

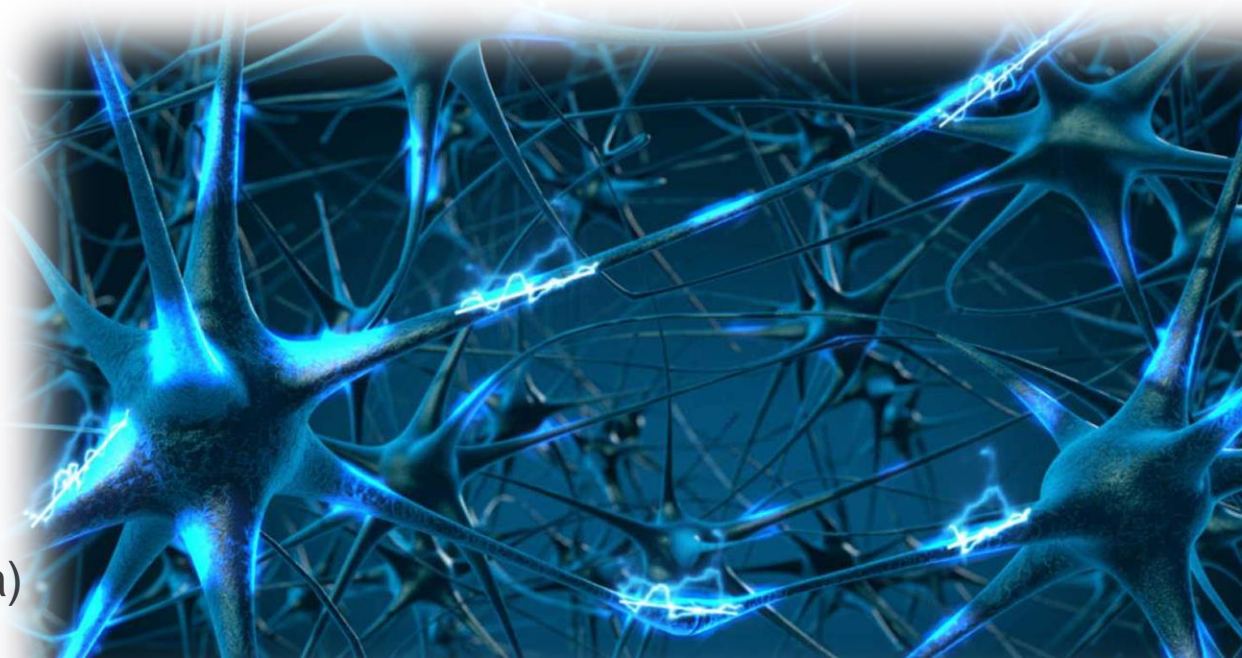
# *INTELLIGENZA ARTIFICIALE*

## *KR: CALCOLO DEI PREDICATI (\*)*

Corsi di Laurea in Informatica, Ing. Gestionale, Ing. Informatica,  
Ing. di Internet  
(a.a. 2022-2023)

Roberto Basili

(\*) alcune *slides* sono di  
Maria Simi (Univ. Pisa)



# Overview

- Logica dei Predicati (FOL): motivazioni
  - Concettualizzazione e Logica
- Sintassi e Semantica
  - Termini, Connettivi e Quantificatori
  - Interpretazione di una formula logica
  - Database Semantics
- Inferenza Logica nel Calcolo dei Predicati
  - Conseguenza Logica e Dimostrazione
- Formule ground e Interpretazione di Herbrand
- Risoluzione e Completezza
  - Unificazione
  - Algoritmi di Risoluzione
  - Introduzione alla Programmazione Logica

# Il calcolo dei predicati per R.C.

- Nella logica dei predicati abbiamo assunzioni ontologiche più ricche: gli *oggetti*, le *proprietà* e le *relazioni*
- Si inizia con una *concettualizzazione*: si tratta di decidere quali sono le cose di cui si vuole parlare
  - *Gli oggetti*: un *libro*, un *evento*, una *persona*, un istante di tempo, un *insieme*, una *funzione*, un *unicorno* ...
    - Gli oggetti possono essere identificati con *simboli* o relativamente ad altri oggetti, mediante *funzioni*: “*la madre di Pietro*”
    - L’insieme degli oggetti rilevanti costituiscono il *dominio del discorso*. Il dominio potrebbe essere infinito.
  - Le *proprietà*: “*la madre di Pietro è simpatica*”
  - Le *relazioni* tra gli oggetti: “*Pietro è amico di Paolo*”

# Esempio: il mondo dei blocchi

Ci interessano i blocchi e alcune loro relazioni spaziali

*Dominio*: {a, b, c, d, e} ← *blocchi veri!*

*Le funzioni*: si individuano le funzioni rilevanti che servono anch'esse per identificare oggetti.

Es. *Hat* la funzione unaria che dato un blocco identifica il blocco che ci sta sopra;  $Hat(b)=a$

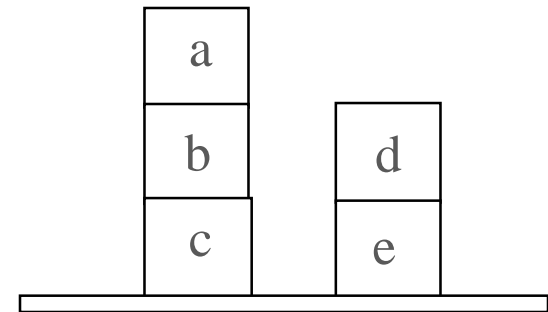
*Le relazioni*: si individuano le relazioni interessanti. Es.

*On* = {<a, b>, <b, c>, <d, e>}

*Clear* = {a, d}

*Table* = {c, e}

*Block* = {a, b, c, d, e}



# Concettualizzazione

$\langle \{a, b, c, d, e\}, \{Hat\}, \{On, Clear, Table, Block\} \rangle$

- Le concettualizzazioni possibili sono infinite: un aspetto importante è il livello di astrazione *giusto* per gli scopi della rappresentazione.

Es. se fosse rilevante il colore o la grandezza dei blocchi dovremmo introdurre predicati anche per questi aspetti

# La logica dei predicati del prim'ordine (FOL)

- Il linguaggio: vocabolario

- *Connettivo* →  $\wedge$  |  $\vee$  |  $\neg$  |  $\Rightarrow$  |  $\Leftrightarrow$  |  $\Leftarrow$

- *Quantificatore* →  $\forall$  |  $\exists$

- *Variabile* →  $x$  |  $y$  | ...  $a$  |  $s$  ... (lettere minuscole)

- *Costante* → Es.  $A$  |  $B$  | ... Mario | Pippo | 2 ...

- *Funzione* → Es. *Hat* | *Padre-di* |  $+$  |  $-$  | ...

