Università di Parma
Dipartimento di Ingegneria e Architettura

Introduzione all'Intelligenza Artificiale

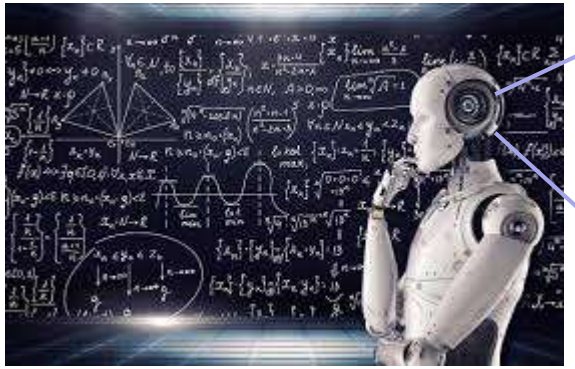
Big Data & Business Intelligence

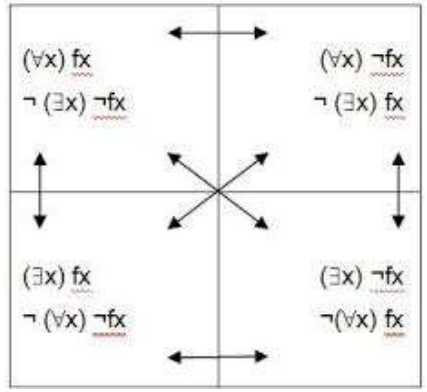
A.A. 2022/2023

Corso di «Introduzione all'Intelligenza Artificiale»
Corso di «Big Data & Business Intelligence»

Reasoning and Planning

Monica Mordonini (monica.mordonini@unipr.it)





AGENTE LOGICO
LA LOGICA DEI PREDICATI

Limiti della logica proposizionale

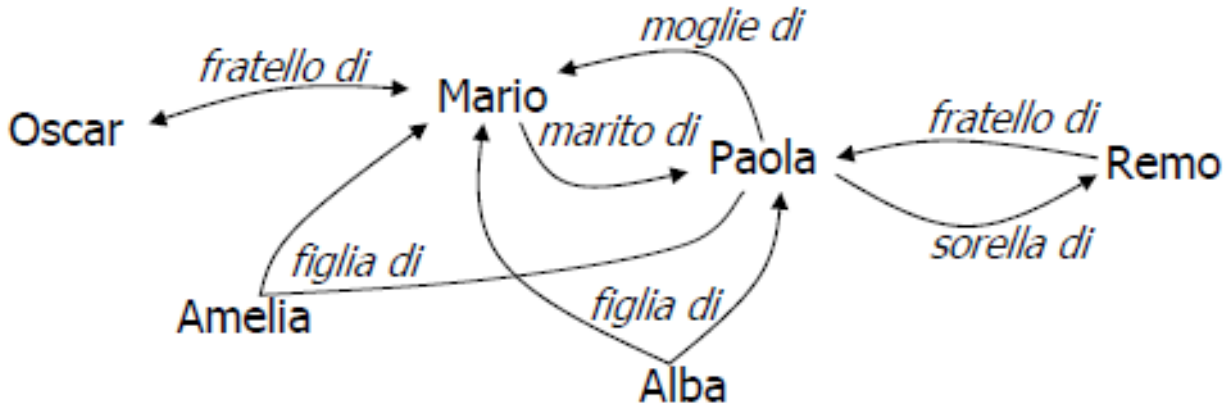
- La logica proposizionale ha molte interessanti proprietà:
  - è completa (tutte le conseguenze logiche sono derivabili per via sintattica e viceversa)
  - è decidibile in modo automatico (al massimo verifica di tutte le combinazioni)

Limiti della logica proposizionale

- Il difetto principale è la semplicità del linguaggio:
  - non è possibile rappresentare la struttura interna delle affermazioni (solo V o F)
  - e quindi mettere in evidenza legami logici più sottili
- e la conseguente semplicità delle strutture semantiche:
  - solo un insieme {0, 1}
  - nessuna possibilità di caratterizzare strutture più complesse

Limiti della logica proposizionale

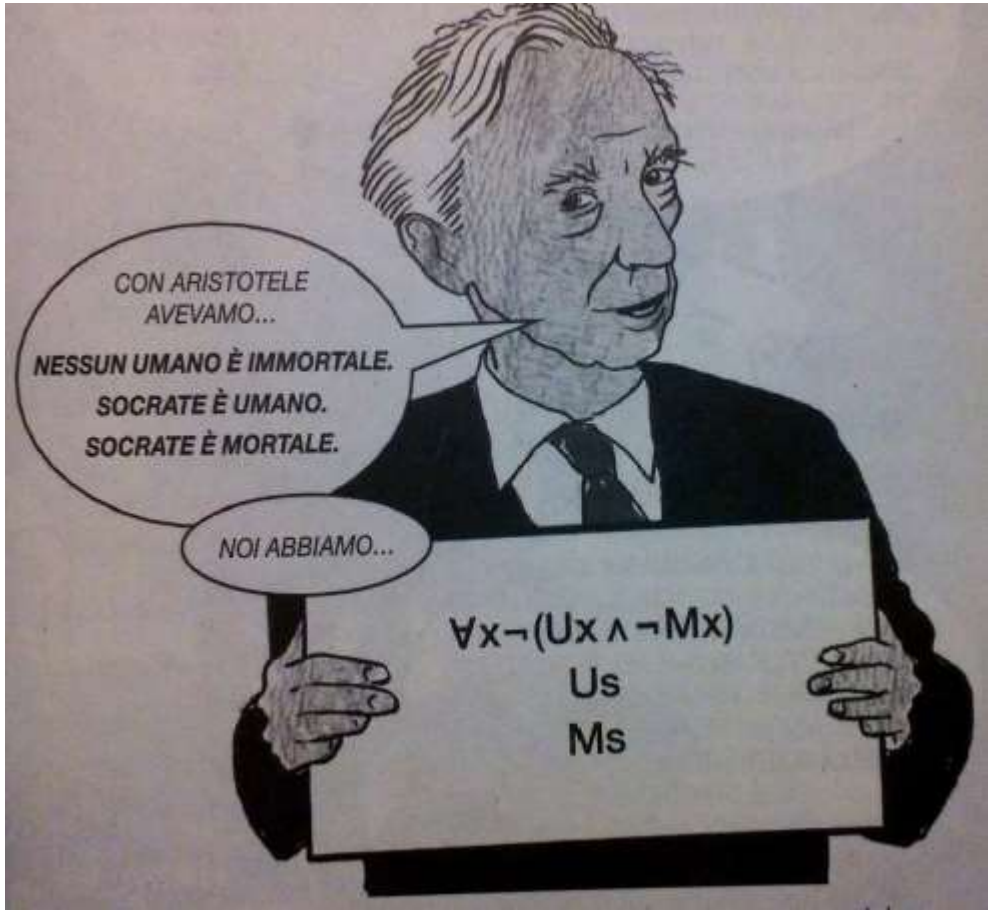
- Dal punto di vista della rappresentazione:
  - il mondo che osserviamo è fatto di oggetti e di relazioni tra oggetti



  - come si possono tradurre questi elementi in forma simbolica?
  - come si stabilisce la correttezza dei ragionamenti in questi casi?
    - p.es. Amelia e Alba sono sorelle?

