

# Introduzione alla FRAME LOGIC

# Frame Logic: concetti di base

Schema (/Frame/Classe)

$x::y$   $y$  è sottoclasse  $x$ ;

$a:y$   $a$  è istanza di  $y$ ;

$x[s \Rightarrow y]$   $s$  è uno slot di  $x$  di tipo  $y$ ;

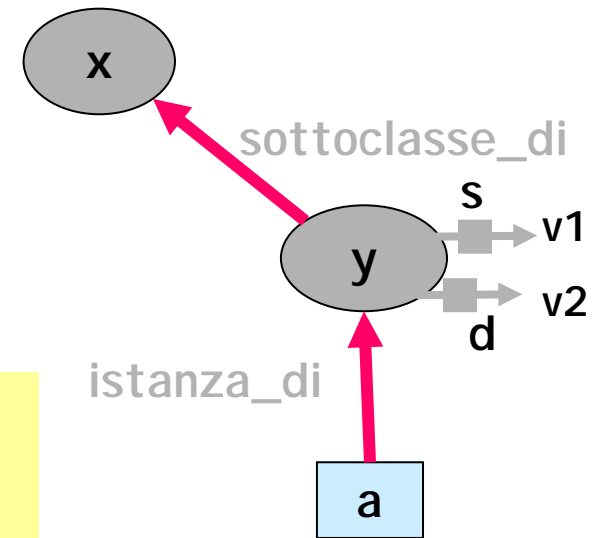
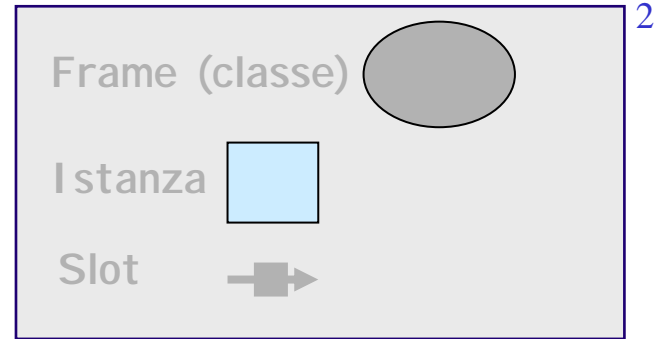
$y[s \Rightarrow\Rightarrow tx]$   $s$  è uno slot di  $y$  a valori multipli di tipo  $tx$