(con *arità*  $\geq 1$ )    1                    1            2    2

- *Predicato* → Es. *On* | *Clear* |  $\geq$  |  $<$  ...

(con *arità*  $\geq 0$ )    2            1            2    2



# Il linguaggio: le formule

La sintassi delle formule:

*Formula-atomica*  $\rightarrow$  *True* | *False* |

*Termine = Termine* |

*Predicato* (*Termine*, ...)

(un numero di termini pari alla arità del predicato)

*Formula*  $\rightarrow$  *Formula-atomica* |

*Formula* *Connettivo* *Formula* |

*Quantificatore* *Variabile* *Formula* |

$\neg$  *Formula* | (*Formula*)



# Il linguaggio: formule ben formate

## Esempi di formule atomiche:

*Ama*(Giorgio, Lucia)

$+(2, 3) = 5$

*On*(A, B)

$x = 5$

*Madre-di*(Luigi) = Silvana

*Amico*(*Padre-di*(Giorgio), *Padre-di*(Elena))

## Esempi di formule complesse:

*On*(A, B)  $\wedge$  *On*(B, C)

(*congiunzione*)

*Studia*(Paolo)  $\Rightarrow$  *Promosso*(Paolo)

(*implicazione materiale*)

# Il linguaggio: quantificatori

- Quantificatore universale

- $\forall x \text{ Ama}(x, \text{Gelato})$

- Quantificatore esistenziale

- $\exists x \text{ Mela}(x) \wedge \text{Rossa}(x)$

- Nota: l'ordine dei quantificatori è importante:

- $\forall x (\exists y \text{ Ama}(x, y))$  *Tutti amano qualcuno*

- $\exists y (\forall x \text{ Ama}(x, y))$  *Esiste qualcuno amato da tutti*

- Ambito dei quantificatori:

*ambito di y*  
 $\forall x (\exists y \text{ Ama}(x, y))$

*ambito di x*  
 $\forall x (\exists y \text{ Ama}(x, y))$

# Formule *chiuse*, *aperte*, *ground*

- Di solito le variabili sono usate nell'ambito di quantificatori. In tal caso le *occorrenze* si dicono *legate*. Se non legate sono *libere*.

$Mela(x) \Rightarrow Rossa(x)$                        $x$  è libera in entrambe le occ.

$\forall x Mela(x) \Rightarrow Rossa(x)$                        $x$  è legata ...

$Mela(x) \Rightarrow \exists x Rossa(x)$                       la 1a è libera, la 2a legata

- *Def. Formula chiusa*: una formula che non contiene occorrenze di variabili libere.
- Altrimenti è detta *aperta*.
- *Def. Formula ground*: una formula che non contiene variabili.

# Il linguaggio: precedenza tra gli operatori

*Precedenza tra gli operatori logici:*

$= > \neg > \wedge > \vee > \Rightarrow, \Leftrightarrow > \forall, \exists$

Es.  $\forall x \text{ Persona}(x) \Rightarrow \text{Sesso}(x)=M \vee \text{Sesso}(x)=F$

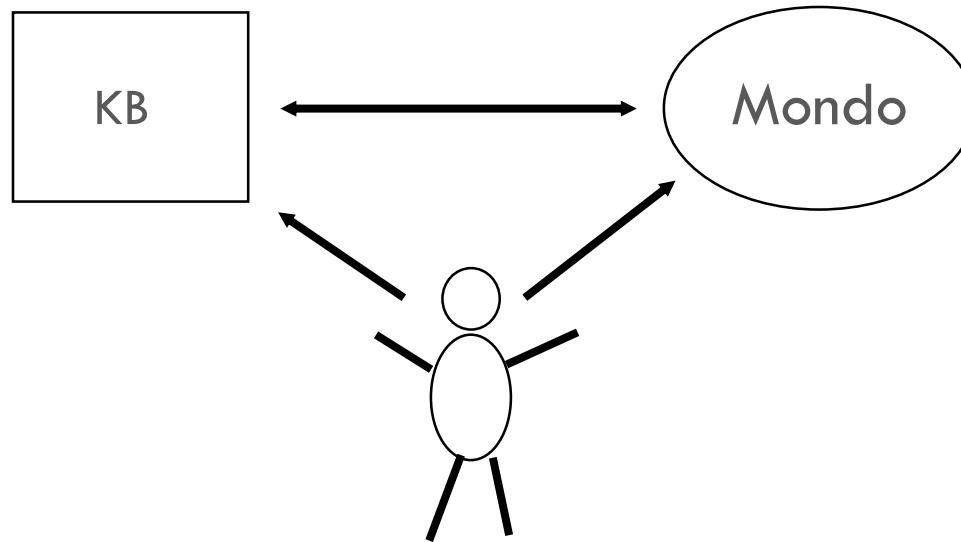
è da interpretare come ...

$\forall x \text{ Persona}(x) \Rightarrow (\text{Sesso}(x)=M) \vee (\text{Sesso}(x)=F)$

$\forall x \text{ Persona}(x) \Rightarrow ((\text{Sesso}(x)=M) \vee (\text{Sesso}(x)=F))$

$\forall x(\text{Persona}(x) \Rightarrow ((\text{Sesso}(x)=M) \vee (\text{Sesso}(x)=F)))$

# Semantica dichiarativa



Consiste nello stabilire una corrispondenza tra:

- i termini del linguaggio e gli oggetti del mondo
- le formule chiuse e i valori di verità

# Interpretazione

- Una interpretazione  $I$  stabilisce una corrispondenza precisa tra elementi atomici del linguaggio ed elementi della concettualizzazione.
- $I$  interpreta:
  - i simboli di costante come elementi del dominio  $D$
  - i simboli di funzione come funzioni da  $n$ -uple di  $D$  in  $D$ , ad es.  $f:D^n \rightarrow D$
  - i simboli di predicato come insiemi di  $n$ -uple, ad es.  $P \subseteq D^n$

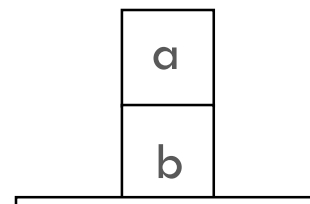
# Semantica: un esempio

*On*(A, B)

*Clear*(A)

*Table*(B)

Due interpretazioni possibili:



... quella intesa

$I(A)=a$

$I(B)=b$

$I(On)=\{ \langle a, b \rangle \}$

$I(Clear)=\{a\}$

$I(Table)=\{b\}$

... un'altra possibile

$I(A)=a$

$I(B)=b$

$I(On)=\{ \langle b, a \rangle \}$

$I(Clear)=\{b\}$

$I(Table)=\{a\}$

# Semantica composizionale

- Il significato di un termine o di una formula composta è determinato in funzione del significato dei suoi componenti:
  - es. *Sorella(Madre(Pietro))*
  - La formula  $A \wedge B$  è vera in una certa interpretazione se entrambe  $A$  e  $B$  sono vere
  - $\neg A$  è vera se  $A$  è falsa
  - $A \vee B$  è vera se  $A$  è vera oppure  $B$  è vera (o entrambe)
  - $A \Rightarrow B$  è vera se  $A$  è falsa oppure  $B$  è vera (come  $\neg A \vee B$ )



# Semantica ( $\forall$ )

- $\forall x A(x)$  è vera se per ciascun elemento del dominio  $A$  è vera di quell'elemento
- Se il dominio è finito equivale a un grosso  $\wedge$   
 $\forall x \text{Mortale}(x)$   
 $\text{Mortale}(\text{Gino}) \wedge \text{Mortale}(\text{Pippo}) \wedge \dots$
- Tipicamente, siccome difficilmente una proprietà è universale,  $\forall$  si usa quasi sempre insieme a  $\Rightarrow$   
 $\forall x \text{Persona}(x) \Rightarrow \text{Mortale}(x)$

# Semantica ( $\exists$ )

- $\exists x A(x)$  è vera se esiste almeno un elemento del dominio per cui  $A$  è vera
- Se il dominio è finito equivale a un grosso  $\vee$   
 $\exists x Persona(x)$   
 $Persona(Gino) \vee Persona(Pippo) \vee \dots$
- Tipicamente  $\exists$  si usa con  $\wedge$   
 $\exists x Persona(x) \wedge Speciale(x)$   
 $\exists x Persona(x) \Rightarrow Speciale(x)$     troppo debole

## Relazione tra $\forall$ ed $\exists$

$$\forall x \neg P(x) \equiv \neg \exists x P(x)$$

$$\neg \forall x P(x) \equiv \exists x \neg P(x)$$

$$\forall x P(x) \equiv \neg \exists x \neg P(x)$$

$$\neg \forall x \neg P(x) \equiv \exists x P(x)$$

$$\neg P \wedge \neg Q \equiv \neg(P \vee Q)$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q)$$

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$$

# Perché logica del *primo ordine*?

- Le variabili possono essere usate per denotare oggetti del dominio, non per denotare funzioni o predicati o formule.
- Funzioni e predicati possono essere oggetti del dominio; ma è così non possono essere usati al posto dei nomi di funzioni o predicati.

Es.  $\exists f \forall x f(x)=x$  (esistenza dell'identità)

NO

$\exists f$ . *Funzione-Identità*( $f$ )      SI

$\forall p$  *BuonaQualità*( $p$ )  $\Rightarrow$  *Ha*(*Giorgio*,  $p$ )      SI

$\forall p$  *BuonaQualità*( $p$ )  $\Rightarrow$   $p$ (*Giorgio*)      NO

- Il superamento di questa restrizione porta a linguaggi del second'ordine, o di ordine superiore.

# Semantica standard e semantica 'database'

- *Riccardo ha due fratelli: Giovanni e Goffredo*  
 $Fratello(Riccardo, Giovanni) \wedge Fratello(Riccardo, Goffredo)$   
 $\wedge Giovanni \neq Goffredo$   
 $\wedge \forall x Fratello(Riccardo, x) \Rightarrow (x = Giovanni) \vee (x = Goffredo)$
- **Semantica dei database**
  - Ipotesi dei nomi unici: simboli distinti, oggetti distinti
  - Ipotesi del mondo chiuso: tutto ciò di cui non si sa che è vero è falso
  - Chiusura del dominio: esistono solo gli oggetti di cui si parla

# Interazione con la KB in FOL

- Asserzioni
  - TELL(KB, *King(John)*)
  - TELL(KB,  $\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$ )
- Conseguenze logiche
  - ASK(KB, *Person(John)*)    Sì, se  $\text{KB} \models \text{Person}(\text{John})$
  - ASK(KB,  $\exists x \text{ Person}(x)$ )
    - 'Sì' sarebbe riduttivo
    - Lista di sostituzioni o legami: [ $\{x/\text{John}\} \{x/\text{George}\} \dots$ ]  
è una risposta più collaborativa

# Inferenza nella logica del prim'ordine

- Riduzione a inferenza proposizionale
- Il *metodo di risoluzione* per FOL
  - Trasformazione in forma a clausole
  - Unificazione
- Casi particolari: sistemi a regole
  - *Backward chaining* e programmazione logica
  - *Forward chaining* e basi di dati deduttive

# Regole di inferenza per $\forall$

- **Istanziamento dell'Universale** ( $\forall$  eliminazione)

$$\frac{\forall x A[x]}{A[g]}$$

dove  $g$  è un termine *ground* e  $A[g]$  è il risultato della sostituzione di  $g$  per  $x$  in  $A$ .

- Da:  $\forall x King(x) \wedge Greedy(x) \Rightarrow Evil(x)$  si possono ottenere
  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$



# Regole per l'esistenziale ( $\exists$ )

- **Istanziamento dell'esistenziale** ( $\exists$  eliminazione)

$$\frac{\exists x A[x]}{A[k]}$$

1. se  $\exists$  non compare nell'ambito di  $\forall$ ,  $k$  è una costante nuova (*costante di Skolem*)
2. altrimenti va introdotta una funzione (di Skolem) nelle variabili quantificate universalmente

$\exists x \text{ Padre}(x, G)$	diventa	$\text{Padre}(k, G)$
$\forall x \exists y \text{ Padre}(y, x)$	diventa	$\forall x \text{ Padre}(p(x), x)$
	e non	$\forall x \text{ Padre}(k, x)$

... altrimenti tutti avrebbero lo stesso padre !

# Riduzione a inferenza proposizionale

- **Proposizionalizzazione**
  - Creare tante istanze delle formule quantificate universalmente quanti sono gli oggetti menzionati
  - Eliminare i quantificatori esistenziali skolemizzando
- A questo punto possiamo trattare la KB come proposizionale e applicare gli algoritmi visti
- Problemi?
  - Le costanti sono in numero finito ...
  - ...ma se ci sono funzioni, il numero di istanze da creare è infinito: *John, Padre(John), Padre(Padre(John))* ...

# Teorema di *Herbrand*

- Se  $KB \models A$  allora c'è una dimostrazione che coinvolge solo un sotto-insieme finito della KB proposizionalizzata
- Si può procedere incrementalmente ...
  1. Creare le istanze con le costanti
  2. Creare poi quelle con un solo livello di annidamento  
*Padre(John), Madre(John)*
  3. Poi quelle con due livelli di annidamento  
*Padre(Padre(John)), Padre(Madre(John)) ...*
- Se  $KB \not\models A$  il processo non termina. E' quindi semidecidibile.