Limiti della logica proposizionale e calcolo dei predicati

- Il problema è stato affrontato verso la fine dell'800 ed inzio '900
- Russell reintroduce la distinzione aristotelica tra soggetto e predicato nel suo ***calcolo dei predicati***
- Per fare questo introduce i quantificatori esistenziali e una versione estesa delle regole di inferenza della logica proposizionale
- Così si può dimostrare il sillogismo di Aristotele che nella logica proposizionale è visto come una unità atomica.
- Rappresenta le basi per la "Logica del primo ordine" o calcolo dei predicati formulato in modo definitivo negli anni '30 del 900



Limiti della logica proposizionale

- Ma non si riesce a spiegare tutto: il paradosso del barbiere
  - «In un villaggio vi è un solo barbiere, un uomo ben sbarbato, che rade tutti e solo gli uomini del villaggio che non si radono da soli. La domanda è: il barbiere si fa la barba da solo?»
  - Cioè appartiene alla categoria degli uomini che si radono da soli o a quella che, non radendosi da soli, vengono rasati dal barbiere ?

- Sia che venisse incluso nella prima, sia che venisse incluso nella seconda, la situazione sarebbe contraddittoria => Antinomia
- *Antinomia*: una proposizione che risulta autocontraddittoria sia nel caso che sia vera, sia nel caso che sia falsa
- *Paradosso*: una conclusione logica e non contraddittoria che si scontra con il nostro modo abituale di vedere le cose

Limiti della logica proposizionale e calcolo dei predicati

- E non è l'unico problema…
- La visione moderna del linguaggio naturale è che serve come mezzo di comunicazione piuttosto che come pura rappresentazione
- il significato della frase dipende sia dalla frase stessa che dal contesto in cui la frase è stata pronunciata. "Stai attento " ha senso solo se relazionata al contesto… come lo rappresentiamo?
- Inoltre le lingue naturali soffrono di ambiguità, un problema per la matematica e la logica

Limiti della logica proposizionale e calcolo dei predicati

- Nonostante questo si è cercato di implementare con la logica del primo ordine una logica che precisa ed adatta al calcolo automaticoma che sia in grado di rappresentare idee e concetti del linguaggio natural
  - ***Con dei limiti***

Logica del primo ordine

- Mentre la logica proposizionale presuppone che il mondo contenga fatti, la logica del primo ordine (come il linguaggio naturale) presuppone che il mondo contenga oggetti e relazioni
- Oggetti: persone, case, numeri, teorie, Ronald McDonald, colori, partite di baseball, guerre, secoli…
- Relazioni: rosso, rotondo, fasullo, primo, a più piani …, fratello di, più grande di, dentro, parte di, ha colore, si è verificato dopo, possiede si trova in mezzo, …
- Funzioni: padre di, migliore amico, terzo inning di, uno in più, fine di...

Logica del primo ordine

La logica dei predicati permette di rappresentare:

- Oggetti: persone, cose, numeri etc.
- Relazioni: fratello, maggiore, parte di etc.
- Proprietà: rosso, primo, grande etc.
- Funzioni: successore, somma, padre di etc.

Logica del primo ordine

Esempi:

- “Uno più due uguale tre”
  - uno,due,tre sono oggetti, più è una funzione, uguale è una relazione ("Uno più due" è un nome per l'oggetto che si ottiene applicando la funzione "più" agli oggetti "uno" e "due". "Tre" è un altro nome per questo oggetto.)
- “Il diabolico Re Giovanni imperversò in Inghilterra nel 1200”
  - Giovanni, Inghilterra e 1200 sono oggetti, imperversò è una relazione, re e diabolico sono proprietà
- “I quadrati vicini al wumpus sono fetidi.”
  - Oggetti: wumpus, quadrati; Proprietà: fetore; Relazione: vicinato.

Logica del primo ordine

- Il linguaggio della logica del primo ordine è costruito attorno a oggetti e relazioni.
- È stato così importante per la matematica, la filosofia e l'intelligenza artificiale proprio perché quei campi, e in effetti, gran parte dell'esistenza umana quotidiana, possono essere utilmente pensati come attinenti agli oggetti e alle relazioni tra di essi.
- La logica del primo ordine può anche esprimere fatti su alcuni o tutti gli oggetti nell'universo.
- Ciò consente di rappresentare leggi o regole generali, come l'affermazione «I quadrati vicino al wumpus sono fetidi".

Logica del primo ordine vs logica proposizionale vs altre logiche

- **Impegno ontologico (ontological commitment)** assunto da ogni linguaggio: questo è ciò che assume sulla natura della realtà. Matematicamente, questo impegno si esprime attraverso la natura dei modelli formali rispetto ai quali si definisce la verità degli enunciati
- **Impegno epistemologico (epistemological commitment)** assunto da ogni linguaggio: stati di conoscenza che il linguaggio consente rispetto a ciascun fatto. Sia nella logica proposizionale che in quella del primo ordine, una frase rappresenta un fatto e l'agente:o crede che la frase sia vera, o falsa o non ha opinioni.

Logica del primo ordine vs logica proposizionale vs altre logiche

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Figure 8.1 Formal languages and their ontological and epistemological commitments.

Logica del primo ordine

Come ogni logica, la logica predicativa del primo ordine o elementare (FOL) è costituita da tre componenti:

• *un linguaggio formale*, che modella un frammento del linguaggio naturale (studiato per la prima volta come sistema logico a sé stante da Hilbert e Ackermann -1928);

• *una semantica del linguaggio formale*, che definisce le condizioni sotto cui un'espressione del linguaggio è vera oppure è falsa;

• un *calcolo*, ovvero un algoritmo per stabilire la validità di frasi e argomentazioni in modo puramente meccanico (*regola di inferenza*)

Logica del primo ordine

La logica dei predicati assume che tutte le rappresentazioni riguardino un insieme non vuoto (e per il resto arbitrario) di individui, detto ***dominio***.

Di questi individui possiamo rappresentare proprietà oppure relazioni che li leghino fra loro.

