

Rappresentare la conoscenza serve ad attivare dei **meccanismi di ragionamento automatico** e trarre delle **inferenze** (costruire nuove conoscenze).

Ci appoggeremo su formalismi di tipo **logico**.

1) Tutti i cetacei vivono in mare

2) Tutte le balene sono cetacei

① \wedge ② \Rightarrow Tutte le balene vivono in mare
↓ si dice

Deduzione : fare un'assunzione data per vera da cui si ricava una conclusione / inferenza tramite regola di ragionamento.

modus ponens

Inferire nuova conoscenza = esplicitare della conoscenza che si "nasconde" dentro della conoscenza già appresa

//

Base di conoscenza / **Knowledge Base (KB)**

- Immaginiamo di avere un agente con base di conoscenza iniziale **K_B** (detta **Background Knowledge**)
- L'agente è in grado di **percepire** (incamerare nuova conoscenza \Rightarrow **fatti**) costruendo una conoscenza più ampia (**K_{B1}**)
 \Rightarrow K_B varia nel tempo

Due comandi : • **TELL (ASSERT)** \rightarrow aggiunge elementi di conoscenza all'agente
• **ASK (QUERY)** \rightarrow interroga l'agente

K_B è spesso incompleta e inesatta

E' importante distinguere tra :

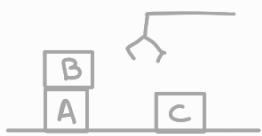
- **dato** = risultato di percezione
- **informazione** = interpretazione dei dati
- **conoscenza** = legami / connessioni tra informazioni di tipo diverso.

KB : rappresentazione in forma dichiarativa

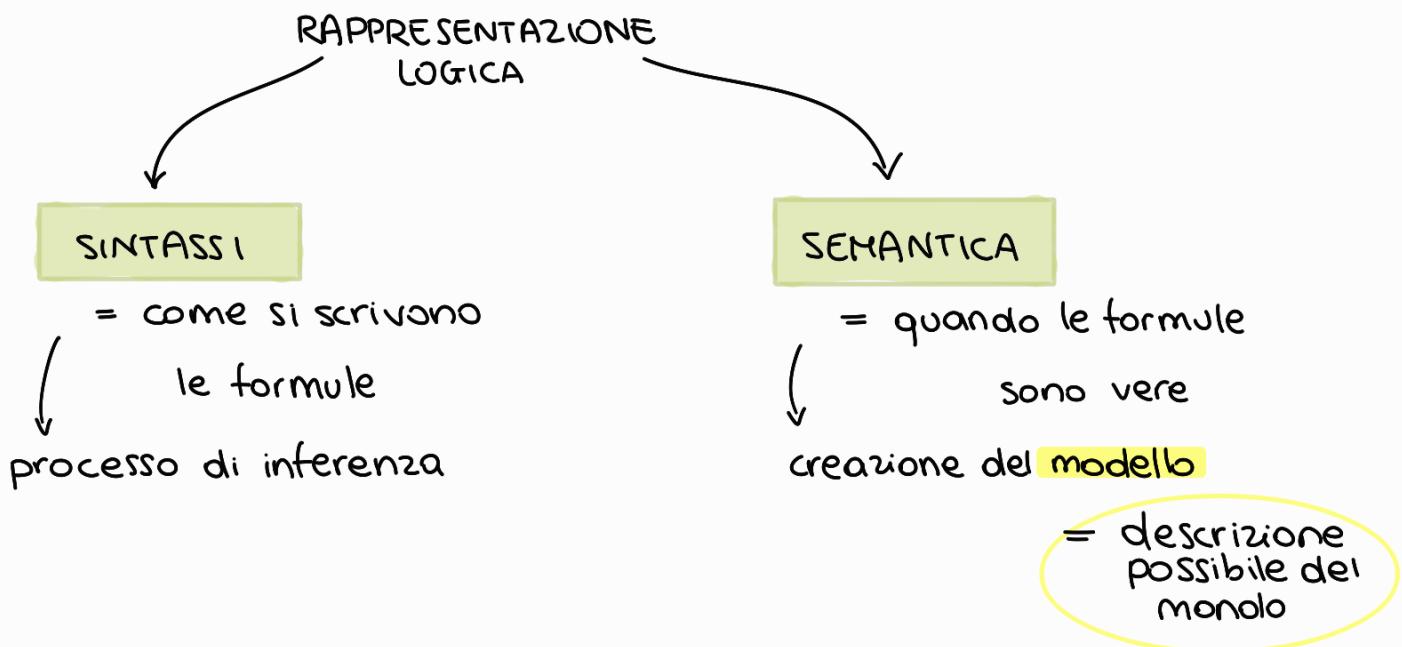


Vi si associa un meccanismo generale di ragionamento

MONDO DEI BLOCCHI :



KB :
ONTABLE(C)
ONTABLE(A)
ON(B,A)
...



Andremo a trattare una conoscenza di tipo simbolico.

I formalismi logici permettono di rappresentare formule, il cui significato (o valore semantico) e' un valore di verita' (vero o falso).

Una formula di per se' non ha associato un valore di verita', ma esso dipende da dei fattori definiti dal modello.

MODELLO = Attribuzione di valori che permette di decidere il valore di verita' di una formula

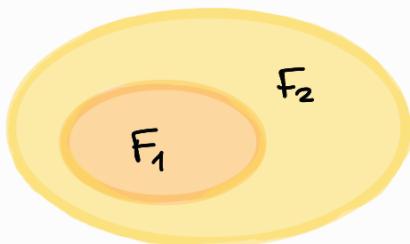
Nel ragionamento automatico non lavoriamo su une formula in isolamento.

CONSEGUENZA LOGICA = relazione tra formule

$F_1 \models F_2 \rightarrow F_2$ e' conseguenza logica di F_1

$F_1 \models F_2$ sse $\forall m$ modello che rende vera F_1 ,
 m e' anche un modello che rende vera F_2

in generale:



Se $F_1 \models F_2 \wedge F_2 \models F_1 \rightarrow F_1 \equiv F_2$

LOGICAMENTE EQUIVALENTI

//

vere negli stessi modelli

Caratteristiche delle formule

- **VALIDITA'** : formula vera in ogni modello = TAUTOLOGIA True
- **INSODDISFACIBILITA'** : falsa in tutti i modelli = CONTRADDIZIONE False
- **SODDISFACIBILITA'** : $\exists m$ che la rende vera

INFERENZA : meccanismo che si basa su regole ben stabilite che, a partire da una formula (o insieme di formule) considerata vera, produce un'altra formula considerata vera.

