Agenti logici: la logica del prim'ordine

Sintassi, semantica, inferenza

Maria Simi

a.a. 2021-2022

La logica del prim'ordine 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 interessanno i blocchi e alcune loro relazioni spaziali

Dominio: {a, b, c, d, e} $\leftarrow$ *blocchi veri!*

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

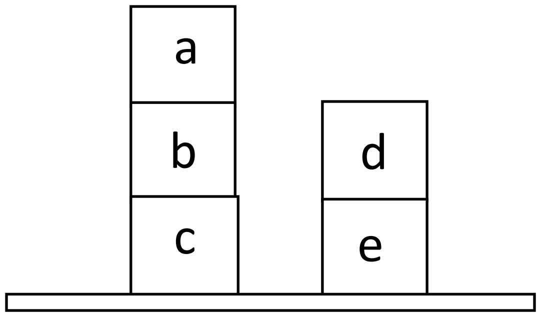
Es. *Hat* la funzione unaria che dato un blocco identifica il blocco che ci sta sopra; *Hat*(b)=a

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

<{a, b, c, d, e}, {*Hat*}, {*On*, *Clear*, *Table*, *Block*}>

- 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* $\rightarrow \wedge \mid \vee \mid \neg \mid \Rightarrow \mid \Leftrightarrow \mid \Leftarrow$
  - *Quantificatore* $\rightarrow \forall \mid \exists$
  - *Variabile* $\rightarrow x \mid y \mid \ldots a \mid s \ldots$ (lettere minuscole)
  - *Costante* $\rightarrow$ Es. A | B | … Mario | Pippo | 2 … (lettere maiuscole)
  - *Funzione* $\rightarrow$ Es. *Hat* | *Padre-di* | + | - | …

    (con *arità* $\geq$ 1) 1 1 2 2
  - *Predicato* $\rightarrow$ Es. *On* | *Clear* | $\geq$ | $<$ …

    (con *arità* $\geq$ 0) 2 1 2 2

Il linguaggio: i termini

- La sintassi dei termini:

  $\mathit{Termine} \rightarrow \mathit{Costante} \mid \mathit{Variabile} \mid \mathit{Funzione}\ (\mathit{Termine}, \ldots)$

  (un numero di termini pari alla arità della funzione)

- Esempi di termini ben formati:

f(x, y)	+(2, 3)
Padre-di(Giovanni)	x, A, B, 2
Prezzo(Banane)	*Hat*(A)

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
  - $\forall x\ Ama(x, Gelato)$
- Quantificatore esistenziale
  - $\exists x\ Mela(x) \wedge Rossa(x)$
- Nota: l'ordine dei quantificatori è importante:
  - $\forall x\ (\exists y\ Ama(x, y))$ *Tutti amano qualcuno*
  - $\exists y\ (\forall x\ Ama(x, y))$ *Esiste qualcuno amato da tutti*
- Ambito dei quantificatori:

ambito di ∃y *ambito di ∀x*

$\forall x\ (\exists y\ Ama(x, y))$ $\forall x\ (\exists y\ Ama(x, y))$

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 occorrenze
$\forall x\ Mela(x) \Rightarrow Rossa(x)$	x è legata …
$Mela(x) \Rightarrow \exists x\ Rossa(x)$	la prima è libera, la seconda 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. $\forall x\ \mathit{Persona}(x) \Rightarrow \mathit{Sesso}(x)=M \vee \mathrm{Sesso}(x)=F$