Da questo punto di vista, un ***fatto*** è semplicemente il sussistere di una proprietà di un determinato individuo ("Barbara è bionda")

oppure il sussistere di una relazione fra più individui ("Alberto è più alto di Barbara"),

Logica del primo ordine

- ✓ I modelli di un linguaggio logico sono le strutture formali che costituiscono i mondi possibili in esame.
- ✓ Ogni modello collega il vocabolario delle frasi logiche agli elementi del mondo possibile, in modo che la verità di ogni frase possa essere determinata.
- ✓ Pertanto, i modelli per la logica proposizionale collegano i simboli di proposizione a valori di verità predefiniti.
- ✓ I modelli per la logica del primo ordine sono molto più interessanti: contengono oggetti e l'insieme di oggetti o elementi di dominio che contiene

Logica del primo ordine

La figura mostra un modello con cinque oggetti:

Riccardo Cuor di Leone, re d'Inghilterra dal 1189 al 1199; suo fratello minore, il malvagio re Giovanni, che regnò dal 1199 al 1215; le gambe sinistre di Richard e John; e una corona

- Gli oggetti nel modello possono essere correlati in vari modi

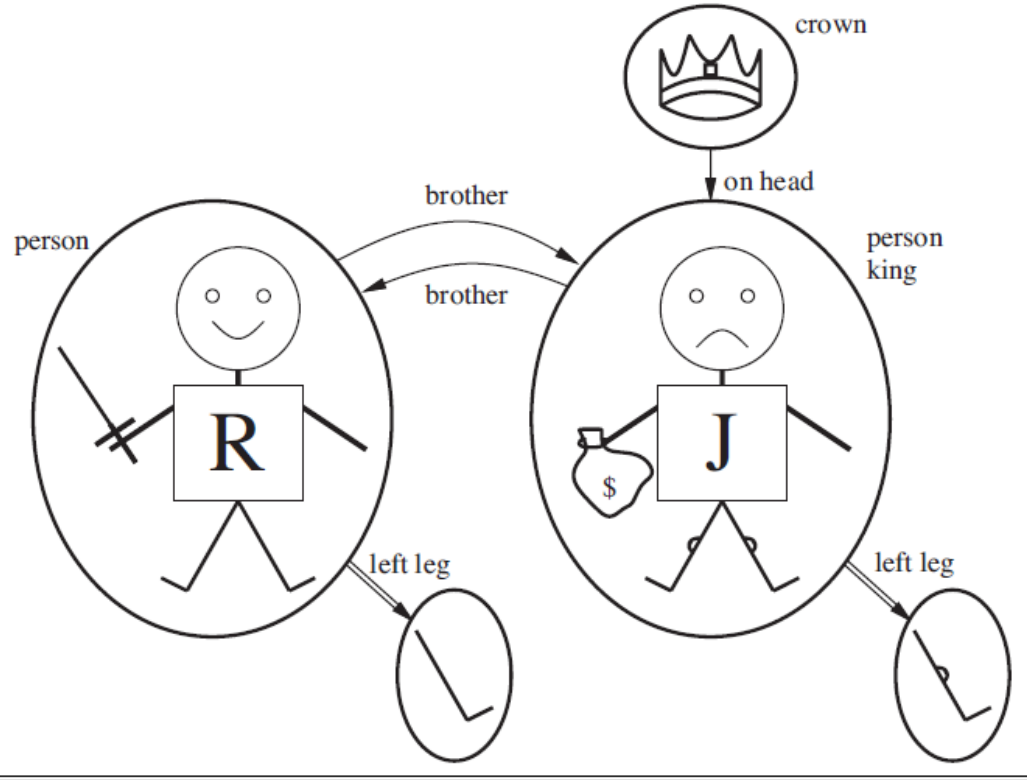


Figure 8.2 A model containing five objects, two binary relations, three unary relations (indicated by labels on the objects), and one unary function, left-leg.

Logica del primo ordine : Sintassi

- ✓ Le caratteristiche strutturali del linguaggio formale (mediante una grammatica) senza attribuire alcun significato ai simboli
- ➢ Si un compone di un alfabeto e di connettivi logici

•***Espressione o formula***: sequenza di simboli appartenenti all'alfabeto

Formule ben formate : frasi sintatticamente corrette del linguaggio.
Si ottengono attraverso combinazione di formule atomiche, utilizzando i connettivi e i quantificatori

Logica del primo ordine : Sintassi

Simboli (l'alfabeto)

Insieme di:

- Simboli di costante C: *A,B,C,Giovanni*
- Simboli di predicato (o relazione) F: *Tondo*, *Fratello*
- Simboli di funzione F: *Padre_di*, *Quadrato*
- Simboli di variabile *V X*, *Y (*entità non note del dominio)

Logica del primo ordine : Sintassi

Termini:

I simboli di costante sono termini es. Giovanni

Applicando una funzione n-adica a una n-pla di termini si ottiene un termine es. Padre_di(Giovanni)

N.b, le funzioni, in logica, non presuppongono alcun concetto di valutazione

Logica del primo ordine : Sintassi

Formule atomiche:

Formata da un simbolo di predicato seguito da una lista di termini

es:
Fratello(Riccardo,Giovanni)
Sposati(Madre_di(Riccardo),Padre_di(Riccardo))

Logica del primo ordine : Sintassi

- Le formule atomiche possono essere combinate per formare delle altre **formule complesse**:
  - Congiunzione: $A \& B$
  - Disgiunzione: $A \vee B$
  - Implicazione: $A \rightarrow B$
  - Negazione: $\neg A$

- Le formule con variabili possono essere interpretate tramite i quantificatori:
  - Universali: $\forall X\ p(X)$
  - Esistenziali: $\exists X\ p(X)$