Lavora sulla forma delle formule (piano sintattico)

MODUS PONENS : è una regola di inferenza

1) Tutti gli uomini sono mortali

2) Socrate è un uomo

① + ② \Rightarrow Socrate è mortale

Si scrive così :
$$\frac{\text{Uomo}(x) \Rightarrow \text{Mortale}(x) \quad \text{Uomo}(\text{Socrate})}{\text{mortale}(\text{Socrate})}$$

\rightarrow
$$\frac{\text{Assunzioni}}{\text{Conclusione}}$$
 \rightarrow

$$\boxed{\frac{A(x) \Rightarrow B(x) \quad A(c)}{B(c)}}$$

Altre regole di inferenza (alcune) :

• ELIMINAZIONE DEI CONGIUNTI :

$$\frac{A \wedge B}{A}$$

• ELIMINAZIONE DEL DOPPIO NEGATO :

$$\frac{\neg\neg A}{A}$$

ALGORITMO DI INFERNZA i

$\text{KB} \vdash_i A$: è possibile derivare la formula A a partire da KB applicando l'algoritmo di inferenza i

Per essere efficace deve godere della proprietà di:

- CORRETTEZZA (soundness) : preserva la verità.

ovvero $\forall A_i \in KB, KB \vdash_i A_i \Rightarrow KB \models_i A_i$

- COMPLETEZZA (completeness) :

$\forall KB \forall A, KB \models A \Rightarrow KB \vdash A$

↳ difficile da dimostrare

LOGICA PROPOZIONALE

→ la più semplice

Grammatica :

FORMULA → atomica o complessa

FORMULA ATOMICA → TRUE / FALSE / simbolo

SIMBOLO → P | Q | R | ...

FORMULA COMPLESSA → . . . Formula |

Formula \wedge Formula |

Formula \vee Formula |

Formula \Rightarrow Formula |

Formula \Leftrightarrow Formula |

PRIORITÀ

KB :

R₁] Piove \Rightarrow Atmosfera-umida

R₂] Notte \wedge \neg Vento \Rightarrow Atmosfera-umida

R₃] Atmosfera-umida \Rightarrow (Prato-bagnato \wedge Strade-bagnate)

R₄] Inneffettore-ON \Rightarrow Prato-bagnato

R₅] Piove \Rightarrow Ombrello-Aperto

R₆] Sole \wedge Vento \Rightarrow Inneffettore-ON

R₇] Sole \wedge Vento \Rightarrow Atmosfera-Asciutta

R₈] Sole \Rightarrow \neg Notte

R₉] Notte \rightarrow \neg Sole

R₁₀] Atmosfera-Asciutta \Rightarrow \neg Atmosfera-umida

Consideriamo : Piove \wedge Vento V o F ?

Come calcolo il valore di verità di una formula complessa
in logica proposizionale ?

Piove $\left\{ \begin{matrix} \text{vero} \\ \text{falso} \end{matrix} \right.$?

Tabelle di verità

A	B	$A \wedge B$
T	F	F
T	T	T
F	T	F
F	F	F

Le tabelle di verità definiscono il valore semantico degli operatori logici, ma non dicono niente senza un modello fissato.

Se ho N simboli proposizionali in KB posso comporre 2^N modelli.

Fissato un modello diventa possibile calcolare la verità delle formule.

⚠️ IMPLICAZIONE $P \Rightarrow Q$

↳ ha varie nature

- ontologica Fido è un cane \Rightarrow Fido è un mammifero
- temporale Aldo ha vinto la partita \Rightarrow Aldo ha giocato la partita
- causale Amelia è stata arrestata per furto \Rightarrow il furto è un crimine

Nelle logiche proposizionale:

$$P \Rightarrow Q \quad (P \rightarrow Q)$$

→ l'implicazione ci "interessa"

quando è vero l'antecedente.

Quando l'antecedente è F

non ci interessa la verità

P	Q	$P \Rightarrow Q$
F	F	T
F	T	T
T	F	F
T	T	T

$$P \Rightarrow Q \equiv \neg P \vee Q$$

Immaginiamo di avere una KB. Vogliamo capire se una formula F e' conseguenza logica delle nostre conoscenze.

$KB_0 \models F ?$

Due strategie:

- MODEL CHECKING → usare la definizione di conseguenza logica e generare tutti i modelli
→ lentissimo
- Ci appoggiamo all'inferenza

TEOREMA DI DEDUZIONE: Date due formule R e Q

$R \models Q$ sse $(R \Rightarrow Q)$ e' valida

Ricordando che se F valida allora $\neg F$ e' contraddizione,
 $R \Rightarrow Q$ valida $\equiv \neg(R \Rightarrow Q)$ contraddizione

!!!

$$\neg(\neg R \wedge Q) \equiv \textcircled{R \wedge \neg Q} \text{ contraddizione}$$

- R e' la KB
 - Q e' la query
(formula che vogliamo dimostrare essere vera)
- \rightarrow se $R \wedge \neg Q$ e' contraddizione allora Q e' vera
- ↳ sto realizzando un ragionamento per assurdo / refutazione

L'algoritmo di inferenza che dimostra l'assurdo puo' essere visto come un algoritmo di ricerca in cui:

- STATO INIZIALE : $KB_0 (R)$
- Funzione successore : regole di inferenza
- Goal : Stato che contiene la formula da dimostrare.

Nel RAGIONAMENTO PER REFUTAZIONE :

lo stato e' lo stato goal se contiene una contraddizione

Ci serve una regola per applicare il ragionamento per refutazione:

REGOLA DI RISOLUZIONE (RESOLUTION)

↳ se aggiunta ad un algoritmo di ricerca completa, produce un algoritmo di inferenza completa.

$$P_i \equiv \neg Q_j$$

$$\frac{P_1 \vee \dots \vee P_i \vee P_{i+1} \vee \dots \vee P_n \quad Q_1 \vee \dots \vee Q_j \vee Q_{j+1} \vee \dots \vee Q_m}{P_1 \vee \dots \vee \underbrace{P_{i-1} \vee P_{i+1} \vee \dots \vee P_n}_{\text{continuo a semplificare}} \vee Q_1 \vee \dots \vee \underbrace{Q_{j-1} \vee Q_{j+1} \vee \dots \vee Q_m}_{\text{RISOLVENTE}}} \rightarrow \text{disgiunzione di letterali}$$

"clausola"

$$P_1 \vee \dots \vee \underbrace{P_{i-1} \vee P_{i+1} \vee \dots \vee P_n}_{\text{continuo a semplificare}} \vee Q_1 \vee \dots \vee \underbrace{Q_{j-1} \vee Q_{j+1} \vee \dots \vee Q_m}_{\text{RISOLVENTE}}$$

↓ continuo a semplificare

Se mi trovo nella situazione :

$$\frac{A \quad \neg A}{\square}$$

→ clausola vuota

→ vuol dire che ho trovato una **contraddizione**



Quando applichiamo la risoluzione, stiamo cercando una situazione che dia come risultato la clausola vuota.