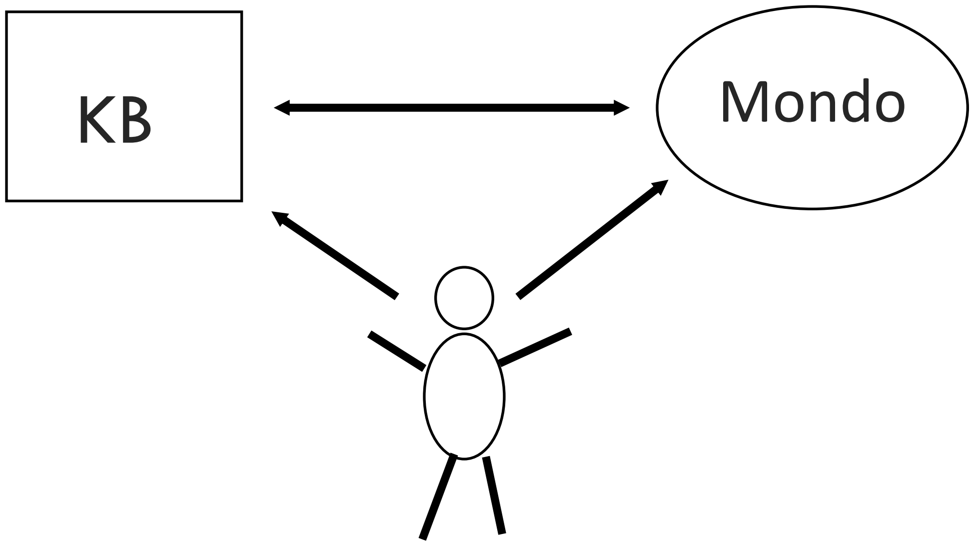
è da interpretare come …

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

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

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

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

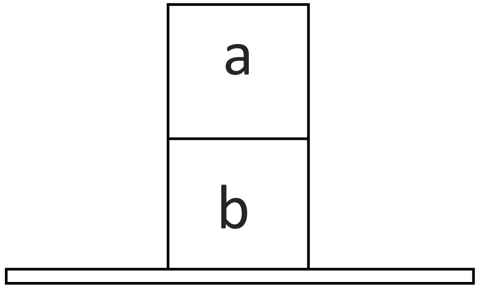
Interpretazione

- Una interpretazione $\mathscr{I}$ stabilisce una corrispondenza precisa tra elementi atomici del linguaggio ed elementi della concettualizzazione. $\mathscr{I}$ interpreta:
  - i simboli di costante come elementi del dominio
  - i simboli di funzione come funzioni da *n*-uple di D in D
  - i simboli di predicato come insiemi di *n*-uple

Semantica: un esempio

On(A, B)
Clear (A)
Table (B)

Due interpretazioni possibili:



… quella intesa

$\mathscr{I}(A)=a$

$\mathscr{I}(B)=b$

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

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

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

… un'altra possibile

$\mathscr{I}(A)=a$

$\mathscr{I}(B)=b$

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

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

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

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 $\wedge$

  $\forall x$ Mortale(x)

  Mortale(Gino) $\wedge$ Mortale(Pippo) $\wedge$ …
- Tipicamente $\forall$ si usa quasi sempre insieme a $\Rightarrow$

  $\forall x$ Persona(x) $\Rightarrow$ Mortale(x)

  Difficilmente una proprietà è universale; le condizioni nell'antecedente restringono la portata dell'asserzione e la qualificano.

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) $\vee$ …
- Tipicamente $\exists$ si usa con $\wedge$

  $\exists x$ Persona(x) $\wedge$ Speciale(x)

  $\exists x$ Persona(x) $\Rightarrow$ Speciale(x)

  Troppo debole: no sto nemmeno asserendo che esiste una persona.

Relazione tra ∀ e ∃

Da qui discendono delle proprietà che mettono in relazione ∀ e ∃.