Logica del primo ordine : Sintassi-riassunto

Elementi base

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Proposizioni /enunciati

Atomic sentence $= predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

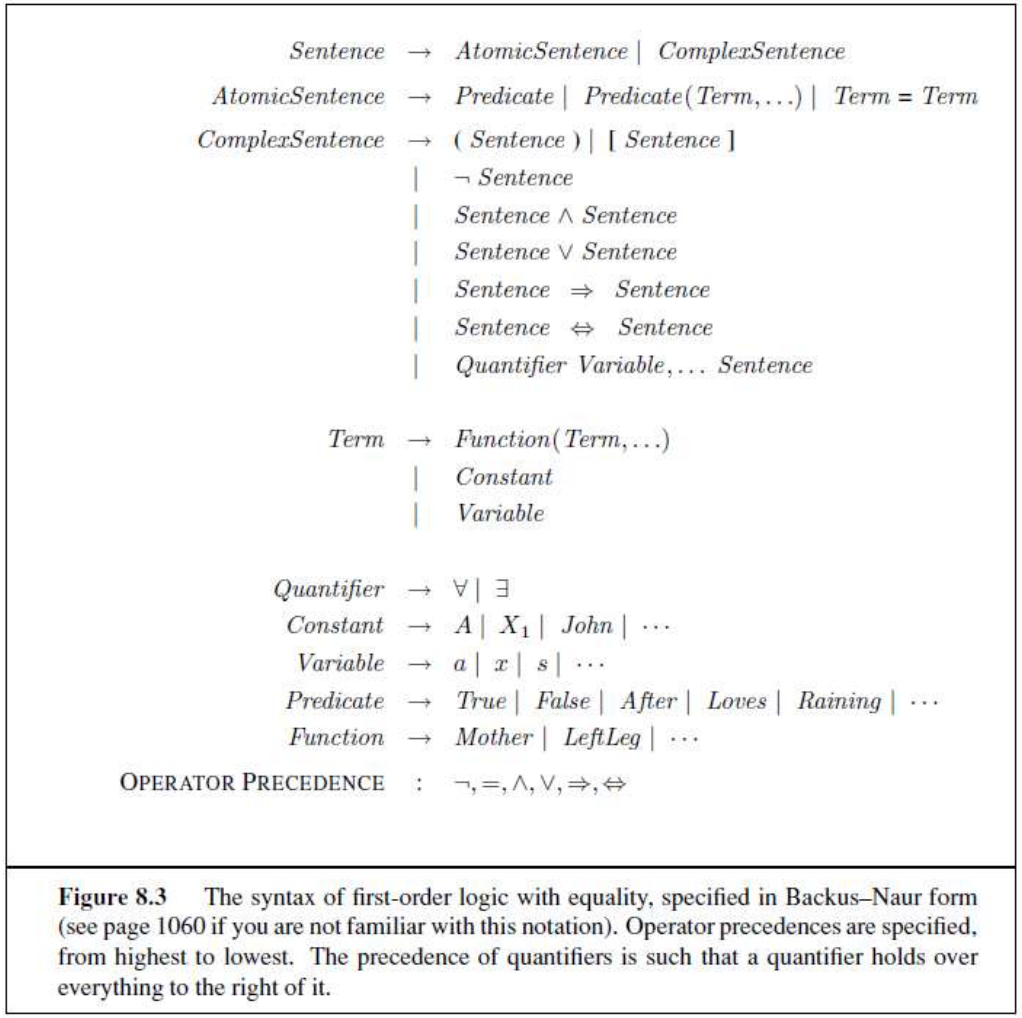
Term $= function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionhear$

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$$

Logica del primo ordine : Sintassi-riassunto



$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
 & \mid & \neg\ \textit{Sentence} \\
 & \mid & \textit{Sentence} \wedge \textit{Sentence} \\
 & \mid & \textit{Sentence} \vee \textit{Sentence} \\
 & \mid & \textit{Sentence} \Rightarrow \textit{Sentence} \\
 & \mid & \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
 & \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
 & & \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
 & \mid & \textit{Constant} \\
 & \mid & \textit{Variable} \\
 & & \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Figure 8.3 The syntax of first-order logic with equality, specified in Backus–Naur form (see page 1060 if you are not familiar with this notation). Operator precedences are specified, from highest to lowest. The precedence of quantifiers is such that a quantifier holds over everything to the right of it.

Logica del primo ordine: Semantica

Per dare un significato a una formula bisogna interpretarla come una affermazione sul dominio del discorso.

Cioè interpreta le frasi sintatticamente corrette del linguaggio.
Si dà una interpretazione alle formule stabilendo se una frase è vera o falsa

Logica del primo ordine: Semantica

- Occorre associare un significato ai simboli.
- Ogni sistema formale è la modellizzazione di una certa realtà (ad esempio la realtà matematica).
- Un'interpretazione è la costruzione di un rapporto fra i simboli del sistema formale e tale realtà (chiamata anche dominio del discorso).
- Ogni formula atomica o composta della logica dei predicati del primo ordine può assumere il valore vero o falso in base alla frase che rappresenta nel dominio del discorso

Logica dei predicati: Semantica

Un *dominio D* è un insieme non vuoto (anche infinito) ad es. Insieme di persone, l'insieme dei naturali etc.

Una *interpretazione* si ottiene associando

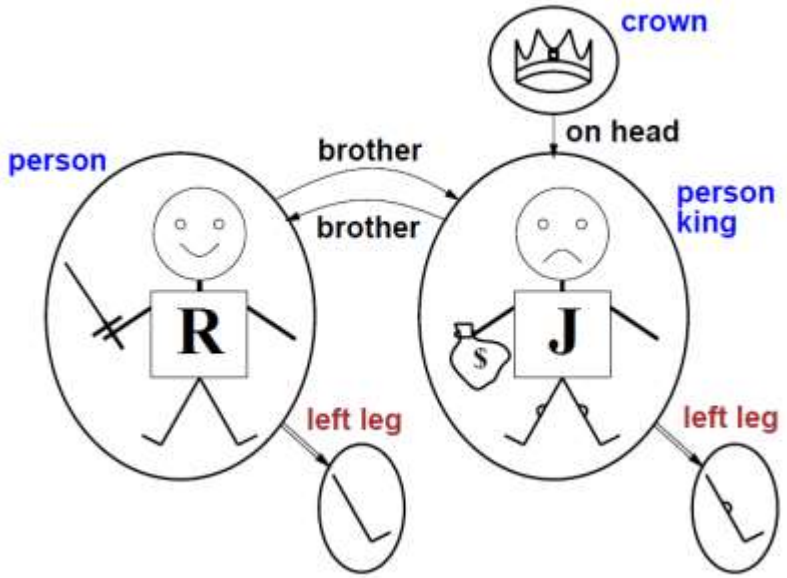
- ad ogni simbolo costante un elemento di ***D***
- ad ogni simbolo di funzione una funzione su ***D***
- ad ogni predicato n-ario una relazione n-aria su ***D***

Ad ogni formula atomica si assegna un valore ***vero*** o ***falso***

Ad ogni formula complessa si assegna un valore ***vero*** o ***falso*** utilizzando le tavole di verità

Logica del primo ordine: Semantica

- Si consideri l'interpretazione in cui
- Riccardo →Riccardo Cuor di Leone
- Giovanni →il malvagio re Giovanni
- Fratello → la relazione di fratellanza
- Secondo questa interpretazione, il fratello (Richard, John) è vero nel caso in cui Riccardo Cuor di Leone e il malvagio Re Giovanni siano nella relazione di fratellanza nel modello



Logica del primo ordine: Semantica

Esempio di interpretazione

Sia data la formula ***P(a,f(b,c))***

Una possibile interpretazione è:

D è il dominio degli interi

- ***a*** è l'intero 2
- ***b*** è l'intero 4
- ***c*** è l'intero 6
- ***f*** è la funzione *addizione*
- ***P*** è la relazione *maggiore di*

In questa interpretazione si afferma che: "***2 è maggiore di 4 + 6***" .
In questa interpretazione la formula ha valore ***falso***

In una seconda interpretazione possiamo dire ***a*** è l'intero 11e la formula assume valore vero

Logica del primo ordine: Semantica

- L'implicazione nella logica proposizionale può essere calcolato enumerando i modellos
- Possiamo farlo anche per the FOL datoil vocabolario della KB :
  1. Per ogni numero di elementi di dominio n da 1 a 1
  2. Per ogni predicato k-ary Pk nel vocabolario
  3. Per ogni possibile relazione k-ary su n oggetti
  4. Per ogni simbolo costante C nel vocabolario
  5. Per ogni scelta di un riferimento per C da n oggetti...

