta.

- La relación es_mas_alto(X,Y), expresa que X es más alto que Y. La Base de hechos es la siguiente:
  - es_mas_alto(john,luke).
  - es_mas_alto(john,marie).
  - es_mas_alto(martine,marie).
  - es_mas_alto(catherine,john).
  - es_mas_alto(mark,john).
  - es_mas_alto(mark,martine).
- ¿Qué responde Prolog a las siguientes consultas? Responder haciendo el análisis mentalmente y luego comprobarlo codificando el ejercicio.
  - es_mas_alto(X,john).
  - es_mas_alto(X,Y).
  - es_mas_alto(mark,X).
  - es_mas_alto(catherine,Y), es_mas_alto(Y,Z).

Seguimos

- La siguiente base de hechos contiene información sobre los vuelos de una compañía aérea.
  - vuelo(numero,origen,destino,hora_salida,hora_arribo,plazas).
  - vuelo(1,toronto,montreal,1200,1300,42).
  - vuelo(2,paris,toronto,1400,1525,245).
  - vuelo(3,toronto,otawa,1345,1500,234).
  - vuelo(4,vancuver,roma,0920,1100,51).
  - vuelo(5,montreal,mexico,1030,1250,58).
- El último hecho significa que existe un vuelo, el numero 5 de Montreal a México que parte a las 10:30 hs y arriba a las 12:50 hs. y dispone de 58 plazas.

- Escribir consultas para:
  - Listar los vuelos que salen de Paris.
  - Listar los vuelos que arriban a Toronto.
  - Listar los vuelos que parten de Toronto después de las 12.30 hs.
  - Listar los vuelos que tienen más de 100 plazas.
  - Listar los vuelos de duración mayor a dos horas.

Seguimos

- Usando las relaciones (hechos):
  - conoce(X,Y).
  - amigo(X,Y).
- Realizar consultas para:
  - Ver a todos los amigos de un amigo.
  - Ver a todos los conocidos de un amigo.
- Definir una base de hechos con la relación hijo(X,Y), que expresa que X es hijo de Y; y la propiedad que indique el sexo de los objetos utilizados en los hechos hijo. Definir reglas para las siguientes relaciones:
  - hermana(X,Y).
  - hermano(X,Y).
  - bisabuelo(X,Y).
  - bisabuelo(X,Y).

Seguimos

- Para los siguientes tipos de hechos:
  - curso(Nombre_profesor,Numero_curso).
  - estudiante(Nombre_estudiante,Numero_de_curso).
- Definir:
  - alumno_de(Nombre_estudiante, Nombre_profesor).

- Para los siguientes tipos de hechos:
  - transporte(Ciudad_origen, Ciudad_destino, Costo).
  - alojamientoXdia(Nombre_ciudad, Costo).
- Definir:
  - costo_viaje(Ciudad_origen, Ciudad_destino, Cantidad_dias, Costo_total).