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

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

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

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

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

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

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

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

Semantica “standard” e semantica “database”

- *Riccardo ha solo due fratelli: Giovanni e Goffredo*. In logica classica:

  $\text{Fratello}(\text{Riccardo}, \text{Giovanni}) \land \text{Fratello}(\text{Riccardo}, \text{Goffredo})$

  $\land\ \text{Giovanni} \neq \text{Goffredo}$

  $\land\ \forall x\ \text{Fratello}(\text{Riccardo}, x) \Rightarrow (x = \text{Giovanni}) \lor (x = \text{Goffredo})$

- Semantica dei database (e di alcuni linguaggi per la RC)
  - 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, *King*(*George*)),

  TELL(KB, $\forall$x *King*(x) $\Rightarrow$ *Person*(x))

- Conseguenze logiche

  ASK(KB, *Person*(*John*))   Sì, se KB $\models$ *Person*(*John*)

  ASK(KB, $\exists$x *Person*(x))

  - ✓ ’Sì’ sarebbe riduttivo
  - ✓ La risposta è una lista di sostituzioni o legami: [{x/John} {x/George} …]

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 $\forall$

- Istanziazione 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 (∃)

- Istanziazione dell'esistenziale (∃-eliminazione)

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

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

∃x Padre(x, G)	diventa	Padre(K, G)
∀x ∃y Padre(x, y)	diventa	∀x Padre(x, p(x))
	e non	∀x Padre(x, K)

... altrimenti tutti avrebbero lo stesso padre !

Riduzione a inferenza proposizionale

- Proposizionalizzazione (*Grounding*)
  - 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)) Madre(Padre(John)) Madre(Madre(John)) …
- Se KB $\not\models$ A il processo non termina. Il problema è *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 ...
  1. Dobbiamo estendere al FOL la trasformazione in forma a clausole
  2. 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*, … , *Letterale*}

  Letterale $\rightarrow$ *Formula_atomica* | $\neg$*Formula_atomica*

- Una KB è un insieme di clausole.

Trasformazione in forma a clausole

- *Teorema*: per ogni formula chiusa α del FOL è possibile trovare in maniera *effettiva* un insieme di clausole $FC(\alpha)$ che è *soddisfacibile* sse α lo era [*insoddisfacibile* sse α 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 $(\neg A \vee B) \wedge (\neg B \vee A)$

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

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

$\forall x \; \neg(\forall y \; \neg Animale(y) \vee Ama(x,y)) \vee (\exists y \; Ama(y, x))$

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	$\forall x\, \neg A$	

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

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

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

Trasformazione: passo 3

3. Standardizzazione delle variabili: facciamo in modo che ogni quantificatore usi una variabile diversa

$$\forall x\ (\exists y\ (Animale(y) \land \neg\ Ama(x, y))) \lor (\exists y\ Ama(y, x))$$

$$\forall x\ (\exists y\ (Animale(y) \land \neg\ Ama(x, y))) \lor (\exists z\ Ama(z, x))$$

Trasformazione: passo 4

4. Skolemizzazione: eliminazione dei quantificatori esistenziali

   $\forall x\ (\exists y\ (Animale(y) \land \neg\ Ama(x, y))) \lor (\exists z\ Ama(z, x))$

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

   $\forall x\ (Animale(F(x)) \land \neg Ama(x, F(x))) \lor Ama(G(x), x))$

Trasformazione: passo 5

5. Eliminazione quantificatori universali
   - Possiamo portarli tutti davanti (forma *prenessa*)

     $(\forall x\ A) \vee B$ diventa $\forall x\ (A \vee B)$

     $(\forall x\ A) \wedge B$ diventa $\forall x\ (A \wedge B)$

     equivalente se B non contiene x
   - … e poi eliminarli usando la convenzione che le variabili libere sono quantificate universalmente

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

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

Trasformazione: passo 6

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

   $A \vee (B \wedge C)$ diventa $(A \vee B) \wedge (A \vee C)$

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

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

Trasformazione: passo 7

7. Notazione a clausole:

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

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

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

Trasformazione: passo 8

8. Separazione delle variabili: clausole diverse, variabili diverse

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

$\{Animale(F(x)), Ama(G(x), x)\} \rightarrow \{Animale(F(x_1)), Ama(G(x_1), x_1)\}$
$\{\neg Ama(x, F(x)), Ama(G(x), x)\} \rightarrow \{\neg Ama(x_2, F(x_2)), Ama(G(x_2), x_2)\}$

NOTA: tutti i passi meno la Skolemizzazione preservano l'equivalenza delle formule. È comunque preservata la soddisfacibilità.