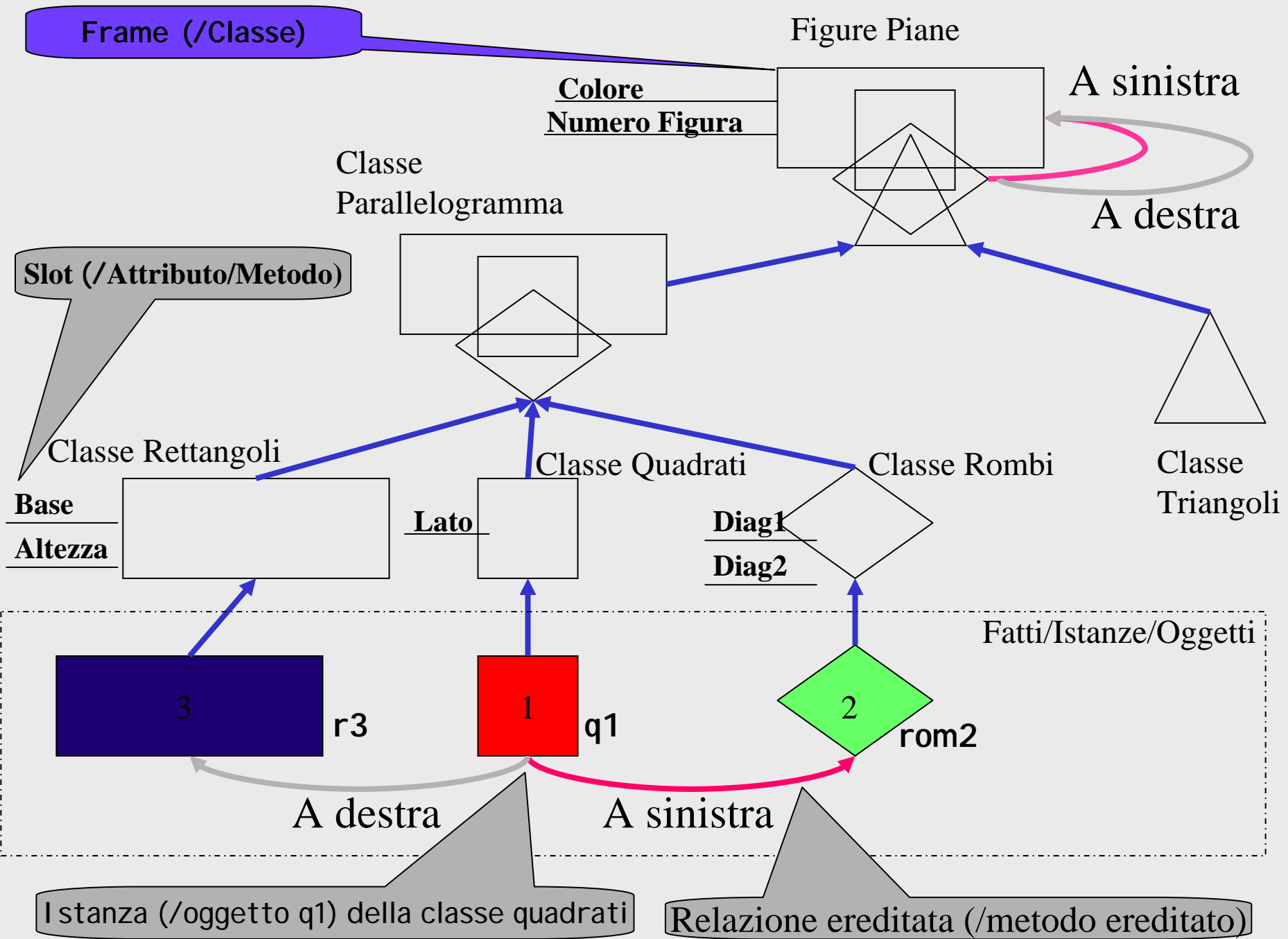
Oggetti

$ix: y[s \rightarrow vx]$   $vx$  è il valore dello slot  $s$  della classe  $y$ ;

$ix: y[s \Rightarrow\Rightarrow\{v1, v2, ..\}]$   $v1, v2, ..$  sono i valori dello slot  $s$  della classe  $y$ .

$ix$  corrisponde al logical object identifier (oid)





Frame (/Classe)

Figure Plane

Colore  
Numero Figura

Classe  
Parallelogramma

A sinistra

A destra

Slot (/Attributo/Metodo)

Classe Rettangoli

Classe Quadrati

Classe Rombi

Classe  
Triangoli

Base  
Altezza

Lato

Diag1  
Diag2

Fatti/Istanze/Oggetti

3

r3

1

q1

2

rom2

A destra

A sinistra

I stanza (/oggetto q1) della classe quadrati

Relazione ereditata (/metodo ereditato)

Un programma in F-Logic è  
composto da:  
uno schema;  
un database di fatti (oggetti);  
regole di deduzione; queries

```
% Regole deduttive
Y[a_sinistra->X]:-X[a_destra->Y]. (inverse slot)
X[a_destra->Y]:-Y[a_sinistra->X].
X[a_sinistra->Z]:-X[a_sinistra->Y],Y[a_sinistra->Z].
X[a_destra->Z]:-X[a_destra->Y],Y[a_destra->Z].
```

```
% Schema (classi/frame)
figure_piane::parallelogrammi.
parallelogrammi::quadrato.
parallelogrammi::rettangolo.
parallelogrammi::rombo.
figure_piane::triangoli.

figure_piane[numero_figura=>number,
              colore=>string,
              a_sinistra=>figure_piane,
              a_destra=>figure_piane].
quadrato[lato=>number].

rombo[diagonale1=>number,
      diagonale2=>number].
rettangolo[base=>number,
            altezza=>number].
```

```
% Database di fatti (I stanze/Oggetti)
q1:quadrato[lato->3,
            colore->rosso,
            numero_figura->1,
            a_sinistra->r3].
r3:rettangolo[altezza->3,
              base->7,
              colore->blu,
              numero_figura->3].
rom2:rombo[colore->verde,
           numero_figura->2,
           diagonale1->3,
           diagonale2->5,
           a_destra->q1].
```

```
% Queries
?- I:rombo[colore->Y].
?- L:rombo[a_sinistra->X].
?- I:C[A->V].
```

# Queries in Frame Logic (Flora2 System) (I)

```
flora2 ?- Ox[colore->V].
```

```
Ox = q1  
V = rosso
```

```
Ox = r3  
V = blu
```

```
Ox = rom2  
V = verde
```

```
3 solution(s) in 0.000000 seconds  
Yes.
```

```
flora2 ?- Ox:quadrato, Ox[colore->V].
```

```
Ox = q1  
V = rosso
```

```
1 solution(s) in 0.000000 seconds  
Yes.
```

Query che può essere scritta anche come:

```
Ox:quadrato[colore->V].
```

## Queries in Frame Logic (Flora2 System) (II)

flora2 ?- Ox:rettangolo[a\_destra->Oy].