# Metodo di risoluzione per il FOL

- Abbiamo visto la regola di risoluzione per PROP: un metodo deduttivo corretto e completo con un'unica regola
- Possiamo estendere al FOL il metodo di risoluzione?
- SI. Ma per arrivare a definire la regola ...
  - Dobbiamo estendere al FOL la trasformazione in forma a clausole
  - Dobbiamo introdurre il concetto di **unificazione**

# Forma a clausole

- Costanti, funzioni, predicati sono come definiti, ma escludiamo nel seguito formule atomiche del tipo  $(t_1=t_2)$
- Una clausola è un insieme di **letterali**, che rappresenta la loro disgiunzione
  - $Clausola \rightarrow \{Letterale, \dots, Letterale\}$
  - $Letterale \rightarrow Formula\_atomica \mid \neg Formula\_atomica$
- Una KB è un insieme di clausole.

# Trasformazione in forma a clausole

- *Teorema*: per ogni formula chiusa  $\alpha$  del FOL è possibile trovare in maniera *effettiva* un insieme di clausole  $FC(\alpha)$  che è *soddisfacibile* sse  $\alpha$  lo era [*insoddisfacibile* sse  $\alpha$  lo era]
- Vediamo la trasformazione in dettaglio ... per la frase “*Tutti coloro che amano tutti gli animali sono amati da qualcuno*”  
$$\forall x (\forall y \text{ Animale}(y) \Rightarrow \text{Ama}(x,y)) \Rightarrow (\exists y \text{ Ama}(y, x))$$

# Trasformazione: passo 1

1. Eliminazione delle implicazioni ( $\Rightarrow$  e  $\Leftrightarrow$ ):

$A \Rightarrow B$  diventa  $\neg A \vee B$

$A \Leftrightarrow B$  diventa  $(\neg A \vee B) \wedge (\neg B \vee A)$

$\forall x (\forall y \text{Animale}(y) \Rightarrow \text{Ama}(x,y)) \Rightarrow (\exists y \text{Ama}(y, x))$

$\forall x \neg(\forall y \text{Animale}(y) \Rightarrow \text{Ama}(x,y)) \vee (\exists y \text{Ama}(y, x))$

$\forall x \neg(\forall y \neg\text{Animale}(y) \vee \text{Ama}(x,y)) \vee (\exists y \text{Ama}(y, x))$

# Trasformazione: passo 2

## 2. Negazioni all'interno

$\neg\neg A$  diventa  $A$

$\neg(A \wedge B)$  diventa  $\neg A \vee \neg B$  (De Morgan)

$\neg(A \vee B)$  diventa  $\neg A \wedge \neg B$  (De Morgan)

$\neg\forall x A$  diventa  $\exists x \neg A$

$\neg\exists x A$  diventa  $\forall x \neg A$

$\forall x \neg(\forall y \neg\text{Animale}(y) \vee \text{Ama}(x,y)) \vee (\exists y \text{Ama}(y, x))$

$\forall x (\exists y \neg(\neg\text{Animale}(y) \vee \text{Ama}(x,y))) \vee (\exists y \text{Ama}(y, x))$

$\forall x (\exists y (\text{Animale}(y) \wedge \neg\text{Ama}(x,y))) \vee (\exists y \text{Ama}(y, x))$



# Trasformazione: passo 3

3. Standardizzazione delle variabili: ogni quantificatore una variabile diversa

$$\forall x (\exists y (\text{Animale}(y) \wedge \neg \text{Ama}(x, y))) \vee (\exists y \text{Ama}(y, x))$$

$$\forall x (\exists y (\text{Animale}(y) \wedge \neg \text{Ama}(x, y))) \vee (\exists z \text{Ama}(z, x))$$

# Trasformazione: passo 4

## 4. Skolemizzazione: eliminazione dei quantificatori esistenziali

$$\forall x (\exists y (\text{Animale}(y) \wedge \neg \text{Ama}(x, y))) \vee (\exists z \text{Ama}(z, x))$$

*Ci sono due quantificatori esistenziali nell'ambito di uno universale, dobbiamo introdurre due funzioni di Skolem*

$$\forall x (\text{Animale}(F(x)) \wedge \neg \text{Ama}(x, F(x))) \vee \text{Ama}(G(x), x)$$

# Trasformazione: passo 5

## 5. Eliminazione quantificatori universali

- Possiamo portarli tutti davanti (forma *prenessa*)

$$(\forall x A) \vee B \quad \text{diventa} \quad \forall x (A \vee B)$$

$$(\forall x A) \wedge B \quad \text{diventa} \quad \forall x (A \wedge B)$$

equivalente se  $B$  non contiene  $x$

- ... e poi eliminarli usando la convenzione che le variabili libere sono quantificate universalmente

$$\forall x (Animale(F(x)) \wedge \neg Ama(x, F(x))) \vee Ama(G(x), x))$$

$$(Animale(F(x)) \wedge \neg Ama(x, F(x))) \vee Ama(G(x), x))$$

# Trasformazione: passo 6

6. Forma normale congiuntiva (congiunzione di disgiunzioni di letterali):

$$A \vee (B \wedge C) \quad \text{diventa} \quad (A \vee B) \wedge (A \vee C)$$

$$(Animale(F(x)) \wedge \neg Ama(x, F(x))) \vee Ama(G(x), x)$$

$$(Animale(F(x)) \vee Ama(G(x), x)) \wedge (\neg Ama(x, F(x)) \vee Ama(G(x), x))$$

# Trasformazione: passo 7

7. Notazione a clausole:

$$(Animale(F(x)) \vee Ama(G(x), x)) \wedge$$
$$(\neg Ama(x, F(x)) \vee Ama(G(x), x))$$
$$\{Animale(F(x)), Ama(G(x), x)\}$$
$$\{\neg Ama(x, F(x)), Ama(G(x), x)\}$$

# Trasformazione: passo 8

8. Separazione delle variabili: clausole diverse, variabili diverse

Nota:  $\forall x (P(x) \wedge Q(x)) \Leftrightarrow \forall x_1 P(x_1) \wedge \forall x_2 Q(x_2)$

$\{Animale(F(x)), Ama(G(x), x)\}$

$\{\neg Ama(x, F(x)), Ama(G(x), x)\}$

$\{Animale(F(x_1)), Ama(G(x_1), x_1)\}$

$\{\neg Ama(x_2, F(x_2)), Ama(G(x_2), x_2)\}$

NOTA: tutti i passi meno la Skolemizzazione preservano l'equivalenza delle formule.