$P(a) \vDash \exists x\, P(x)$ ma $\exists x\, P(x) \nvDash P(a)$

Unificazione: definizione

- *Unificazione*: operazione mediante la quale si determina se due espressioni possono essere rese **identiche** mediante una sostituzione di termini a 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. $\{x_1/A, x_2/f(x_4), x_3/B\}$

  Il significato è che A va sostituita a x_1, $f(x_4)$ va sostituito a x_2, B a x_3

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

- Nota: sulla sinistra solo variabili, sulla destra costanti, variabili, funzioni … con la restrizione che una variabile sulla sinistra non può comparire anche sulla destra.
  - $\{x/f(x)\}$ NO, sostituzione circolare
  - $\{x/g(y), \mathbf{y}/z\}$ NO, non normalizzata $\{x/g(z), \mathbf{y}/z\}$

Applicazione di sostituzione

Sia σ una sostituzione, A un'espressione:

- $A\sigma$ istanza generata dalla sostituzione (delle variabili con le corrispondenti espressioni)
- In AIMA $\text{S}\scriptstyle{\text{UBST}}\displaystyle(\sigma, A)$

  Esempio.

  $\text{S}\scriptstyle{\text{UBST}}\displaystyle(\{x/A, y/f(B), z/w\}, P(x, x, y, v)) = P(A, A, f(B), v)$

  $\text{S}\scriptstyle{\text{UBST}}\displaystyle(\{x/g(y), y/z, z/f(x)\}, Q(x, y, z)) = Q(g(y), z, f(x))$ non normalizzata

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*) oppure FAIL

  Es. $\mathrm{P(A, y, z.\,)}$ e $\mathrm{P(x, B, z)}$ sono unificabili con $\tau=\{x/A, y/B, z/C\}$
- τ è un unificatore, ma non l'unico … un altro è

  $\sigma=\{x/A, y/B\}$
- σ è *più generale* di τ (istanzia '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).

Algoritmo di unificazione

- L'algoritmo di unicazione prende in input due espressioni p, q e restituisce un MGU θ se esiste
  - UNIFY(p, q)=θ tale che SUBST(θ, p)= SUBST(θ, q)
  - altrimenti FAIL
- L'algoritmo esplora in parallelo le due espressioni e costruisce l'unificatore strada facendo
- Appena trova espressioni non unificabili fallisce
- Una causa di fallimento sono sostituzioni del tipo $x=f(x)$; questo controllo si chiama *occurr check*

Algoritmo di unificazione (AIMA)

```
function UNIFY(x, y, θ) returns a substitution or failure
  θ, the substitution built up so far, θ={ } per default
  if θ = failure then return failure
  else if  x = y  then return θ                               // caso di successo
  else if  VARIABLE?(x)  then return  UNIFY-VAR(x, y, θ)
  else if  VARIABLE?(y)  then return  UNIFY-VAR(y, x, θ)
  else if  COMPOUND?(x) and  COMPOUND?(y)  then              //  OP(ARGS)  è il compound
     return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if  LIST?(x) and LIST?(y) then                         // (arg1, arg2, … argk) x lista
     return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure
```

Algoritmo di unificazione (cont.)

function UNIFY-VAR(var, x, θ) **returns** *a substitution or failure* — *var è una variabile*

if {var/val} ∈ θ **then return** UNIFY(val, x, θ) — *var ha già un valore*

else if {x/val} ∈ θ **then return** UNIFY(var, val , θ) — *x ha già un valore*

else if OCCUR-CHECK?(var, x) **then return** failure — *controllo di occorrenza*

else return EXTEND({var/x} , θ)

OCCUR-CHECK controlla se var occorre all'interno dell'espressione x. In tal caso fallisce. È un controllo di complessità quadratica.