Calcolare l'implicazione enumerando i modelli FOL non è facile!

Logica del primo ordine: I quantificatori

- Una volta che abbiamo una logica che consente di rappresentare gli oggetti, è naturale voler esprimere le proprietà di intere raccolte di oggetti, invece di enumerare gli oggetti per nome.
- I quantificatori ci permettono di farlo.
- La logica del primo ordine contiene due quantificatori standard, chiamati quantificatore universale ed esistenziale.

Universal quantification (∀)

- Risolve il problema di come esprimere regole generali

“All kings are persons,” è scritto in FOL come

➢ ∀ x King(x) ⇒ Person(x)

- Quindi, la frase dice: "Per ogni x, se x è un re, allora x è una persona".
- Il simbolo x è chiamato **variabile**.
- Per convenzione, le variabili sono lettere minuscole.
- Una variabile è un termine a sé stante e, come tale, può anche fungere da argomento di una funzione, ad esempio LeftLeg(x).
- Un termine senza variabili è chiamato termine di base (**ground term)**.

Universal quantification (∀)

- Intuitivamente, la frase ∀x P, dove P è una qualsiasi espressione logica, dice che P è vero per ogni oggetto x.
- Più precisamente, ∀x P è vero in un dato modello se P è vero in tutte le possibili ***interpretazioni estese*** costruite dall'interpretazione data nel modello, dove ogni interpretazione estesa specifica un elemento di dominio a cui x si riferisce

$\forall \langle variables \rangle \ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x \ \ At(x, Berkeley) \Rightarrow Smart(x)$

$\forall x \ \ P$ is true in a model m iff P is true with x being **each** possible object in the model

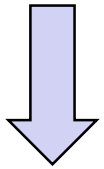
Roughly speaking, equivalent to the conjunction of instantiations of P

$$\begin{aligned} & (At(KingJohn, Berkeley) \Rightarrow Smart(KingJohn)) \\ \wedge \ & (At(Richard, Berkeley) \Rightarrow Smart(Richard)) \\ \wedge \ & (At(Berkeley, Berkeley) \Rightarrow Smart(Berkeley)) \\ \wedge \ & \ldots \end{aligned}$$

Universal quantification (∀)

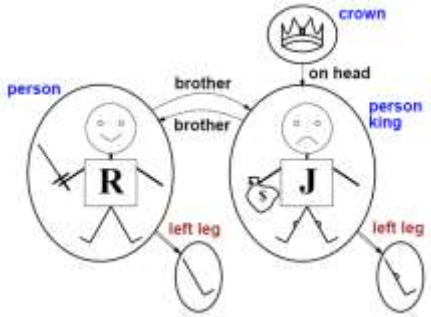
Un errore comune è usare la congiunzione invece dell'implicazione….

∀ x King(x) ∧ Person(x)



- Riccardo Cuor di Leone è un re ∧ Riccardo Cuor di Leone è una persona,
- Re Giovanni è un re ∧ Re Giovanni è una persona,
- La gamba sinistra di Riccardo è un re ∧ La gamba sinistra di Riccardo è una persona…

Universal quantification (∀)



Ci sono 5 elmenti nel dominio

x → Richard the Lionheart,
x → King John,
x → Richard's left leg,
x → John's left leg,
x → the crown.

∀ x King(x) ⇒ Person(x) is TRUE

BUT without quantifier
King(x) ⇒ Person(x)
is FALSE in the model

Richard the Lionheart is a king ⇒ Richard the Lionheart is a person.
King John is a king ⇒ King John is a person.
Richard's left leg is a king ⇒ Richard's left leg is a person.
John's left leg is a king ⇒ John's left leg is a person.
The crown is a king ⇒ the crown is a person.

Existential quantification (∃)

- Si riesce a dire una proposizione su qualche oggetto nell'universo senza nominarlo, ma usando un quantificatore esistenziale

$\exists x \; Crown(x) \land OnHead(x, John)$

- Cioè, almeno una delle seguenti è vera:
  1. Riccardo Cuor di Leone è una corona ∧ Riccardo Cuor di Leone è sulla testa di Giovanni;
  2. Re Giovanni è una corona ∧ Re Giovanni è sulla testa di Giovanni;
  3. La gamba sinistra di Richard è una corona ∧ La gamba sinistra di Richard è sulla testa di John;
  4. La gamba sinistra di Giovanni è una corona ∧ La gamba sinistra di Giovanni è sulla testa di Giovanni;
  5. La corona è una corona ∧ la corona è sulla testa di Giovanni.
- La quinta affermazione è vera nel modello, quindi la frase originale quantificata esistenzialmente è vera nel modello.

Existential quantification (∃)

- Intuitively, the sentence ∃x P says that P is true for at least one object x. More
- precisely, ∃x P is true in a given model if P is true in at least one extended interpretation that assigns x to a domain element.

$\exists \langle variables \rangle \ \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \ \ At(x, Stanford) \wedge Smart(x)$

$\exists x \ \ P$ is true in a model m iff P is true with x being **some** possible object in the model

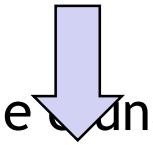
Roughly speaking, equivalent to the disjunction of instantiations of P

$$\begin{aligned} & (At(KingJohn, Stanford) \wedge Smart(KingJohn)) \\ \vee \ & (At(Richard, Stanford) \wedge Smart(Richard)) \\ \vee \ & (At(Stanford, Stanford) \wedge Smart(Stanford)) \\ \vee \ & \ldots \end{aligned}$$

Existential quantification (∃)

- Proprio come ⇒ sembra essere il connettivo naturale da usare con ∀, ∧ è il connettivo naturale da usare con ∃.
- L'uso di ∧ come connettivo principale con ∀ ha portato a un'affermazione eccessivamente forte
- L'uso di ⇒ con ∃ di solito porta a un'affermazione molto debole, infatti:

- ∃ x Crown(x) ⇒ OnHead(x, John)



- Riccardo Cuor di Leone è una corona ⇒ Riccardo Cuor di Leone è sulla testa di Giovanni;
- Re Giovanni è una corona ⇒ Re Giovanni è sulla testa di Giovanni;
- La gamba sinistra di Richard è una corona ⇒ La gamba sinistra di Richard è sulla testa di John
- …

Existential quantification (∃)

Ora una implicazione è vera se
1. sia la premessa che la conclusione sono vere,
2. *o se la sua premessa è falsa*.

quindi se Riccardo Cuor di Leone non è una corona, allora la prima affermazione è vera e l'esistenziale è soddisfatto.
Quindi, una proposizione di implicazione derivata dal quantificatore esistenziale è vera ogni volta che un oggetto non soddisfa la premessa
- quindi tali frasi in realtà non dicono molto.