$P(a) \models \exists x P(x)$       ma       $\exists x P(x) \not\models P(a)$

# Forma normale implicativa

Forma normale implicativa (forse più intuitiva)

$$\neg P_1 \vee \dots \vee \neg P_k \vee Q_1 \vee \dots \vee Q_n$$
$$\neg(P_1 \wedge \dots \wedge P_k) \vee Q_1 \vee \dots \vee Q_n$$
$$P_1 \wedge \dots \wedge P_k \Rightarrow Q_1 \vee \dots \vee Q_n$$

Caso particolare (un solo letterale positivo, clausole di Horn)

$$P_1 \wedge \dots \wedge P_k \Rightarrow Q$$

Forma a regole come in programmazione logica o nelle basi di dati deduttive

# Unificazione: definizione

- *Unificazione*: operazione mediante la quale si determina se due espressioni possono essere rese identiche mediante una **sostituzione** di termini alle variabili
- Il risultato è la sostituzione che rende le due espressioni identiche, detta *unificatore*, o FAIL, se le espressioni non sono unificabili



# Sostituzione

- *Sostituzione*: un insieme finito di associazioni tra variabili e termini, in cui ogni variabile compare una sola volta sulla sinistra.

Es.  $\{x_1/A, x_2/f(x_3), x_3/B\}$

Il significato è che  $A$  va sostituita a  $x_1$ ,  $f(x_3)$  va sostituito a  $x_2$  ...

Es.  $\{x/g(y), y/z, z/f(x)\}$

- Nota: sulla sinistra solo variabili

# Applicazione di sostituzione

Sia  $\sigma$  una sostituzione,  $A$  un'espressione:

- $A\sigma$  istanza generata dalla sostituzione (delle variabili con le corrispondenti espressioni)

*Esempi.*

$$P(x, x, y, v)\{x/A, y/f(B), z/w\} = P(A, A, f(B), v)$$

$$Q(x, y, z) \{x/g(y), y/z, z/f(x)\} = Q(g(y), z, f(x))$$

*Nota:* le variabili vengono sostituite **simultaneamente** e si esegue un solo passo di sostituzione

# Espressioni unificabili

- *Espressioni unificabili*: se esiste una sostituzione che le rende identiche (*unificatore*)  
Es.  $P(A, y, z)$  e  $P(x, B, z)$  sono unificabili con  
 $\tau = \{x/A, y/B, z/C\}$
- $\tau$  è un unificatore, ma non l'unico ... un altro è  
 $\sigma = \{x/A, y/B\}$
- $\sigma$  è *più generale* di  $\tau$  (istanza 'meno')
- vorremmo l'*unificatore più generale* di tutti (MGU)
- *Teorema*: l'unificatore più generale è unico, a parte i nomi delle variabili (l'ordine non conta).

# Composizione di sostituzioni

- Siano  $\sigma$  e  $\tau$  due sostituzioni:

$$\sigma = [t_1/x_1, \dots, t_k/x_k] \text{ e } \tau = [s_1/y_1, \dots, s_n/y_n]$$

1.  $\sigma\tau' = [t_1\tau/x_1, \dots, t_k\tau/x_k, s_1/y_1, \dots, s_n/y_n]$
2. Eliminare da  $\sigma\tau'$  le identità (Es.  $x/x$ ) e le coppie  $s_i/y_i$  tali che  $y_i \in \{x_1, \dots, x_k\}$ , ottenendo così  $\sigma\tau$

- Es.  $\sigma = [g(x, y)/w, x/y]$        $\tau = [y/x, B/w, C/z]$

1.  $\sigma\tau' = [g(y, y)/w, y/y, y/x, B/w, C/z]$

2.  $\sigma\tau = [g(y, y)/w, y/x, C/z]$

# Unificatore più generale

- *L'unificatore più generale  $\gamma$  (MGU): è tale che ogni altro unificatore può essere ottenuto componendo  $\gamma$  con qualche sostituzione:*

$$\forall \sigma \exists \delta \text{ tale che } \gamma\delta = \sigma$$

Es. l'MGU di  $P(A, y, z)$  e  $P(x, y, v)$  è  $[A/x, z/v]$ ;

un altro unificatore è:

$$[A/x, B/y, z/v] = [A/x, z/v][B/y]$$

# Algoritmo di unificazione: le regole

1.  $f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \rightarrow s_1 = t_1, \dots, s_n = t_n$
2.  $f(s_1, \dots, s_n) = g(t_1, \dots, t_m) \rightarrow \text{fail}$  se  $f \neq g$  o  $n \neq m$
3.  $x = x \rightarrow \text{cancella}$
4.  $t = x \rightarrow x = t$
5.  $x = t$ ,  $x$  non occorre in  $t \rightarrow$  applica  $\{x/t\}$  a tutte le altre equazioni
6.  $x = t$ ,  $t$  non è  $x$ ,  $x$  occorre in  $t \rightarrow \text{fail}$  (*occur check*)

Nota: come caso particolare della 2, quando  $n=m=0$ , si fallisce su due costanti diverse

# Algoritmo di unificazione: esempio 1

- Calcolo dell'MGU tra  $P(A, y, z)$  e  $P(x, B, z)$

Passo 0

$$P(A, y, z) = P(x, B, z) \quad \text{regola 1}$$

# Algoritmo di unificazione: esempio 1

- Calcolo dell'MGU tra  $P(A, y, z)$  e  $P(x, B, z)$

Passo 1

$A = x$             regola 4

$y = B$

$z = z$

1.  $f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \rightarrow s_1 = t_1, \dots, s_n = t_n$
2.  $f(s_1, \dots, s_n) = g(t_1, \dots, t_m) \rightarrow \text{fail}$  se  $f \neq g$  o  $n \neq m$
3.  $x = x \rightarrow \text{cancella}$
4.  $t = x \rightarrow x = t$
5.  $x = t$ ,  $x$  non occorre in  $t \rightarrow$  applica  $\{x/t\}$  a tutte le altre equazioni
6.  $x = t$ ,  $t$  non è  $x$ ,  $x$  occorre in  $t \rightarrow \text{fail}$  (*occur check*)



# Algoritmo di unificazione: esempio 1

- Calcolo dell'MGU tra  $P(A, y, z)$  e  $P(x, B, z)$

## Passo 2

$$x = A$$

$$y = B$$

$$z = z$$

regola 3

1.  $f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \rightarrow s_1 = t_1, \dots, s_n = t_n$
2.  $f(s_1, \dots, s_n) = g(t_1, \dots, t_m) \rightarrow \text{fail}$  se  $f \neq g$  o  $n \neq m$
3.  $x = x \rightarrow \text{cancella}$
4.  $t = x \rightarrow x = t$
5.  $x = t$ ,  $x$  non occorre in  $t \rightarrow$  applica  $\{x/t\}$  a tutte le altre equazioni
6.  $x = t$ ,  $t$  non è  $x$ ,  $x$  occorre in  $t \rightarrow \text{fail}$  (*occur check*)

# Algoritmo di unificazione: esempio 1

- Calcolo dell'MGU tra  $P(A, y, z)$  e  $P(x, B, z)$

## Passo 3

$$x = A$$

$$y = B$$

MGU!