• La risoluzione può essere applicata su KB che sono in **forma normale**

congiuntiva (CNF)

N.B. qualunque KB in logica proposizionale può essere convertita in forma normale congiuntiva

- CNF-Formula → Clausola \wedge Clausola $\wedge \dots \wedge$ Clausola
- Clausola → letterale \vee letterale $\vee \dots \vee$ letterale
- Letterale → Simbolo $|$ \neg Simbolo

Esempi CNF :

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

$$(A \vee B) \wedge (\neg B \vee C \vee D) \wedge (D \vee \neg E)$$

$$A \wedge B$$

Attenzione: alcune sembrano CNF ma non lo sono : $\neg(B \vee C)$

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

Traduzione in CNF :

- 1) eliminare \Leftrightarrow : $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
- 2) eliminare \Rightarrow : $\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$
- 3) de Morgan
- 4) distribuire l' \vee sull' \wedge

KB:

- $R_1] \text{ Piove} \Rightarrow \text{Atmosfera-Umida}$
- $R_2] \neg \text{Notte} \wedge \neg \text{Vento} \Rightarrow \text{Atmosfera-Umida}$
- $R_3] \text{Atmosfera-Umida} \Rightarrow (\text{Prato-Bagnato} \wedge \text{Strade-Bagnate})$
- $R_4] \text{Innefficiatore-ON} \Rightarrow \text{Prato-Bagnato}$
- $R_5] \text{Piove} \Rightarrow \text{Ombrello-Aperto}$
- $R_6] \text{Sole} \wedge \text{Vento} \Rightarrow \text{Innefficiatore-ON}$
- $R_7] \text{Sole} \wedge \text{Vento} \Rightarrow \text{Atmosfera-Asciutta}$
- $R_8] \text{Sole} \Rightarrow \neg \text{Notte}$
- $R_9] \text{Notte} \rightarrow \neg \text{Sole}$
- $R_{10}] \text{Atmosfera-Asciutta} \Rightarrow \neg \text{Atmosfera-Umida}$

Trasformiamo in CNF :

- C₁) $\neg \text{Piove} \vee \text{Atmosfera-Umida}$
- C₂) $\neg \text{Notte} \vee \text{Vento} \vee \text{Atmosfera-Umida}$
- C_{3a}) $\neg \text{Atmosfera-Umida} \vee \text{Prato-Bagnato}$
- C_{3b}) $\neg \text{Atmosfera-Umida} \vee \text{Strade-Bagnate}$
- C₄) $\neg \text{Innefficiatore-ON} \vee \text{Prato-Bagnato}$
- C₅) $\neg \text{Piove} \vee \text{Ombrello-Aperto}$
- C₆) $\neg \text{Sole} \vee \neg \text{Vento} \vee \text{Innefficiatore-ON}$
- C₇) $\neg \text{Sole} \vee \neg \text{Vento} \vee \text{Atmosfera-Asciutta}$
- C₈) $\neg \text{Sole} \vee \neg \text{Notte}$
- C₁₀) $\neg \text{Atmosfera-Asciutta} \vee \neg \text{Atmosfera-Umida}$

Teorema Se un insieme di clausole è insoddisfacibile, la chiusura della risoluzione contiene la clausola vuota

Ci chiediamo se : $\text{KB} + \{\text{sole}, \text{vento}\} \models \neg \text{Piove}$

Applichiamo la risoluzione :

- Goal: $\neg \text{Piove} \rightarrow$ Goal negato : $(\text{GN}) \text{ Piove}$

$\text{GN}) C_1$: $\text{Piove} \quad \neg \text{Piove} \vee \text{Atmosfera_umida}$
 Atmosfera_umida C_{21}

$C_{21}) C_{10}$: $\text{Atmosfera_umida} \quad \neg \text{Atmosfera_asciutta} \vee \neg \text{Atmosfera_umida}$
 $\neg \text{Atmosfera_asciutta}$ C_{22}

$C_{22}) C_7$: $\neg \text{Atmosfera_asciutta} \quad \neg \text{Sole} \vee \neg \text{Vento} \vee \text{Atmosfera_Asciutta}$
 $\neg \text{Sole} \vee \neg \text{Vento}$ C_{23}

$C_{23}) F_1$: $\neg \text{Sole} \vee \neg \text{Vento}$ Sole
 $\neg \text{Vento}$ C_{24}

$C_{24}) F_2$: $\neg \text{Vento}$ Vento
 \square Assurdo

$\Rightarrow \text{KB} \models \neg \text{Piove}$

MODUS PONENS

$$\frac{A \quad A \Rightarrow B}{B}$$

e' un caso particolare della risoluzione,
infatti si puo' riscrivere come:

$$\frac{A \quad \neg A \vee B}{B}$$

Il modus ponens si applica su KB costituite da clausole di Horn

= clausole con al piu' un solo letterale positivo

- $\neg A \vee \neg B \vee \neg C \vee D$ clausola definita
- D fatto
- $\neg A \vee \neg B$ (noi non useremo questa forma)

N.B.

$$\neg A \vee \neg B \vee \neg C \vee D \equiv \neg(A \wedge B \wedge C) \vee D \equiv A \wedge B \wedge C \Rightarrow D$$

Alla base della
programmazione logica

Con le clausole di Horn e' possibile realizzare dei meccanismi di ragionamento
che hanno complessita' temporale lineare nel numero delle clausole

Per spiegare gli algoritmi useremo questo esempio:

KB: $P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$

$\left. \begin{array}{l} P \Rightarrow Q \\ L \wedge M \Rightarrow P \\ B \wedge L \Rightarrow M \\ A \wedge P \Rightarrow L \\ A \wedge B \Rightarrow L \end{array} \right\} \text{clausole di Horn}$

FORWARD CHAINING

- Utilizza il modus ponens
- e' una tecnica di ragionamento guidata dai dati.

KB:

$$\begin{aligned} P &\Rightarrow Q \\ L \wedge M &\Rightarrow P \\ B \wedge L &\Rightarrow M \\ A \wedge P &\Rightarrow L \\ A \wedge B &\Rightarrow L \end{aligned}$$

dati : A, B
goal : Q

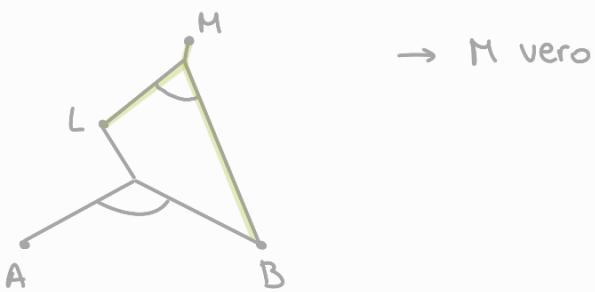
1) Prendo una clausola della KB che e' implicazione con antecedente reso vero dai fatti noti al momento (A, B) \rightarrow prendo $A \wedge B \Rightarrow L$

\rightarrow disegno



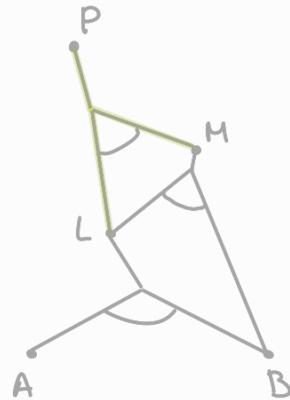
MODUS PONENS : $\frac{A \wedge B \Rightarrow L \quad A \wedge B}{L}$
 $\rightarrow L$ vero