Existential quantification (∃)

$$\exists x \;\; At(x, Stanford) \Rightarrow Smart(x)$$

is true if there is anyone who is not at Stanford!

Quantificatori annidati

- I quantificatori consecutivi dello stesso tipo possono essere scritti come un unico quantificatore con più variabili:
- $\forall$ x, y Sibling(x, y) $\Leftrightarrow$ Sibling(y, x)

- In altri casi l'ordine è importante:

- "Everybody loves somebody" (per ogni persona, c'è qualcuno che ama quella persona):
- $\forall$ x $\exists$ y Loves(x, y)
- "C'è qualcuno che è amato da tutti,"
- $\exists$ y $\forall$ x Loves(x, y)

Quantificatori annidati

- ∀ x (Crown(x) ∨ (∃ x Brother (Richard, x))) .

- Qui la x in Fratello (Riccardo, x) è quantificata dal quantificatore esistenziale. La regola è che la variabile appartiene al quantificatore più interno che la menziona; quindi non sarà soggetta a nessun'altra quantificazione.
- Poiché questo può essere fonte di confusione, utilizzeremo sempre nomi di variabili diversi con quantificatori annidati Brother (Richard, z)

Quantificatori annidati: connections between ∀ and ∃

- Affermare che a tutti non piacciono i broccoli equivale ad affermare che non esiste qualcuno a cui piacciono, e viceversa:
- ∀ x ¬Likes(x, Broccoli) is equivalent to ¬∃ x Likes(x, Broccoli) .
- "Il gelato piace a tutti" significa che non c'è nessuno a cui non piaccia il gelato:
- ∀ x Likes(x, IceCream) is equivalent to ¬∃ x ¬Likes(x, IceCream) .

Logica del primo ordine: uguaglianza

Il simbolo di uguaglianza indica che due termini si riferiscono allo stesso oggetto.

Father (John)=Henry

Per dire che Riccardo ha almeno due fratelli, scriveremmo

$\exists$ x, y Fratello (x,Riccardo) $\wedge$ Fratello (y,Riccardo) $\wedge\neg$(x=y)

La frase

$\exists$ x, y Fratello (x,Riccardo) $\wedge$ Fratello (y,Riccardo)

non ha il significato voluto.

In particolare, è vera nel modello in cui Riccardo ha un solo fratello.

Logica del primo ordine: uguaglianza

Crediamo che Richard abbia due fratelli, John e Geoffrey. Possiamo cogliere questo stato di cose asserendo

Fratello (Giovanni, Riccardo) ∧ Fratello (Geoffrey, Riccardo) ?

Non proprio:

- questa asserzione è vera in un modello in cui Richard ha un solo fratello, dobbiamo aggiungere John ≠Geoffrey.
- così xò la frase non esclude modelli in cui Richard ha altri fratelli oltre a John e Geoffrey.

➢ Pertanto, la traduzione corretta di "I fratelli di Richard sono John e Geoffrey" è la seguente:

Fratello (John,Richard) ∧ Fratello (Geoffrey,Richard) ∧ John ≠ Geoffrey

∧ ∀x Fratello (x,Riccardo) ⇒ (x=Giovanni ∨ x=Geoffrey)

Logica del primo ordine: esempi

Brothers are siblings

$$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$$

"Sibling" is symmetric

$$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x).$$

One's mother is one's female parent

$$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y)).$$

A first cousin is a child of a parent's sibling

$$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow\; \exists p, ps \;\; Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$$

Logica del primo ordine. Esempi

- I poliziotti arrestano i ladri

  $\forall x, y$ Poliziotto$(x) \wedge$ Ladro$(y) \rightarrow$ Arresta(x, y)

- Tutti i poliziotti arrestano un ladro

  $\forall x$ Poliziotto$(x) \rightarrow \exists y$ Ladro$(y) \wedge$ Arresta(x, y)

- Tutti i ladri sono arrestati da un poliziotto

  $\forall x$ Ladro$(x) \rightarrow \exists y$ Poliziotto$(y) \wedge$ Arresta(y, x)

- Esite un ladro che è arrestato da tutti i poliziotti

  $\exists x$ Ladro$(x) \wedge \forall y$ Poliziotto$(y) \rightarrow$ Arresta(y, x)

- Esiste un poliziotto che arresta tutti i ladri

  $\exists x$ Poliziotto$(x) \wedge \forall y$ Ladro$(y) \rightarrow$ Arresta(x, y)

Logica del primo ordine. Formalizziamo una situazione

Un possibile vocabolario è il seguente.

- Studente(x), Intelligente (x), Professore(x): predicati di significato intuitivo
- H: uno specifico corso di storia.
- B: uno specifico corso di biologia
- Segue(x,y): uno studente x segue un corso y.
- Supera(x,y): uno studente x supera l'esame del corso y
- Piace(x,y): y è simpatico a x.

Logica del primo ordine. Formalizziamo una situazione

- ✓ Non tutti gli studenti seguono sia il corso di Storia che quello di Biologia

  $\neg (\forall x \text{ Studente}(x) \Rightarrow (\text{Segue}(x,H) \wedge \text{Segue}(x,B)))$

- ✓ Un solo studente non ha superato l'esame di Storia

$\exists x ((\text{Studente}(x) \wedge \neg\text{Supera}(x,H) \wedge$

$(\forall y \text{ Studente}(y) \wedge \neg\text{Supera}(x,H))) \Rightarrow (x=y))$

- ✓ A nessuno piacciono professori non intelligenti

$\forall x,y ((\text{Persona}(x) \wedge \text{Professore}(y) \wedge \neg\text{Intelligente}(y)) \Rightarrow \neg\text{Piace}(x,y))$

Logica del primo ordine. Deduciamo un nuovo fatto

PARENTELE

$\forall(m, c)\text{Madre}(m, c) \Leftrightarrow \text{Femmina}(m) \wedge \text{Genitore}(m, c)$