1.  $f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \rightarrow s_1 = t_1, \dots, s_n = t_n$
2.  $f(s_1, \dots, s_n) = g(t_1, \dots, t_m) \rightarrow \text{fail}$  se  $f \neq g$  o  $n \neq m$
3.  $x = x \rightarrow \text{cancella}$
4.  $t = x \rightarrow x = t$
5.  $x = t$ ,  $x$  non occorre in  $t \rightarrow$  applica  $\{x/t\}$  a tutte le altre equazioni
6.  $x = t$ ,  $t$  non è  $x$ ,  $x$  occorre in  $t \rightarrow \text{fail}$  (*occur check*)

# Algoritmo di unificazione: esempio 2

- Calcolo dell'MGU tra  $P(f(x), x)$  e  $P(z, z)$

Passo 3

$$z = f(x)$$

$$x = f(x) \quad \text{regola 6}$$

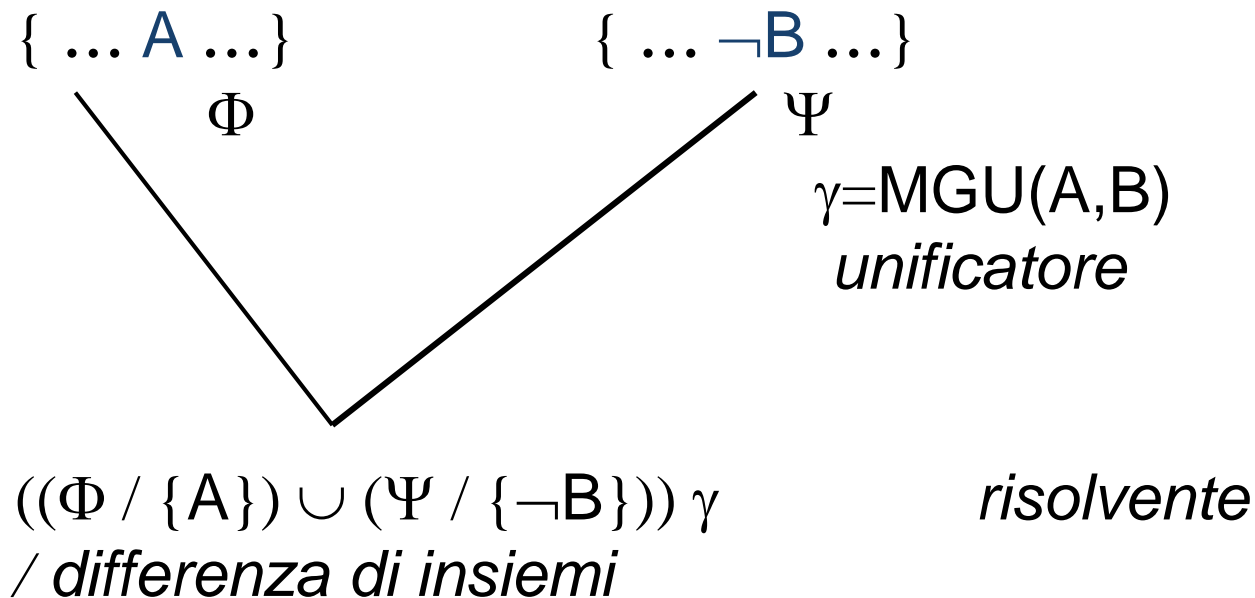
**FAIL!**

(occurr check)

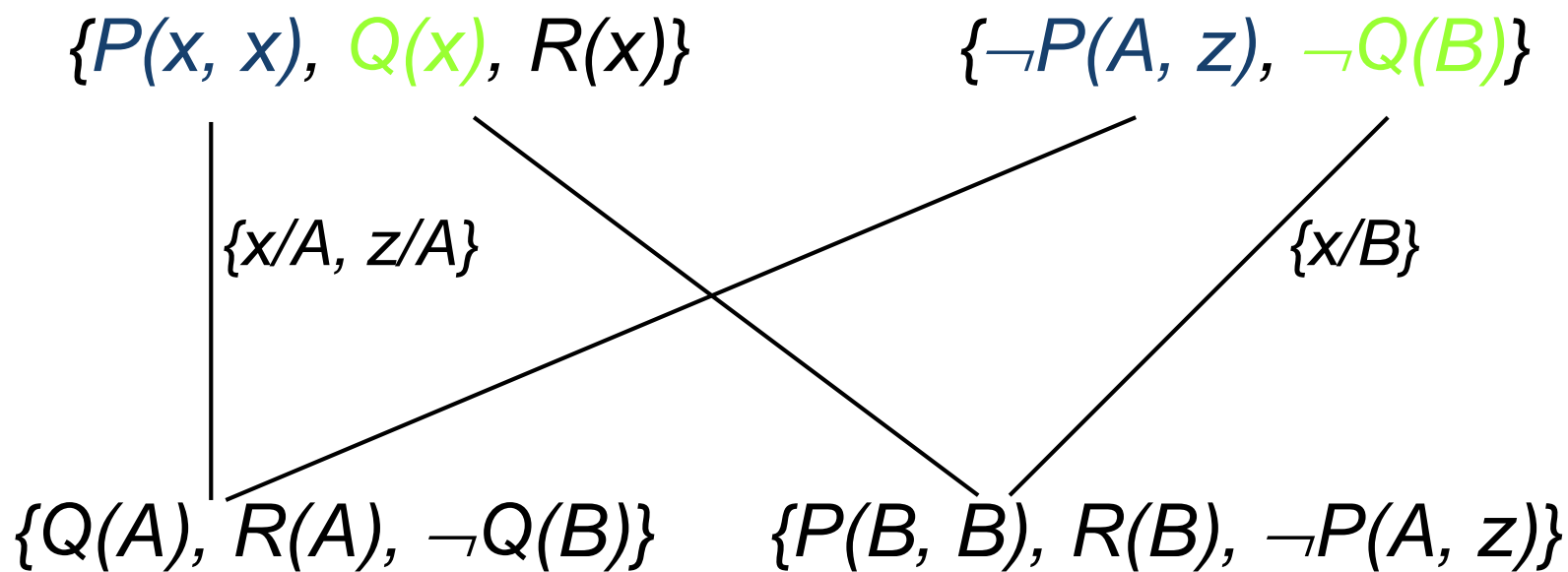
1.  $f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \rightarrow s_1 = t_1, \dots, s_n = t_n$
2.  $f(s_1, \dots, s_n) = g(t_1, \dots, t_m) \rightarrow \text{fail}$  se  $f \neq g$  o  $n \neq m$
3.  $x = x \rightarrow \text{cancella}$
4.  $t = x \rightarrow x = t$
5.  $x = t$ ,  $x$  non occorre in  $t \rightarrow$  applica  $\{x/t\}$  a tutte le altre equazioni
6.  $x = t$ ,  $t$  non è  $x$ ,  $x$  occorre in  $t \rightarrow \text{fail}$  (*occurr check*)

# Il metodo di risoluzione per il FOL

- Siamo ora in grado di definire in generale la regola di risoluzione per FOL



# Risoluzione: esempio



*Grafo di risoluzione*

# Problema dei fattori

- Le seguenti clausole dovrebbero produrre la clausola vuota invece ...