No.

Non esiste alcun fatto/oggetto nel database che soddisfi la query - e inoltre non può essere soddisfatta applicando regole di deduzione

flora2 ?- Ox:quadrato[a\_destra->Oy].

Ox = q1

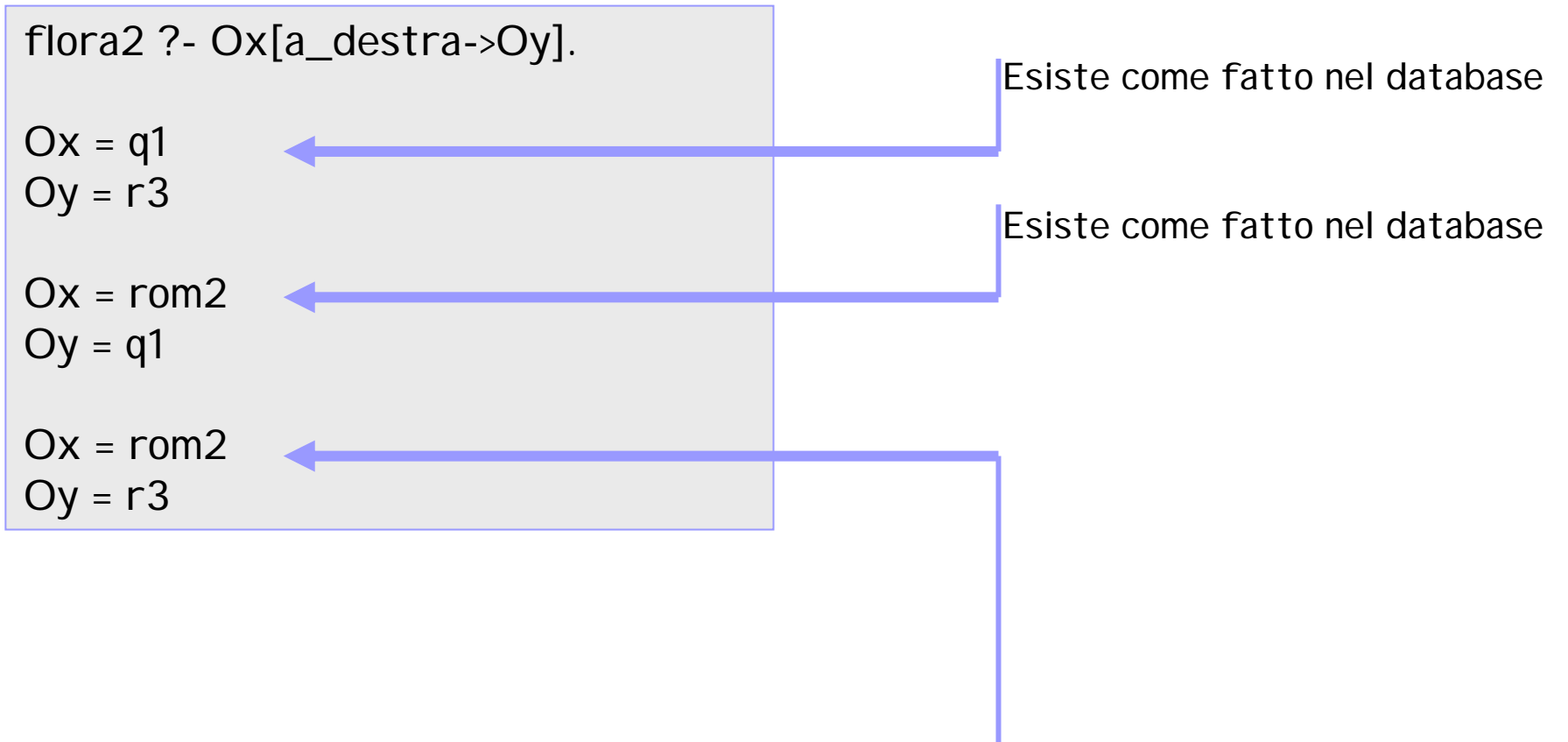
OY = r3

1 solution(s) in 0.000000 seconds

Yes.