2) $B \wedge L \Rightarrow M$



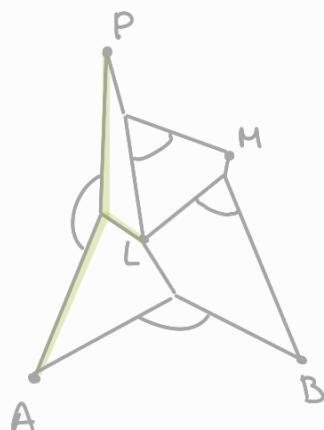
$\rightarrow M$ vero

3) $L \wedge M \Rightarrow P$



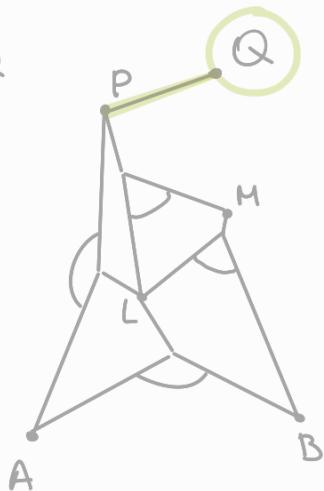
$\rightarrow P$ vero

4) $A \wedge P \Rightarrow L$



non ci dice niente
di nuovo

5) $P \Rightarrow Q$



Verificato il goal



Albero AND-OR

- Si applica iterativamente il modus ponens fino a trovare il goal.
- N.B. e' un meccanismo blind, quindi e' possibile che effettui "mosse" inutili (vedi passo 4)

BACKWARD CHAINING

- utilizza gli stessi elementi del F.C. →
 - KB : clausole di Horn
 - Goal : singolo simbolo proposizionale
 - Fatti : modello
- utilizza il modus ponens "al contrario" → **parte dal goal** e non dai fatti
- Cerca nella KB una clausola di Horn che ha il goal come conseguente.
- Si ripete ricorsivamente per dimostrare che ogni letterale da cui dipende il goal e' un fatto. A questo punto ho trovato il goal.

$$\begin{aligned} \text{KB: } & A \wedge G_1 \Rightarrow G_2 \\ & B \wedge C \Rightarrow G_1 \end{aligned}$$

$$\begin{aligned} \text{Fatti: } & A, B, C \\ \text{Goal: } & G_2 \end{aligned}$$

1) prendo $A \wedge G_1 \Rightarrow G_2$

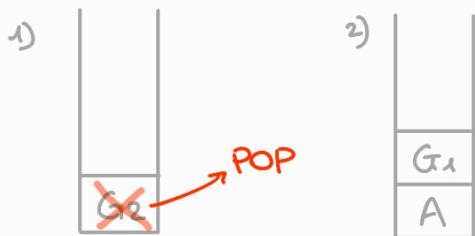
A e' vero (fatto), ma G_1 ?

2) prendo $B \wedge C \Rightarrow G_1$

B, C veri $\rightarrow G_1$ vero

→ ora posso dire che
 $A \wedge G_1$ e' vero
 $\rightarrow G_2$ vero ✓

- Si implementa utilizzando uno **STACK**



- **Vantaggio** rispetto al forward Chaining : di solito **attiva meno clausole**
- **Complessità**: temporale meno che lineare nel numero delle clausole

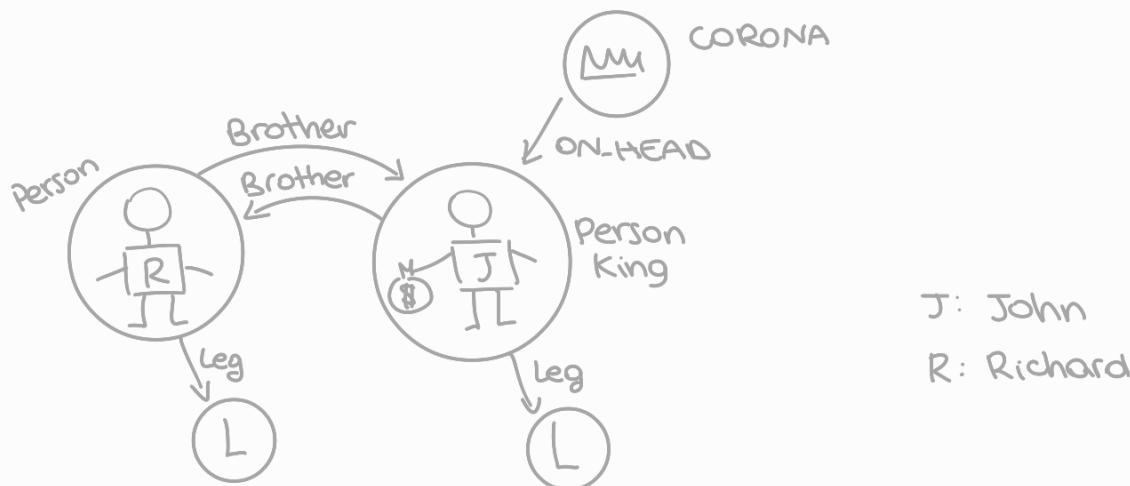
LOGICA DEL PRIM' ORDINE (FOL)

- Permette di ragionare su oggetti e relazioni che appartengono ad un dominio.
- Comprende diversi elementi sintattici, più vari rispetto alla logica proposizionale:
 - simboli di costante
 - simboli di predicato: valgono T o F
 - simboli di funzione: identifica un oggetto del mondo tramite una relazione associata ad una costante o variabile
 - simboli di variabile
 - simboli di quantificazione: \forall \exists
 - connettivi: \neg \wedge \vee \Rightarrow \Leftrightarrow
 - uguaglianza: =
 - punteggiatura: (), ,

MODELLO in FOL : < Dominio, Interpretazione >

tra costanti, predici e funzioni

RELAZIONE (comprende predicati e funzioni) : insieme di tuple



Person(x) : predicato

Leg(x) : funzione

Crown(x) : predicato

King(x) : predicato

Brother(x,y) : funzione

MADRE (Anna, Rosa) = T / F \rightarrow Predicato

MADRE (Rosa) \rightarrow identifica un individuo \rightarrow Funzione

Grammatica FOL

- Formula \rightarrow F. atomica |
 - Formula Connettivo Formula |
 - Quantificatore Variabile, ... Formula |
 - \neg Formula
 - F. atomica \rightarrow Predicato (Termine, ...) |
 - Termine = Termine
 - Termine \rightarrow Funzione (Termine, ...) |
 - Costante |
 - Variabile
 - Connettivi \rightarrow \Rightarrow | \Leftrightarrow | \wedge | \vee
 - Quantificatori \rightarrow \forall | \exists
 - Costante \rightarrow Richard | John | Corona | ...
 - Variabile \rightarrow x | y | ...
 - Predicato \rightarrow Brother | King | ...
 - Funzione \rightarrow Gamba | Madre | ...

Esempi Formule:

- Fratello (Richard, John) \wedge Re (John)
 - $\exists x$ MADRE(x, Richard) \vee MADRE(x, John)

Le variabili sono sempre quantificate

 - $\forall x \exists y$ AMA(y, x)

Interpretazione = associazione tra · costante - oggetto del dominio

- predicato - relazione del dominio
 - funzione - relazione del dominio

Se cambio in modo coerente le etichette, non cambio il valore di verità delle formule.