$\{P(u), P(v)\}$

$\{\neg P(x), \neg P(y)\}$

$\{P(v), \neg P(y)\}$  e qui ci si ferma

- Se un sottoinsieme dei letterali di una clausola può essere unificato allora la clausola ottenuta dopo tale unificazione si dice *fattore* della clausola originaria.

- Il metodo di risoluzione va applicato ai *fattori* delle clausole:

$\{P(u)\}$

$\{\neg P(x)\}$

$\{\}$

# Completezza del metodo di risoluzione

- La deduzione per risoluzione è corretta

*Correttezza:* Se  $\Gamma \vdash_{RES} A$  allora  $\Gamma \models A$

- La deduzione per risoluzione *non* è completa:  
può essere  $\Gamma \models A$  e non  $\Gamma \vdash_{RES} A$

- Infatti un semplice controesempio è:

- $\{\} \models \{P(a), \neg P(a)\}$  per cui non vale  
 $\{\} \vdash_{RES} \{P(a), \neg P(a)\}$

# Risoluzione per *refutazione*

- Il *teorema di refutazione* ci suggerisce un metodo alternativo *completo*
- *Teorema di refutazione:*  
 $\Gamma \cup \{\neg A\}$  è insoddisfacibile sse  $\Gamma \models A$
- *Teorema:*  $\Gamma$  è insoddisfacibile sse  $\Gamma \vdash_{RES} \{ \}$   
(la risoluzione è completa rispetto alla refutazione)

Abbiamo un metodo *meccanizzabile, corretto* e *completo*: basta aggiungere il negato della formula da dimostrare e provare a generare la clausola vuota



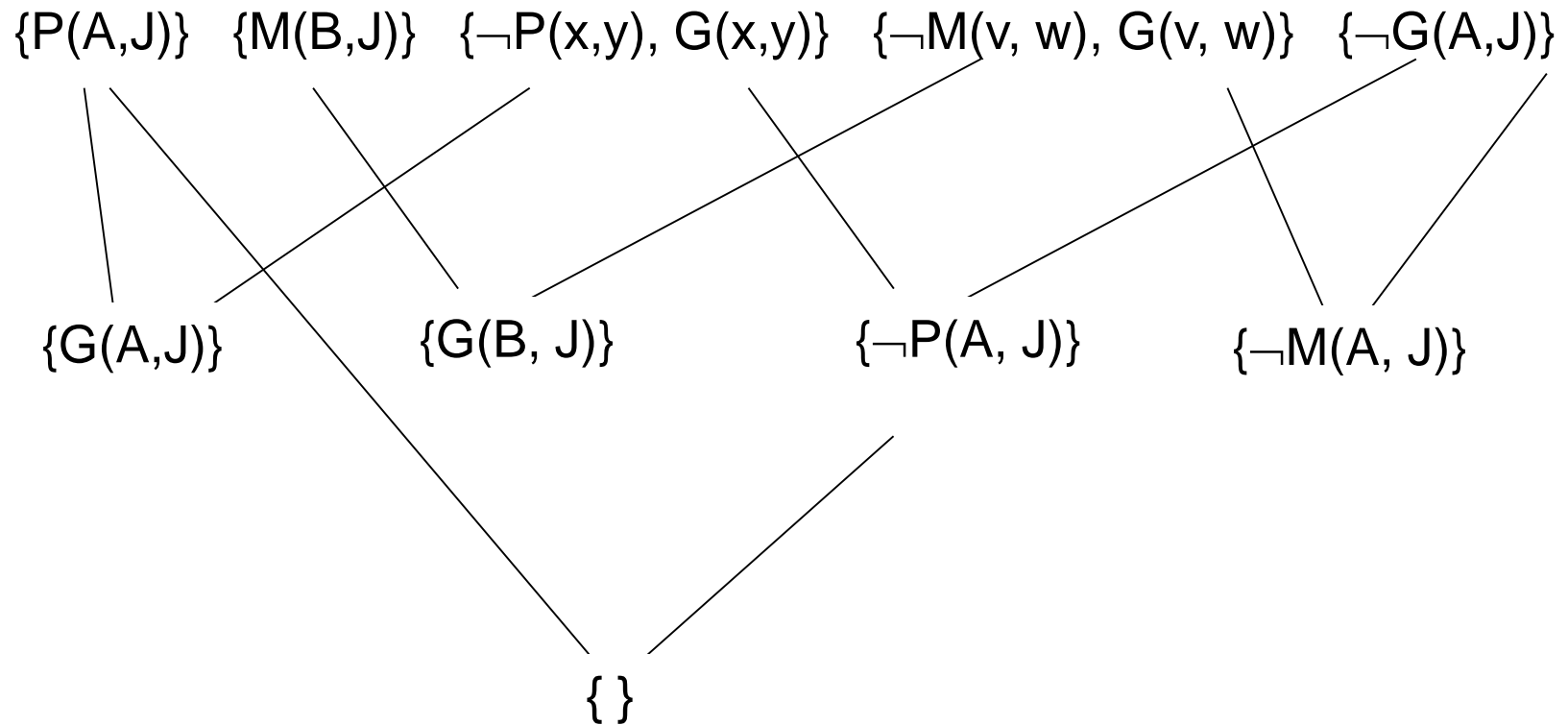
# Esempio di refutazione

1.	$\{P(A, J)\}$	] KB	<i>A è padre di J</i>
2.	$\{M(B, J)\}$		<i>B è madre di J</i>
3.	$\{\neg P(x, y), G(x, y)\}$		<i>padre implica genitore</i>
4.	$\{\neg M(v, w), G(v, w)\}$		<i>madre implica genitore</i>
•	<i>Goal: G(A, J)?</i>		<i>A è genitore di J?</i>

- Aggiungiamo a KB la negazione del goal e proviamo a dedurre la clausola vuota

5.  $\{\neg G(A, J)\}$

# Esempio di refutazione: il grafo



## Refutazione per domande di tipo “trova ...”

- Esempio: “*Chi sono i genitori di J?*”