Esiste come fatto nel database: oggetto q1.

# Queries in Frame Logic (Flora2 System) (III)



Il fatto non esiste nel database viene dedotto dalla regola 4:  
 $\text{rom2}[a\_destra \rightarrow r3] :- \text{rom2}[a\_destra \rightarrow q1], q1[a\_destra \rightarrow r3].$   
(rom2[a-destra ->q1] e q1[a\_destra->r3] sono fatti presenti nel database)

# Queries in Frame Logic (Flora2 System) (IV)

flora2 ?- Ox[a\_sinistra->Oy].

Ox = q1  
Oy = rom2

Ox = r3  
Oy = q1

Ox = r3  
Oy = rom2

Non esiste come fatto nel database. - viene dedotto attraverso i seguenti passi:

1- La classe quadrato eredita lo slot a\_sinistra dalla classe figure\_piane;

2- La prima regola viene istanziata:

q1[a\_sinistra->rom2]:-rom2[a\_destra->q1]

Non esiste come fatto nel database. - viene dedotto attraverso i seguenti passi:

1- La classe rettangolo eredita lo slot a\_sinistra dalla classe figure\_piane;

2- La prima regola viene istanziata:

r3[a\_sinistra->q1]:-q1[a\_destra->r3]

Non esiste come fatto nel database. - viene dedotto attraverso i seguenti passi:

1- La classe rettangolo eredita lo slot a\_sinistra dalla classe figure\_piane.

2- La terza regola di deduzione viene istanziata:

r3[a\_sinistra->rom2]:-r3[a\_sinistra->q1], q1[a\_sinistra->rom2].

con r3[a\_sinistra->q1] e q1[a\_sinistra->rom2] dedotti come nei casi precedenti

## Schema

```

strada::strutture_viabili.
via::strutture_viabili.
strutture_viabili[nome=>string, incontra=>>struttura_viabili].
strada[sinonimo*=>via].
via[sinonimo*=>strada].
    
```

## Regole di deduzione

```

X[incontra->Y]:-Y[incontra->X].
X[sinonimo->Y]:-Y[sinonimo->X].
    
```

## Fatti

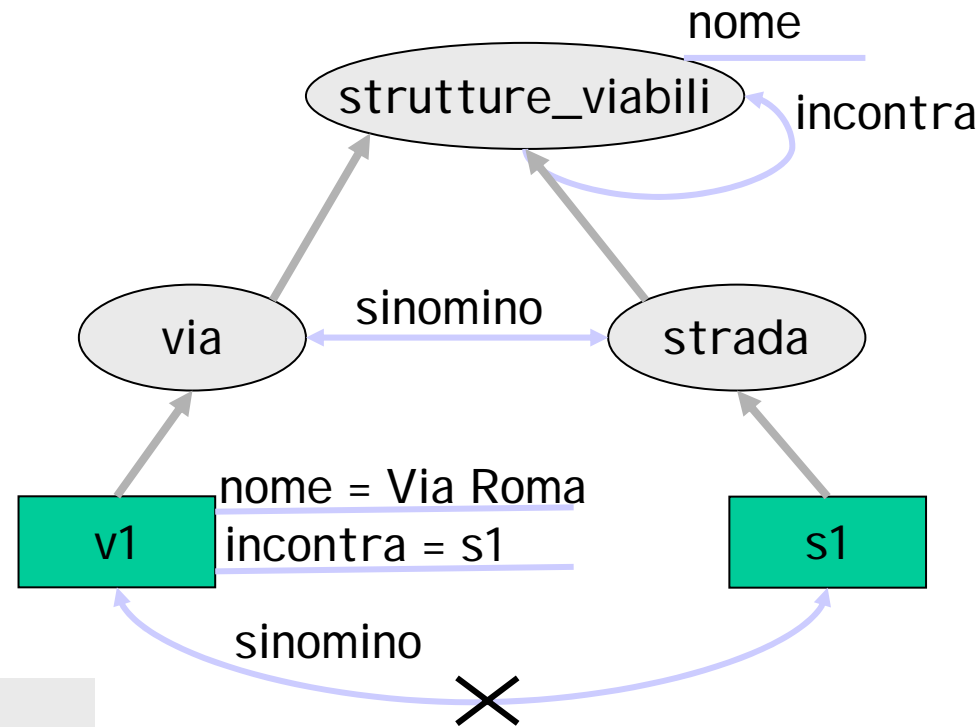
```

s1:strada.
s1[nome -> 'Federico Lombardi' ].
v1:via.
v1[nome->'Via Roma', incontra->s1].
    
```

## Queries

```

flora2 ?- s1[incontra-> X], X[nome->N].
X = v1
N = Via Roma
flora2 ?- s1[sinonimo-> X], X[nome->N].
no.
    
```



## Flora2 e Prolog (I)

Attivando programmi Prolog da altri moduli.

```
flora2 ?- member(ay,[ax,ay,az])@prolog basics.
```

Yes.

```
flora2 ?- append([ax],[ay,az],C)@prolog basics.
```

```
C = [ax,ay,az]
```

Yes.