MODEL CHECKING FOL

Nella logica del prim'ordine, il numero dei modelli puo' essere infinito.

→ enumerare i modelli puo' essere impossibile !

→ faremo uso dell'inferenza

Cerchiamo di capire meglio cosa sono i termini:

Termine → Funzione (Termine, ...)

Costante |

Variabile

- Termine ground : non contiene variabili

↳ devono essere interpretati

- Predicato "=" fra due termini

$$\frac{\text{Padre(Richard)} = \text{Padre(John)}}{\text{O} \quad \text{O}} \quad \text{permette di catturare degli alias}$$

Quantificatori

- Universale : \forall

- Tutti gli uomini : $\forall x, \text{Uomo}(x)$

- $\forall x \text{ PARTECIPA}(x, \text{SISINT}) \Rightarrow \text{INTELLIGENTE}(x)$

= Tutti quelli che partecipano al corso SisInt sono intelligenti

- $\forall x \text{ PARTECIPA}(x, \text{SISINT}) \wedge \text{INTELLIGENTE}(x)$

= Tutti partecipano a SisInt e sono intelligenti

- Esistenziale : \exists

- $\exists x \text{ PARTECIPA}(x, \text{SISINT}) \wedge \text{INTELLIGENTE}(x)$

= $\text{PARTECIPA}(\text{Marco}, \text{SISINT}) \wedge \text{INT.}(\text{Marco})$

↓
 $\text{PARTECIPA}(\text{Anna}, \text{SISINT}) \wedge \text{INT.}(\text{Anna})$

↓

...

Rapporto tra \forall e \exists

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

$$\forall x \neg Ama(x, Cavolfiore) \equiv \neg \exists x Ama(x, Cavolfiore)$$

: nessuno ama il cavolfiore

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

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

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

John ha almeno due fratelli :

$$\exists y, z \text{ Fratello(John, } y) \wedge \text{Fratello(John, } z) \quad \times \text{ potrei legare } y \text{ e } z \text{ allo stesso individuo!}$$

$$\rightarrow \exists y, z \text{ Fratello(John, } y) \wedge \text{Fratello(John, } z) \wedge \underline{\neg(y=z)} \quad \checkmark$$

↓
devo specificarlo
anche con le costanti

Per usare dei formalismi logici per fare programmazione logica vengono poste delle restrizioni :

- unicità dei nomi
- closed world assumption : ogni fatto di cui non conosciamo la verità è assunto falso
- domain closure : si assume che non vi siano più elementi rispetto a quelli nominati dalle costanti
 - Mondo circoscritto da individui nominati dalle costanti
 - rende il dominio finito

Non facciamo programmazione logica

Come viene gestita l'inferenza in FOL ?

Due famiglie di approcci :

1) PROPOZIONALIZZAZIONE

2) LIFTING delle regole di inferenza (da LP a FOL)

Fare ragionamento in FOL serve a interrogare un agente dotato di una qualche base di conoscenza e un algoritmo di inferenza

• ASK(RE(John)) \rightarrow T / F

• ASK(RE(x)) \rightarrow F se \nexists

\rightarrow se $\exists x$ t.c. RE(x) : return un termine ground

che, sostituito a x, rende

vera la query

si scrive

{x / John} (ad esempio)

ASK(F(x,y,...))

dai un risultato come $\{x | v_1, y | v_2, \dots\}$ oppure False

SOSTITUZIONE

Le sostituzioni generalmente sono rappresentate con la lettera θ (theta)

PROPOZIONALIZZAZIONE

• Prendere la KB in FOL e provare a trasformarla in KB in LP.

• La traduzione mi farà perdere delle informazioni !

D togliere variabili e quantificatori : var \rightarrow termine ground

$\forall \exists \rightarrow$ eliminati

come ?

UNIVERSAL INSTANTIATION (UI)

e' una regola di inferenza :
da una formula quantificata universalmente e' possibile inferire tutte le formule ottenute sostituendo a x un termine ground

$$\frac{\forall x \alpha(x)}{\text{SUBST}(\{x/g\}, \alpha(x))}$$

formula in FOL

Sostituzione termine ground

$$\forall x \text{ Persona}(x)$$



$$\text{Persona(John)}$$

$$\text{Persona(Richard)}$$

Madre(x) funzione

- $\forall x \text{ Persona}(x) \rightarrow \text{Persona(John)}$
 - $\text{Persone(Madre(John))}$
 - $\text{Persone(Madre(Madre(John)))}$
- : continua fino al termine che mi serve

Usando UI non perdo informazioni :

$$KB_{FOL} \equiv KB_{LP}$$

EXISTENTIAL INSTANTIATION (EI)

$$\frac{\exists x \alpha(x)}{\text{SUBST}(\{x/k\}, \alpha(x))}$$

possiamo costruire una possibile sostituzione che rende vero $\alpha(x)$

- k : nuova costante \rightarrow COSTANTE DI SKOLEM

$$\frac{\exists x \text{ corona}(x) \wedge \text{sulla-testa}(x, \text{John})}{\text{Corona}(k) \wedge \text{sulla-testa}(k, \text{John})}$$

rende l'idea che non ogni singolo oggetto e' identificato da un nome.

- Non e' possibile derivare tutto cio' che e' derivabile

\rightarrow con EI : $KB_{FOL} \not\equiv KB_{LP}$ \rightarrow perdo conoscenza

- KB_{LP} e' soddisfacibile se KB_{FOL} lo e'

LIFTING \rightarrow trasformazione di regole di inferenza

- MODUS PONENS \rightarrow MODUS PONENS GENERALIZZATO

$$\frac{P_1', P_2', \dots, P_n' \quad P_1 \wedge P_2 \wedge \dots \wedge P_n \Rightarrow q}{q^\theta}$$

con q : atomica

P_i, P_i' : atomici

SOSTITUZIONE UNIFICATORE : $\forall i \in [1, n] \quad P_i' \theta = P_i \theta$

$$\underline{Re(John), \forall y Avido(y)}$$

$$Re(x) \wedge Avido(x) \Rightarrow Malvagio(x)$$

$$Malvagio(John)$$

θ ha reso $Re(John) = Re(x)$ e $Avido(y) = Avido(x)$

$$\rightarrow \theta = \{x / John, y / John\}$$

Il modus ponens generalizzato puo' lavorare solo su clausole di Horn FOL

||

disgiunzione di letterali di cui al piu' uno e' positivo

- P (costante)
- $\forall x P(x)$
- $Re(x) \wedge Avido(x) \Rightarrow Malvagio(x)$
 $\equiv \neg Re \vee \neg Avido(x) \vee Malvagio(x)$

Esempio

- 1) La legge dice che è un reato per un negozio vendere alcolici a un minorenne.
- 2) Marco possiede della birra.
- 3) Tale birra gli è stata venduta del minimarket Sotto Casa.