$\forall(p, c)\text{Genitore}(p, c) \Leftrightarrow \text{Figlio}(c, p)$

$\forall(g, c)(\text{Nonno}(g, c) \Leftrightarrow (\exists p)\text{Genitore}(g, p) \wedge \text{Genitore}(p, c))$

$\forall(x, y)(\text{Fratello}(x, y) \Leftrightarrow x \neq y \wedge (\exists p)\text{Genitore}(p, x) \wedge \text{Genitore}(p, y))$

da :

$\textit{Genitore}(\textit{Giovanni}, \textit{Mario}) \wedge \textit{Genitore}(\textit{Giovanni}, \textit{Luca})$

si inferisce :

$\textit{Fratello}(\textit{Mario}, \textit{Luca})$

Logica del primo ordine. Interazione con un agente

Agenti Logici

TELL(KB, $(\forall xy)(\text{Fratello}(x, y) \Leftrightarrow$

$x \neq y \wedge (\exists p)(\text{Genitore}(p, x) \wedge \text{Genitore}(p, y))))$

TELL(KB, Genitore(Giovanni, Mario))

TELL(KB, Genitore(Giovanni, Luca))

si può chiedere :

$ASK(KB, Fratello(Mario, Luca))$

la risposta sarà si.

$ASK(KB, (\exists x)Genitore(Giovanni, x))$

la risposta sarà : {x/Mario} oppure {x/Luca}

function KB-AGENT(*percept*) **returns** an *action*
  static: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  t ← *t* + 1
  return *action*

Una risposta di questo tipo è chiamata lista di ***sostituzione*** o lista ***vincolante*** (substitution or binding list)

Logica del primo ordine. Il mondo dei wumpus

- ✓ Si può osservare che gli assiomi del primo ordine sono molto più concisi, in quanto catturano in modo naturale esattamente ciò che si deve dire
- ✓ Percept ([Stench, Breeze, Glitter , None, None], 5) .

dove, Percept è un predicato binario, e Stench Breeze … sono costanti messe in una lista.

Le azioni nel modndo del wumpus possono essere rappresentat da termini logici:

- Turn(Right), Turn(Left), Forward , Shoot , Grab .

Per determinare quale sia il migliore, il programma agente esegue la query

- ASKVARS(∃ a BestAction(a, 5))

Logica del primo ordine. Il mondo dei wumpus

"Perception"

$\forall b, g, t \;\; Percept([Smell, b, g], t) \Rightarrow Smelt(t)$

$\forall s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Reflex: $\forall t \;\; AtGold(t) \Rightarrow Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Logica del primo ordine. Il mondo dei wumpus

Dedurre le proprietà nascoste

Properties of locations:

$\forall x, t \;\; At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$

$\forall x, t \;\; At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \;\; Breezy(y) \Rightarrow \exists x \;\; Pit(x) \wedge Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \;\; Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the $Breezy$ predicate:

$$\forall y \;\; Breezy(y) \Leftrightarrow [\exists x \;\; Pit(x) \wedge Adjacent(x, y)]$$

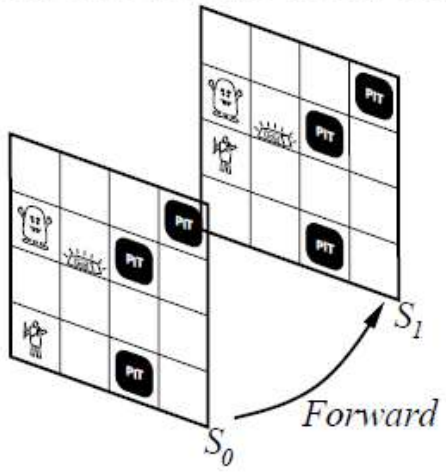
Logica del primo ordine. Il mondo dei wumpus

Tenere traccia del cambiamento

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:
Adds a situation argument to each non-eternal predicate
E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing a in s



Logica del primo ordine. Il mondo dei wumpus

Fare piani

Initial condition in KB:

$At(Agent, [1, 1], S_0)$

$At(Gold, [1, 2], S_0)$

Query: $Ask(KB, \exists s \;\; Holding(Gold, s))$

i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$

i.e., go forward and then grab the gold

Logica del primo ordine e regole di inferenza e forma a clausole

- La difficoltà di ottenere nuovi teoremi sta nel scegliere, di volta in volta, la regola di inferenza da applicare come nela logica proposizionale
- Le formule della logica dei predicati possono essere trasformate in una forma equivalente *a clausole*.
- **Una *clausola* è una formula priva di quantificatori ed è formata da una disgiunzione di termini**.
- Con la rappresentazione a clausole è così utilizzare un'unica regola di inferenza detta ***regola di risoluzione*.**

Logica del primo ordine e regole di inferenza e forma a clausole

- **Regola di risoluzione**: da $A \vee B, \neg B \vee C$ deduci $A \vee C$
- Il, modo analogo alla logica proposizionale, metodo di risoluzione permette di dimostrare un teorema mediante il metodo della *refutazione*, cioè si dimostra che la negazione del teorema è falsa:
  - Assumiamo che la negazione del teorema sia vera.
  - Mostriamo che gli assiomi più la negazione del teorema sono fra loro inconsistenti.
  - Dato che gli assiomi sono per definizione veri, allora la responsabile della contraddizione è la negazione del teorema.
  - Quindi il teorema è vero.
- Si ottiene una contraddizione quando si arriva a dedurre la clausola vuota.

Modus Ponens Generalizzato

- Per trasformare la base di conoscenza in una formula clausole devo utilizzare il modus ponens generalizzato in grado di eliminare i quantificatori universali
- E' il Modus Ponens Generalizzato

- Modus Ponens:
  - dall'implicazione e dalla premessa dell'implicazione si può inferire la conclusione (a->B, A =>B)

Modus Ponens Generalizzato con un esempio

Ad es. Se la KB contiene

- Barbie(B1)
- Possiede(Jenny, B1)
- $\forall$x Barbie(x) $\wedge$ Possiede(Jenny, x) $\Rightarrow$ Presta(Linda,Jenny,x)
- si vuole in un solo passo inferire
- Presta(Linda,Jenny,B1) .
- In questo caso la sostituzione $\{x \mid B1\}$ risolve il problema.
- Si verifica che la sostituzione {x/B1} rende la premessa dell'implicazione identica a frasi già in KB
- ***Più in generale, se una sostituzione comprendente x rende la premessa di implicazione identica a formule già presenti in KB, allora vale la conclusione dell'implicazione.***

Forma a Clausole e Modus Ponens Generalizzato

- Con il Modus Ponens generalizzato posso eliminare i quantificatori in un qualche ordine

- Per le formule a clausole (adatte per ragionare con un un'unica regola di risoluzione), vanno eliminati i quantificatori e gli AND, per cui