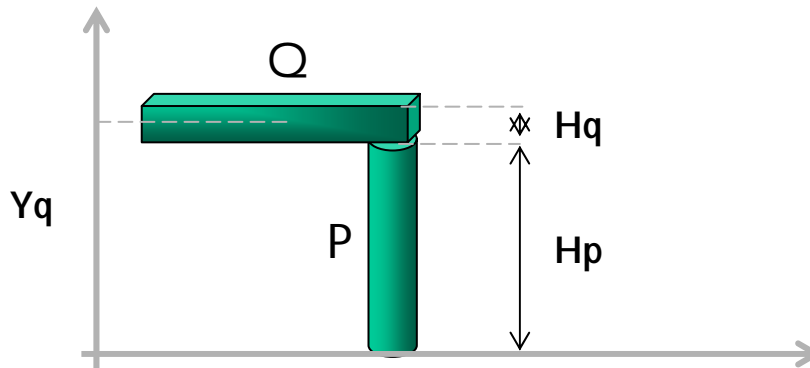
## Flora2 e Prolog (II)

Si possono definire all'interno del contesto di Flora regole del tipo:

`stesso_colore(X,Y):-X:Z1[colore->Cx],Y:Z2[colore->Cx],X\=Y.`

`ordinata(Q,Yq) :-`

`_R1:sotto[obj1->P,obj2->Q], _R2:incontra[obj1->P,obj2->Q],`  
`incontra[obj1->P,obj2->suolo],`  
`altezza(Q,Hq), altezza(P, Hp), Yq is Hp + Hq/2.`



L'inferenza stabilisce che  $Yq$  è il valore della ordinata  $Yq$  del baricentro di un oggetto  $Q$  se: l'oggetto  $P$  sta *sotto* l'oggetto  $Q$ ,  $P$  *incontra*  $Q$ ,  $P$  è posto sul suolo, si conoscono l'altezza di  $Q$  e di  $P$ .

```
% Fatti
```

```
franco[conosce_linguaggi->>{xml,prolog,f_logic}] .  
giovanni[conosce_linguaggi->>{lisp}].
```

```
giovanni::-franco.
```

```
:- equality flogic. % per default :- equality none.
```

```
flora2 ?- giovanni[conosce_linguaggi->> X].
```

```
X = f_logic
```

```
X = lisp
```

```
X = prolog
```

```
X = xml
```

## Specificità di FLORA 2

## Eguaglianza esplicita

“:=:”

(II)

% Fatti

```
giovanni[similar->franco].
franco[conosce_linguaggi->>{xml,prolog,f_logic}] .
giovanni[conosce_linguaggi->>{lisp}].
```

```
:- equality flogic. % per default :- equality none.
```

% Regola

```
X:=:Y:-X[similar->Y].
```

```
flora2 ?- giovanni[conosce_linguaggi->> X].
```

```
X = f_logic
```

```
X = lisp
```

```
X = prolog
```

```
X = xml
```

```
4 solution(s) in 4.214844 seconds
```

```
Yes.
```

```
flora2 ?- franco[conosce_linguaggi->> X].
```

```
X = f_logic
```

```
X = lisp
```

```
X = prolog
```

```
X = xml
```

```
4 solution(s) in 0.000000 seconds
```

```
Yes.
```

## Specificità di FLORA 2

## Eguaglianza esplicita "::-"

(III)

Un esempio più complesso – similarità dedotta.

```
% Fatti
```

```
franco[conosce_linguaggi->>{xml,prolog,f_logic}] .
giovanni[conosce_linguaggi->>{lisp}].
```

```
:- equality flogic. % per default :- equality none.
```

```
% Regola
```

```
X:::Y:-X[similar->Y].
```

```
X[similar->Y]:-X[professore->Z],Y[professore->Z],
                X[laurea->Lx],Y[laurea->Lx],
                X[gruppo_lavoro->Gx],Y[gruppo_lavoro->Gx].
```