

Programación Lógica

RODRÍGUEZ ARIAS, ALEJANDRO

ALEJANDRO.RODRIGUEZ.ARIAS@UDC.ES

BOUZAS QUIROGA, JACOBO

JACOBO.BOUZAS.QUIROGA@UDC.ES



PROLOG

Cláusulas de Horn

$$(p \wedge q \wedge r) \rightarrow s$$

% base de hechos

% mujeres

mujer(sara).

mujer(selena).

mujer(silvia).

mujer(soraya).

mujer(susana).

% varones

hombre(jaime).

hombre(jeronimo).

hombre(jimeno).

hombre(jorge).

hombre(julian).

```
% parentescos explicitos: padres(hijo , progenitor1 , progenitor2)  
padres(susana , jaime , sara).  
padres(julian , jeronimo , selenia).  
padres(silvia , julian , susana).  
padres(jimeno , jorge , soraya).
```

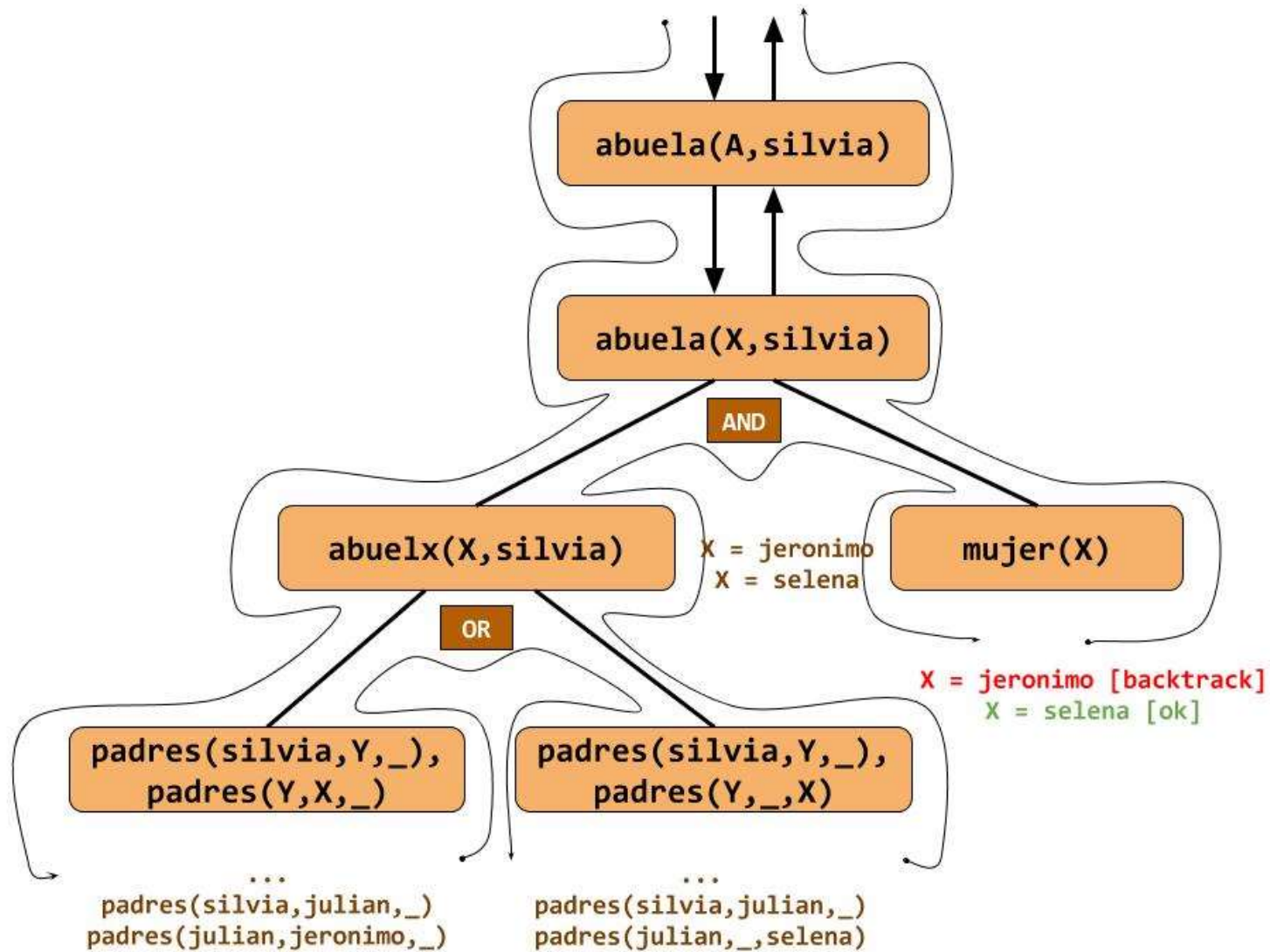
```
% parentescos derivados (logicamente deductibles)  
hija(X,Y) :- mujer(X) , padres(X,Y,_).  
hija(X,Y) :- mujer(X) , padres(X,_,Y).  
hijo(X,Y) :- hombre(X) , padres(X,Y,_).  
hijo(X,Y) :- hombre(X) , padres(X,_,Y).  
abuelx(X,Z) :- padres(Z,Y,_), padres(Y,X,_).  
abuelx(X,Z) :- padres(Z,_,Y), padres(Y,X,_).  
abuelx(X,Z) :- padres(Z,Y,_), padres(Y,_,X).  
abuelx(X,Z) :- padres(Z,_,Y), padres(Y,_,X).  
abuelo(X,Y) :- hombre(X) , abuelx(X,Y).  
abuela(X,Y) :- mujer(X) , abuelx(X,Y).
```

```
% query: quien es abuela de silvia?  
?- abuela(Abuela, silvia).  
Abuela = sara ;  
Abuela = selenia ;  
false.
```

```
% query: es jimeno hijo de susana?  
?- hijo(jimeno, susana).  
false.
```

```
% query: de quien es hijo jimeno?  
?- hijo(jimeno, Progenitor).  
Progenitor = jorge ;  
Progenitor = soraya ;  
false.
```

```
query: que hijos varones tiene selenia?  
?- hijo(Hijo, selenia).  
Hijo = julian.
```