ATTENZIONE: EXTEND non aggiunge semplicemente, ma applica la sostituzione in θ, diversamente la sostituzione non sarebbe normalizzata.

Esempio 1: P(A, y, z) e P(x, B, z)

- UNIFY(P(A, y, z), P(x, B, z), { })
- UNIFY((A, y, z), (x, B, z), UNIFY(P, P, { }))
- UNIFY((A, y, z), (x, B, z), { })
- UNIFY((y, z), (B, z), UNIFY(A, x, { }))
- UNIFY((y, z), (B, z), UNIFY(x, A, { }))
- UNIFY((y, z), (B, z), UNIFY-VAR(x, A, { }))
- UNIFY((y, z), (B, z), {x/A})
- UNIFY((z), (z), {y/B, x/A})
- UNIFY(z, z, {y/B, x/A})
- {y/B, x/A}

Esempio 2: P(f(x), x) e P(z, z)

- UNIFY(P(f(x), x), P(z, z), { })
- UNIFY((f(x), x), (z, z),UNIFY(P, P, { }))
- UNIFY((f(x), x), (z, z), { })
- UNIFY((x), (z), UNIFY(f(x), z, { })
- UNIFY((x), (z), UNIFY(z, f(x), { }))
- UNIFY((x), (z), {z/f(x)})
- UNIFY-VAR(x, z, {z/f(x)})
- UNIFY(x, f(x), {z/f(x)})
- OCCUR-CHECK(x, f(x))
- FAIL

Il metodo di risoluzione per il FOL

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

$\{ \dots A \dots \}$ $\qquad\qquad$ $\{ \dots \neg B \dots \}$

Φ $\qquad\qquad$ Ψ

γ=MGU(A,B)
unificatore

$((\Phi \,/\, \{A\}) \cup (\Psi \,/\, \{\neg B\}))\, \gamma$ $\qquad$ *risolvente*

/ *differenza di insiemi*

Risoluzione: esempio

$\{P(x, x), Q(x), R(x)\}$ $\{\neg P(A, z), \neg Q(B)\}$

$\{x/A, z/A\}$ $\{x/B\}$

$\{Q(A), R(A), \neg Q(B)\}$ $\{P(B, B), R(B), \neg P(A, z)\}$

Grafo di risoluzione

Problema dei fattori

- Le seguenti clausole dovrebbero produrre la clausola vuota invece …

$$\{P(u), P(v)\} \qquad\qquad \{\neg P(x), \neg P(y)\}$$

$$\{P(v), \neg P(y)\}$$ 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:

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

$$\{\,\}$$

Completezza del metodo di risoluzione

- La deduzione per risoluzione è corretta

  Correttezza: Se $\Gamma \vdash_{\mathrm{RES}} A$ allora $\Gamma \vDash A$

- La deduzione per risoluzione *non* è completa:

  può essere $\Gamma \vDash A$ e non $\Gamma \vdash_{\mathrm{RES}} A$

  Esempio

  $\{\,\} \vDash \{P, \neg P\}$ ma non $\{\,\} \vdash_{\mathrm{RES}} \{P, \neg P\}$

Risoluzione per *refutazione*

- Il *teorema di refutazione* ci suggerisce un metodo alternativo *completo*
- *Teorema di refutazione*:

  $\Gamma \cup \{\neg A\}$ è insoddisfacibile sse $\Gamma \vDash A$
- *Teorema*: Γ 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. $\{P(A, J)\}$ — *A è padre di J*
2. $\{M(B, J)\}$ — *B è madre di J*
3. $\{\neg P(x, y), G(x, y)\}$ — *padre implica genitore*
4. $\{\neg M(v, w), G(v, w)\}$ — *madre implica genitore*