Costanti : Marco , Sottocasa

Predicati : vende , negozio , minimarket , birra , alcolico , minorenne , possiede , reo

- Minorenne (Marco)

Funzioni : -

- Minimarket (Sottocasa)

- 1) Negozio(x) \wedge Vende(x,y,z) \wedge Alcolico(y) \wedge Minorenne(z) \Rightarrow Reo(x)
- 2) $\exists x \text{ birra}(x) \wedge \text{possiede}(\text{Marco}, x)$ \rightarrow NO Horn
 → applico EI. B costante di skolem
 $\text{Birra}(B) \wedge \text{Possiede}(\text{Marco}, B)$ \rightarrow NO Horn
 → lo divido in due fatti : Birra(B) , Possiede(Marco, B)
- 3) Possiede(Marco, x) \wedge Birra(x) \Rightarrow Vende(Sottocasa, x, Marco)

Mancano l'informazione :

- Birra(x) \Rightarrow Alcolico(x)
- Minimarket(x) \Rightarrow Negozio(x)

} CONOSCENZA ONTOLOGICA

KB :

- C₁) Negozio(x) \wedge Vende(x,y,z) \wedge Alcolico(y) \wedge Minorenne(z) \Rightarrow Reo(x)
- C₂) Possiede(Marco, B)
- C₃) Birra(B)
- C₄) Possiede(Marco, x) \wedge Birra(x) \Rightarrow Vende(Sottocasa, x, Marco)
- C₅) Birra(x) \Rightarrow Alcolico(x)
- C₆) Minimarket(Sottocasa)
- C₇) Minimarket(x) \Rightarrow Negozio(x)
- C₈) Minorenne(Marco)

DATALOG : KB costituite da clausole di Horn e senza funzioni

Deriviamo nuova conoscenza con modus ponens generalizzato:

sostituzione

$$\frac{\text{Birra (B)} \quad \text{Birra}(x) \Rightarrow \text{Alcolico}(x)}{\text{Alcolico}(\underline{\text{B}})}$$

ragionamento per deduzione

Forward Chaining FOL

- guidato dai dati
- cerca clausole di Horn in cui l'antecedente è reso vero dai fatti
- deriva il conseguente → lo aggiunge a KB tramite sostituzione
- finalizzato a derivare un goal

- C₂) Possiede (Marco, B)
 C₃) Birra (B)
 C₄) Possiede (Marco, x) ∧ Birra(x) → Vende (Sottocasa, x, Marco)

$$\frac{\text{Birra (B)} \quad \text{Possiede}(\text{Marco}, \underline{\text{B}})}{\Theta = \{x / \text{B}\}}$$

$$\frac{\text{Possiede}(\text{Marco}, x) \wedge \text{Birra}(x)}{\rightarrow \text{Vende}(\text{Sottocasa}, x, \text{Marco})}$$

UNIFICATORE

$$\underline{\text{Vende}(\text{Sottocasa}, \text{Birra}, \text{Marco})}$$

- C₅) Birra(x) ⇒ Alcolico(x)

$$C_3) \text{ Birra (B)}$$

$$\frac{\text{Birra (B)}}{\Theta = \{x / \text{B}\}}$$

$$\text{Birra}(x) \Rightarrow \text{Alcolico}(x)$$

$$\underline{\text{Alcolico (B)}}$$

- C₇) Minimarket(x) ⇒ Negozio(x)

$$C_6) \text{ Minimarket (Sottocasa)}$$

$$\rightarrow \underline{\text{Negozio (SottoCasa)}}$$

- C₁) Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)

Minorenne (Marco)

Vende (Sottocasa, Birra, Marco)

Alcolico (B)

Negozio (Sottocasa)

C₁)

Reo (SottoCasa)

$\Theta = \{x / \text{SottoCasa}, y / \text{B}, z / \text{Marco}\}$

UNIFICATORE

Proprietà del Forward Chaining :

- Corretto (se $KB \vdash F$ allora $KB \models F$)
- Completo (se $KB \models F$ allora $KB \vdash F$)
 - se KB DATALOG, sì
 - altrimenti può non terminare

Backward Chaining FOL

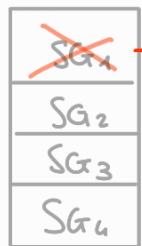
- dal goal verso i fatti
- Si possono dimostrare sottoclausole atomiche e ground
 $\text{Reo}(\text{sottocasa}) ?$

c1) con $\theta \{x / \text{sottocasa}\}$

$$\hookrightarrow \underline{\text{Negozio}(\text{sottocasa})} \wedge \underline{\text{Vende}(\text{sottocasa}, y, z)} \wedge \underline{\text{Alcolico}(y)} \wedge \underline{\text{Minorenne}(z)}$$

$SG_4 \qquad \qquad \qquad SG_3 \qquad \qquad \qquad SG_1$

Inserisco in uno stack



SG_1 nuovo goal



con $G_2 : \theta \{z / \text{Marco}\}$

ottengo $\text{Minorenne}(\text{Marco})$

etc...

Continuo finché stack vuoto o ho esaurito le formule

- Corretto
- Incompleto

LIFTING: REGOLA DI RISOLUZIONE

$$L.P. : \frac{A \vee \cancel{B} \vee C \quad D \vee \cancel{B} \vee A}{A \vee C \vee D}$$

FOL : i disgiunti sono predicati

- applicabile solo su KB CNF

$$\frac{l_1 \vee l_2 \vee \dots \vee \cancel{l_i} \vee \dots \vee l_k \quad m_1 \vee \dots \vee \cancel{m_j} \vee \dots \vee m_n}{\text{SUBST}(\emptyset, l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)}$$

*trovata una sostituzione
che unifica l_i e $\neg m_j$*

- Passare da KB_{FOL} a CNF

- eliminare \Leftrightarrow
- eliminare \Rightarrow
- eliminare quantificatori

$$\overbrace{\forall x [\forall y \text{Animale}(y) \Rightarrow \text{Ama}(x,y)]}^{F_1} \Rightarrow [\exists y \text{Ama}(y,x)] \quad \text{(tutti coloro che amano gli animali sono amati da qualcuno)}$$

$$F_1 \Rightarrow F_2 \equiv \neg F_1 \vee F_2$$

$$\forall x \neg [\underbrace{\forall y \text{Animale}(y)}_{F_1} \Rightarrow \underbrace{\text{Ama}(x,y)}_{F_2}] \vee [\exists y \text{Ama}(y,x)]$$

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

$$\neg \forall y F(y) \equiv \exists y \neg F(y)$$

$$\forall x [\exists y \text{Animale}(y) \wedge \neg \text{Ama}(x,y)] \vee [\exists \underset{\text{sono y diverse}}{y} \text{Ama}(\underset{z}{y},x)]$$

SKOLEMIZZAZIONE :

SKOLEMIZZAZIONE :

$$\begin{aligned} \forall x_1, x_2 \dots [\exists y P(x_1, x_2, \dots, y) \dots \exists z Q(x_1, x_2, \dots, z)] \\ \equiv \\ \forall x_1, x_2, \dots P(x_1, x_2, \dots [S_1(x_1, x_2, \dots)]) \dots Q(x_1, x_2, \dots [S_2(x_1, \dots)]) \end{aligned}$$

Funzione di Skolem

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

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

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

CNF ✓

Sopravvive solo le variabili quantificate universalmente

Vedi 02.1 Curiosity killed The Cat

La risoluzione e' refutation complete :

se KB e' insoddisfacibile allora l'algoritmo derivera' una contraddizione.