- Si cerca di dimostrare che

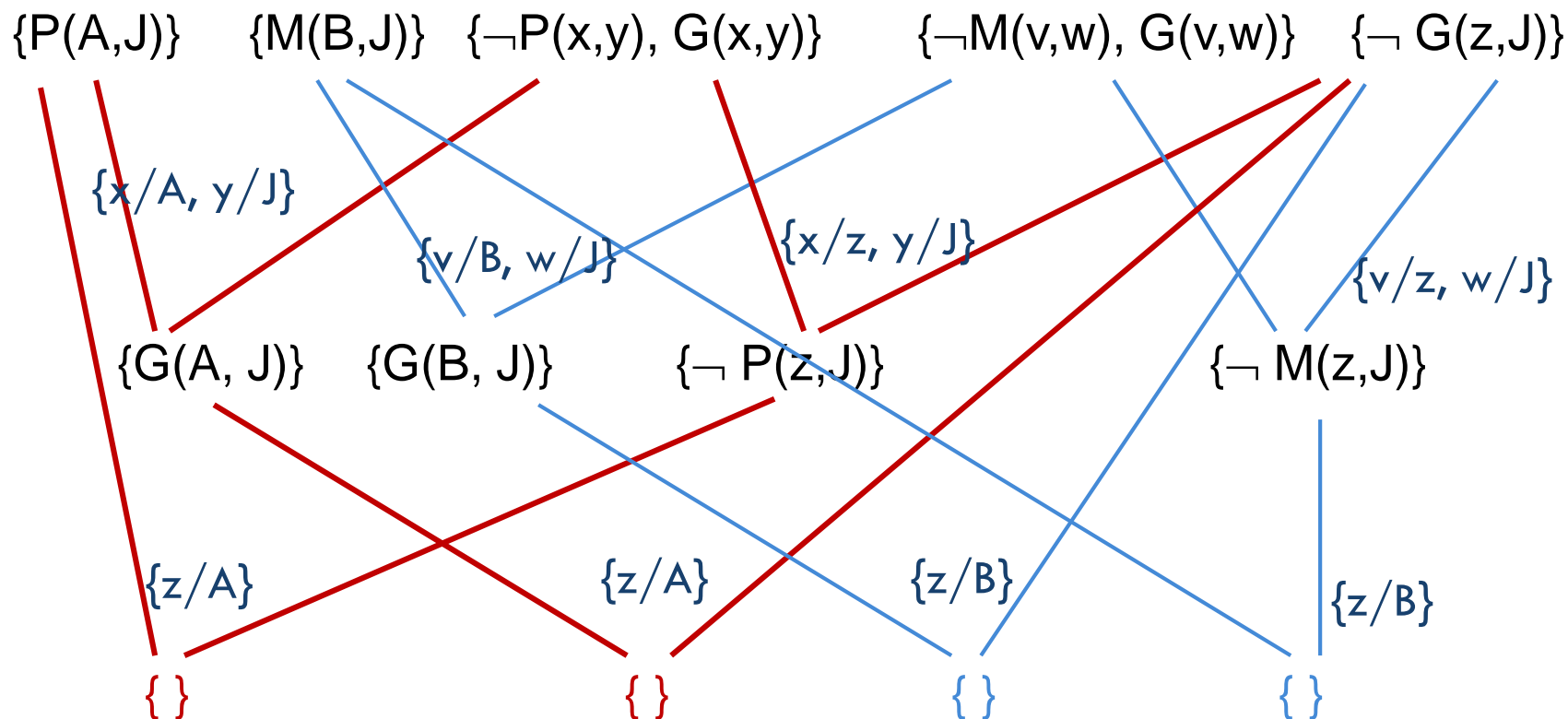
$$\exists z G(z, J)$$

- Clausola *goal* (negato):

$$FC(\neg \exists z G(z, J)) \rightarrow \{ \neg G(z, J) \}$$

- La risposta sono tutti i possibili *legami* per  $z$  che consentono di ottenere la clausola vuota (risposta calcolata)

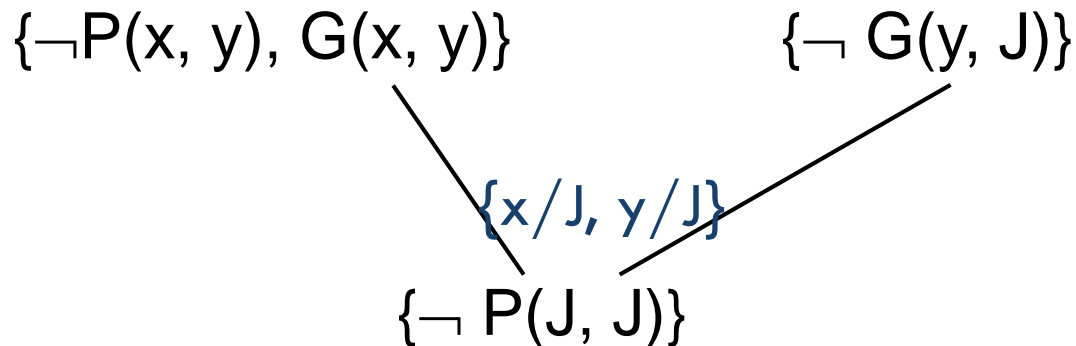
# Esempio



Le risposte sono: **A**, **B**

# Importante ridenominare!

*Osserva:* è importante la restrizione che ogni clausola usi variabili diverse (anche quelle generate)



... e a questo punto non avremmo potuto ottenere la risposta unificando con  $P(A, J)$

# SummarAlzing

- La logica dei predicate esprime in modo più natural le conoscenze degli agenti sul mondo
- La Sintassi e la Semantica dei linguaggi logici vengono utilizzati come metodi di formalizzazione del dominio per automatizzare l'uso della conoscenza nella ricerca di soluzioni a problemi contingent
- Nella direzione di automatizzare la verifica di proprietà utili alla ricerca (ad es. minimizzare il costo/pericolo o ottimizzare la utilità della soluzione) è necessario la applicazione alla dimostrazione della conseguenza logica di metodi algoritmici efficienti per la deduzione
- Le tabelle di verità possono essere applicate come nel caso del CPROP attraverso la proposizionalizzazione delle formule ben formate del CPRED
  - Il teorema di Herbrand ci aiuta
  - Esso è consistente ma non complete
  - Esso ci fornisce però uno strumento completo applicando la prova per refutazione (*risoluzione ground*)

# SummarAlzing

- L'alternativa è usare metodi che definiscono in modo più restrittivo i tipi di formule, le standardizzano e che applicano la risoluzione attraverso i meccanismi di unificazione
- E' la risoluzione applicata ad un s.i. proprio delle formule ben formate che si condurrà alla programmazione logica, dotata di algoritmi efficienti consistenti e completi per la deduzione