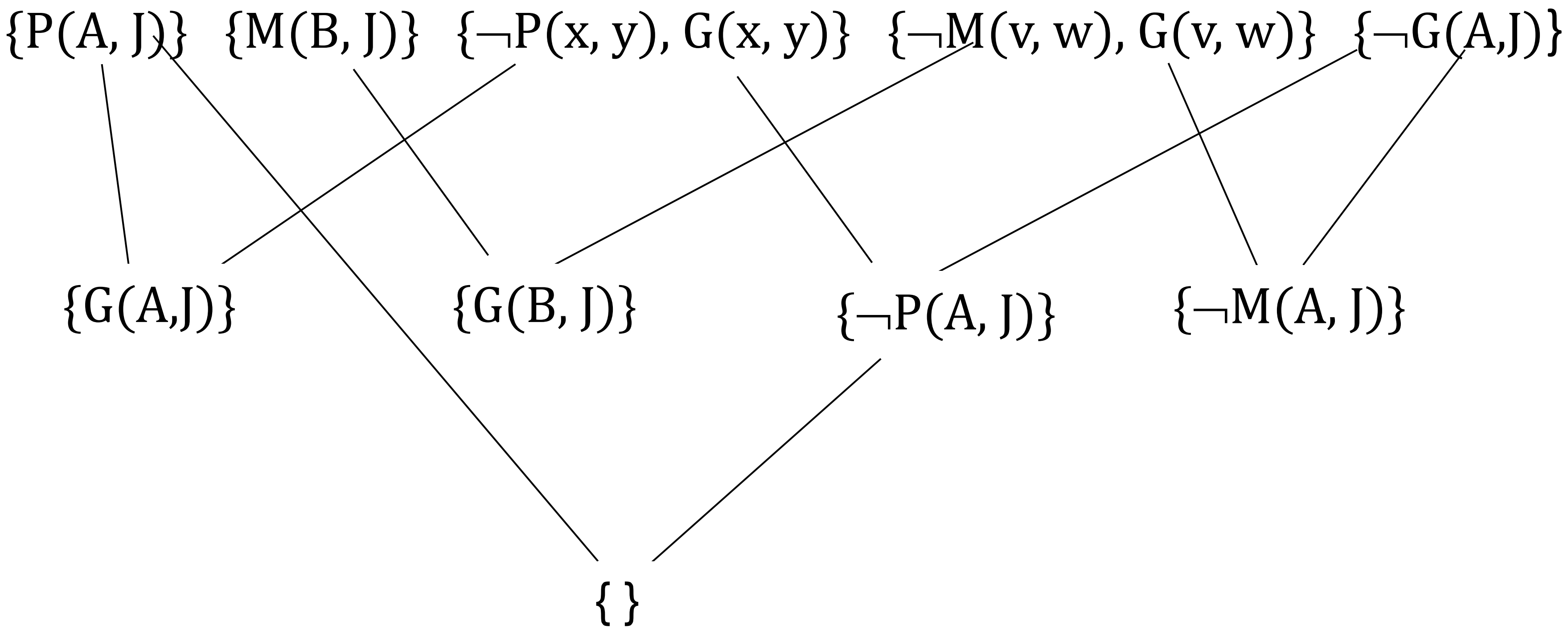
- *Goal*: $G(A, J)$? — *A è genitore di J*?