La FOL e' semidecidibile : alcune dimostrazioni potrebbero essere infinite

Negazione per fallimento : se non si riesce a dimostrare un fatto, questo e' dichiarato come falso

L'occhio umano guarda il mondo effettuando continuamente un'operazione di **categorizzazione**.

In questo contesto : **CATEGORIA = CONCETTO = CLASSE**

Gli esseri umani costruiscono e raffinano inconsciamente una **concettualizzazione** (= rete di concetti).

La conoscenza umana (simbolica) è molto concentrata su **insiemi di concetti** costruiti con l'esperienza.



così importante che si rischia di cadere nella **preconcettualizzazione** (\equiv mi figuro delle strutture di una realtà che ancora non conosco sulla base di concetti che sono nella mia esperienza)

→ **Concetto** : strumento per categorizzare oggetti

Concetto di PALLONE :

• proviamo a ragionare in FOL

(predicato \rightarrow PALLONE(x) = V/F
funzione \rightarrow PALLONE(x) con x proprietario
Non molto comodi

Si utilizzano **logiche descrittive**

RAPPRESENTAZIONE ONTOLOGICA

- Reificare gli oggetti = immagino un oggetto come oggetto
posso scrivere il termine PALLONE per indicare il concetto di PALLONE
→ Non e' ne' un predicato ne' una funzione



Uso dell'ontologia: particolarmente indicato per catturare una conoscenza abbastanza vicina a quella umana perché essa e'

- incompleta
- soggetta a revisione
- inconsistente (non tanto)

- La conoscenza umana non e' mai completa:
la rete di conoscenze cambia e si arricchisce con l'esperienza

IS-A Relazione tra due concetti che indica che uno e' sotto-concetto dell'altro
IS-A (PALLONE, PALLONE-RUGBY)
e' un predicato Standard che non dipende dal dominio trattato

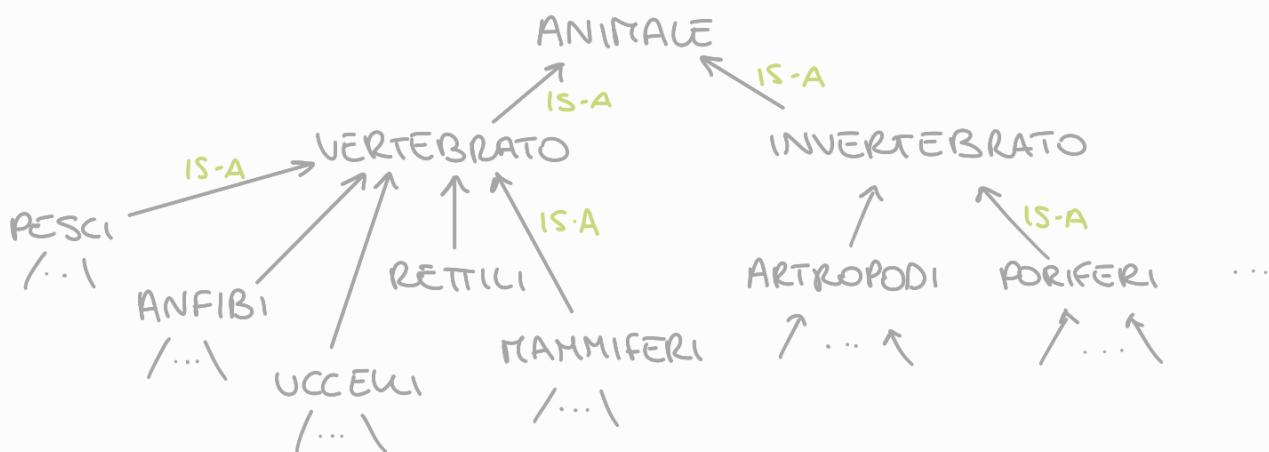
MEMBER Predicato che lega i concetti agli oggetti del mondo:
istanze



PART-OF Predicato che lega due categorie di cui una e' parte dell'altra

PART-OF (PETALO, FIORE)
PART-OF (MOTORE, AUTOMOBILE)

Con i primi due predicati standard posso costruire **Tassonomie**
= concettualizzazioni che hanno forma di albero



L'albero mi crea dei concetti che possono essere più generali (verso l'alto)
o più specifici (verso il basso).

Più scendo, più aggiungo proprietà alla descrizione.
→ descrizione in maniera incrementale
→ ogni concetto eredita le proprietà delle sottocategorie

MEMBER(x, PALLONE) \Rightarrow SFERICO(x) **PROPRIETÀ**

MEMBER(P123, PALLONE-BASKET)
IS-A(PALLONE-BASKET, PALLONE)

} derivo : SFERICO(P123)

MA ad esempio i palloni de Rugby non sono sferici

→ Ad una sottocategoria posso associare la cancellazione di una proprietà

Le tassonomie sono caso particolare di ontologie.

In generale: ontologia
• grafo
• reti semantiche

Esistono tre caratterizzazioni

→ categorie:

- **DISGIUNTE** - riguarda un insieme di categorie
- **COSTITUISCONO UNA DECOMPOSIZIONE ESAUSTIVA**
 - riguarda concetto e insieme di sotto-concetti
- **COSTITUISCONO UNA PARTIZIONE** - riguarda concetto e insieme di sotto-concetti



Sia C : concetto e $S = \{x_1, \dots, x_n\}$

$\text{DISJOINT}(S) \Leftrightarrow \forall x_i, x_j \in S \quad x_i \neq x_j \Rightarrow \text{INTERSEZIONE}(x_i, x_j) = \{\}$

DECOMPOSIZIONE_ESAUSTIVA(S, C)

$\Leftrightarrow \forall i \text{ MEMBER}(i, C) \Leftrightarrow \exists x \in S, \text{IS-A}(x, C) \wedge \text{MEMBER}(i, C)$

PARTIZIONE(S, C) $\Leftrightarrow \text{DISJOINT}(S, C) \wedge \text{DECOMP-ESAUSTIVA}(S, C)$

Ogni concetto rappresenta un insieme di istanze

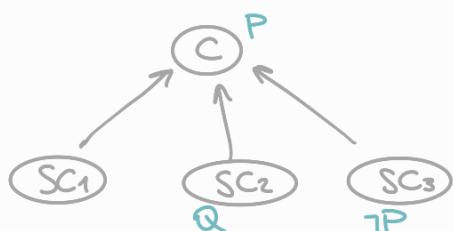
Ragionare su sottoclassi è come ragionare su operazioni insiemistiche.

Rappresentare della conoscenza nel nostro contesto significa:

- rappresentare della **conoscenza generale** (formule) = **T-box** (theoretical)
- rappresentazione della conoscenza fattuale (istanze) = **A-box** (actual)

Posso immaginare una classe C per cui vale la proprietà P

So che C è divisa in sottoclassi, per cui valgono altre proprietà.