MAUDE SYSTEM

*** declaracion de tipos

sort Nat .

sort Zero .

*** se puede utilizar la palabra sorts para hacer

*** varias declaraciones en la misma linea.

sorts Nat Zero .

*** para declarar subtipos existe la palabra clave

*** subsort con la siguiente sintaxis.

subsort Zero < Nat .

subsorts NzNat Zero < Nat .

*** Nat es un kind de Maude

*** Nat NzNat y Zero forman parte de

[Nat]

```
*** declaracion de operadores
op zero : -> Zero .
op length : List -> Nat .
op _+_: Nat Nat -> Nat
```

```
*** declara una variable con identificador N del sort Nat
N : Nat
*** declara una variable con identificador N del kind Nat
N : [Nat]
```

```
*** el scope alcanza todo el modulo
var N : Nat .
var X : [Nat] .
vars M N : nat .
```

```
*** importar un modulo en modo protecting impide definir
*** nuevos terminos irreducibles a los tipos definidos en
*** el modulo importado y agregar reglas de reduccion que
*** hacen que terminos que eran distintos ahora no lo sean
fmod FACTORIAL is
protecting NAT .
```

```
op _! : Nat -> NzNat .
var N : Nat .
```

```
*** eq permite definir reglas de reescritura
eq 0 ! = 1 .|
eq (s N) ! = (s N) * N ! .
endfm
```

Dominios de Aplicación

- ❑ Procesamiento lenguaje natural

- ❑ Clarissa (NASA)

- ❑ IBM Watson

- ❑ Amazon Lex y Alexa

- ...

- ❑ Demostración de teoremas

- ❑ Verificación formal

- ❑ Programación concurrente

- ❑ Interprete Erlang

- ...

- ❑ Sistemas Expertos

- ❑ DealBuilder

- ❑ Arezzo

- ❑ InFlow

- ...