KB

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

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

Esempio di refutazione: il grafo



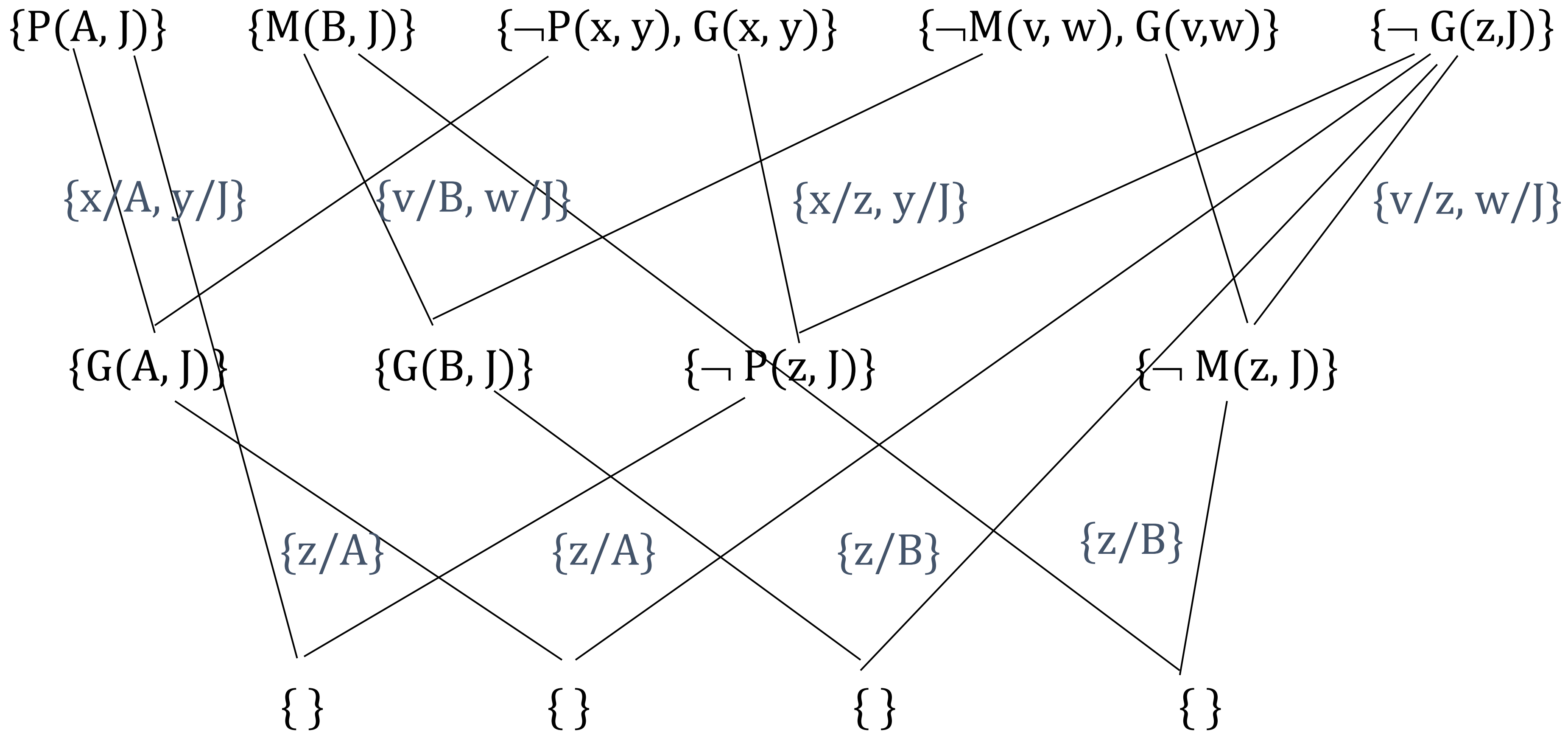
Refutazione per domande di tipo “trova …”

- Esempio: “Chi sono i genitori di J?”
- Si cerca di dimostrare che $\exists z\, G(z, J)$
  1. Si nega la clausola goal: $\neg\exists z\, G(z, J)$
  2. Si porta in forma a clausole:

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

  dove FC sta per “forma a clausole”
- La risposta sono tutti i possibili legami per z che consentono di ottenere la clausola vuota (risposta calcolata)

Esempio: chi sono i genitori di J?



Le risposte sono: A, B

Importante rinominare!

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

$\{\neg P(x, y), G(x, y)\}$ $\qquad$ $\{\neg\, G(y, J)\}$ $\qquad$ y invece che z

$\{x/J, y/J\}$

$\{\neg\, P(J, J)\}$

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

Riferimenti

- AIMA Cap 9: 9.1, 9.2