-
- SC_1 eredita P
 - SC_2 eredita P ed ha in più Q
 - SC_3 cancella P

Nella rappresentazione ontologica **concetto ≠ parola / etichetta**

- le concettualizzazioni devono prescindere da possibili ambiguità e devono essere astratte rispetto alla lingua parlata
- i linguaggi ontologici permettono di associare ad una stessa etichetta una serie di termini in lingue diverse.

Una volta scritta l'ontologia, essa viene usata da un **motore inferenziale**, cioè uno strumento SW che, date delle interrogazioni, astrae fatti da un dominio di riferimento (comprende più ontologie).

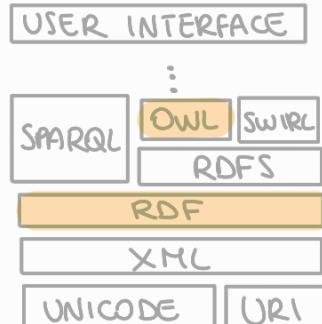
Le ontologie vengono anche usate come interfaccia in sistemi più complessi:

- comprendere un linguaggio naturale e legarlo ad una KB;
- descrivere immagini;
- ...

Un'altra applicazione è il **semantic web**:

- transizione da WWW ad una visione in cui le pagine sono arricchite con informazioni semantiche (= annotazioni ontologiche) che permettono l'interrogazione e l'elaborazione tramite meccanismi di ragionamento automatico.
- la standardizzazione dei linguaggi per il semantic web fa capo al consorzio W3C.
- in generale si fa riferimento alla visione architettonica che si chiama **Semantic Web Tower**

Noi parleremo di **RDF** e **OWL**

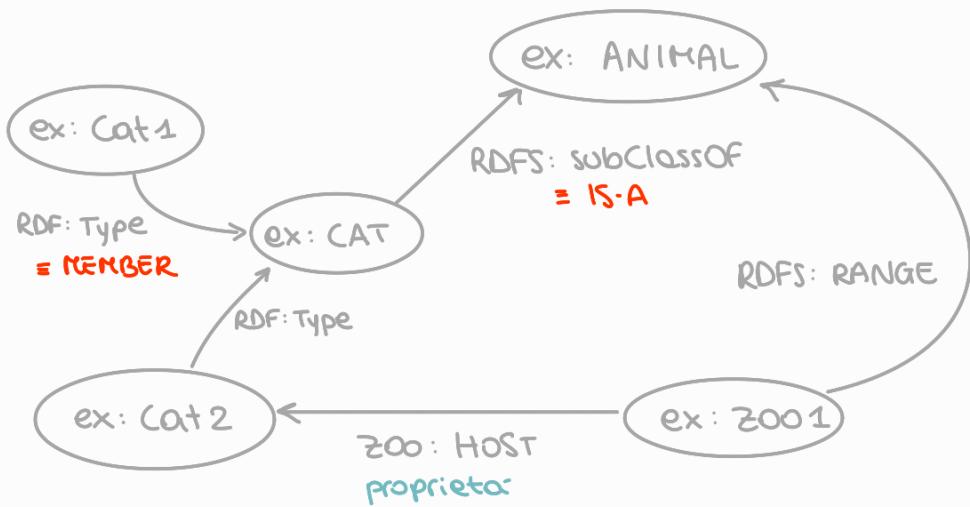


RDF (Resource Description Framework)

- usato sia come linguaggio che come modello rappresentativo.
- KB = insieme di statement (= <soggetto, predicato, oggetto>)



- Costruito in XML



Linguaggio OWL: più generale di RDF

Passi metodologici per costruire un'ontologia

Immaginiamo di avere un dominio applicativo in cui viene utilizzato un DB con un sistema SW tradizionale.

Immaginiamo di voler passare ad un sistema SW arricchito in cui si fa uso di ontologie.

Immaginiamo di dover gestire una rete di canili.

1) Identificare i concetti

- consultare il DB ed elencare tutti i concetti che si incontrano insieme ad una breve descrizione
= sostantivi

CANE : animale ospite
BASSOTTO : tipo di cane
CUCCIOLA : cane giovane
METICCIO ...
...

VOCABOLARIO

2) Identificare le relazioni = verbi / proprietà

OSPITARE
ADOTTARE
VACCINATO
COLORE
TAGLIA
:

3) Costruire tramite un qualche editor → T-box

4) Costruire la A-box andando a convertire i dati che sono nel DB in fatti in rappresentazione ontologica.

5) Agganciare l'ontologia al sito di riferimento

Un importante problema : Allineamento Ontologico

Azienda1 AUTOMOTIVE
↳ Ontologia O1

Azienda2 AUTOMOTIVE
↳ Ontologia O2

E' altamente probabile che $O_1 \neq O_2$

Supponiamo che le due aziende si fondono

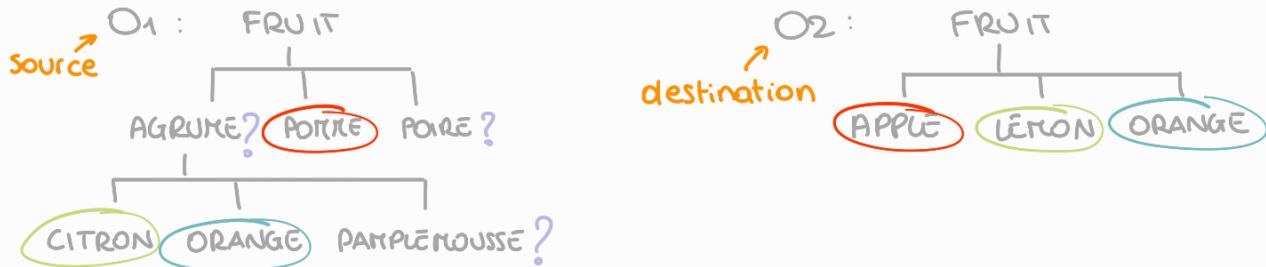
→ Si cercano delle corrispondenze tra le due ontologie

allineamenti = relazioni tra due ontologie

- identiche : due copie dello stesso file
- equivalenti : stessi concetti, stesse assiomatizzazioni, linguaggi diversi
- estensione : O_1 estende O_2 quando tutti i simboli e le assiomatizzazioni di O_2 sono preservati da O_1
MA non vale il viceversa

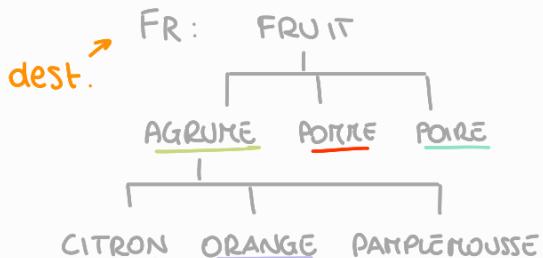
- WEAKLY-TRANSLATABLE

Siano O_1 in FR e O_2 in EN



possiamo fare il mapping dei concetti che sono contenuti nell'ontologia sorgente in quella destinazione , ma con perdite di informazione (da ont. più ricca a più povera)

- STRONGLY TRANSLATABLE



Nel mapping non c'e' perdita di informazione e in piu' la destinazione e' più ricca.

- APPROX TRANSLATABLE

corrisponde ad essere W.T. ma con qualche inconsistenza