$\exists x$ Possiede(Jenny,x) $\wedge$ Barbie(x)

diventa due clausole

- Possiede(Jenny,x) e
- Barbie(x)

Modus Ponens Generalizzato e inferenza

- Con il Modus Ponens generalizzato possiamo:
  - partire dalle formule nella KB e generare nuove conclusioni che portino a nuove inferenze (**concatenazione in avanti**)
  - partire da qualcosa da dimostrare, trovare implicazioni che ci permettono di concluderlo e stabilire le premesse necessarie (**concatenazione all'indietro**)

Concatenazione in avanti

- Quando un nuovo fatto ***p*** è aggiunto alla KB
  - Si fanno scattare tutte le regole le cui premesse sono soddisfatte da p o sono già note.
  - Quindi si aggiunge la conclusione alla KB
  - Si continua nella catena finché si trova una risposta (supponendo che ne basti una) o non è più possibile aggiungere nuovi fatti

La concatenazione in avanti è guidata dai dati
e.g. inferiamo proprietà e categorie dalla percezione

Inferenza nella logica del primo ordine

Forward chaining example

Add facts 1, 2, 3, 4, 5, 7 in turn.
Number in [] = unification literal; $\surd$ indicates rule firing

1. $Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$
2. $Pig(y) \wedge Slug(z) \Rightarrow Faster(y, z)$
3. $Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$
4. $Buffalo(Bob)$ [1a,×]
5. $Pig(Pat)$ [1b,$\surd$] → 6. $Faster(Bob, Pat)$ [3a,×], [3b,×]
   [2a,×]
7. $Slug(Steve)$ [2b,$\surd$]
   → 8. $Faster(Pat, Steve)$ [3a,×], [3b,$\surd$]
      → 9. $Faster(Bob, Steve)$ [3a,×], [3b,×]

Concatenazione all'indietro

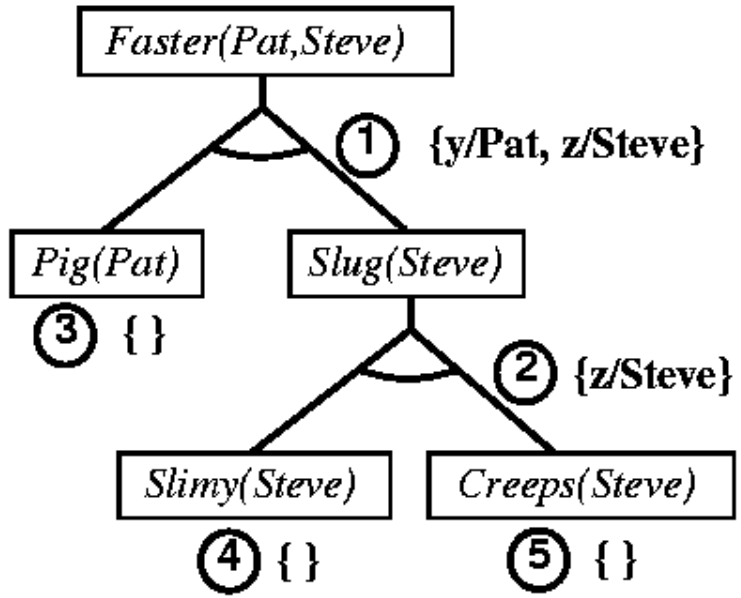
- Quando si formula una domanda
  - Se si conosce un fatto p che soddisfa la query ritorna il puntatore al fatto
  - Per ogni regola le cui conseguenze soddisfano la domanda si cerca di provare se ogni premessa della regola soddisfa la query in un meccanismo di concatenazione all'indietro

La concatenazione all'indietro è la base per la programmazione logica (prolog)

Inferenza nella logica del primo ordine

Backward chaining example

1. $Pig(y) \wedge Slug(z) \Rightarrow Faster(y, z)$
2. $Slimy(z) \wedge Creeps(z) \Rightarrow Slug(z)$
3. $Pig(Pat)$ 4. $Slimy(Steve)$ 5. $Creeps(Steve)$



Inferenza nella logica del primo ordine: esempio

- Ogni persona lavora o come infermiere oppure come insegnante
  $\neg \texttt{Lavora}(x, \texttt{Insegnante}) \vee \neg \texttt{Lavora}(x, \texttt{Infermiere})$
  $\texttt{Lavora}(x, \texttt{Insegnante}) \vee \texttt{Lavora}(x, \texttt{Infermiere})$

- Tutti gli infermieri sono maschi
  $\neg \texttt{Lavora}(x, \texttt{Infermiere}) \vee \texttt{Maschio}(x)$

- Tutte le persone sono o maschi o femmine
  $\texttt{Machio}(x) \vee \texttt{Femmina}(x)$
  $\neg \texttt{Maschio}(x) \vee \neg \texttt{Femmina}(x)$

- Steve è un maschio e Roberta è una femmina
  $\texttt{Maschio}(\texttt{Steve}), \texttt{Femmina}(\texttt{Roberta}),$

- Roberta non è una insegnante (Contraddizione)
  $\neg \texttt{Lavora}(\texttt{Roberta}, \texttt{Insegnante})$

Inferenza nella logica del primo ordine: esempio

1. $S = \{\neg\text{Lavora}(\text{Roberta}, \text{Insegnante})\}$

$$\frac{\text{Lavora}(x,\text{Insegnante}) \vee \text{Lavora}(x,\text{Infermiere}), \neg\text{Lavora}(\text{Roberta},\text{Insegnante})}{\text{SOST}(\{x/\text{Roberta}\},\text{Lavora}(x,\text{Infermiera}))}$$

2. $S = \{\neg\text{Lavora}(\text{Roberta}, \text{Insegnante}), \text{Lavora}(\text{Roberta}, \text{Infermiera})\}$

$$\frac{\neg\text{Lavora}(x,\text{Infermiera}) \vee \text{Maschio}(x), \text{Lavora}(\text{Roberta},\text{Infermiera})}{\text{SOST}(\{x/\text{Roberta}\},\text{Maschio}(x))}$$

3. $S = \{\neg\text{Lavora}(\text{Roberta}, \text{Insegnante}), \text{Lavora}(\text{Roberta}, \text{Infermiera}), \text{Maschio}(\text{Roberta})\}$

$$\frac{\neg\text{Maschio}(x) \vee \neg\text{Femmina}(x), \text{Maschio}(\text{Roberta})}{\text{SOST}(\{x/\text{Roberta}\}, \neg\text{Femmina}(x))}$$

4. $S = \{\neg\text{Lavora}(\text{Roberta}, \text{Insegnante}), \text{Lavora}(\text{Roberta}, \text{Infermiera})$
$\text{Maschio}(\text{Roberta}), \neg\text{Femmina}(\text{Roberta})\}$

$$\frac{\text{Femmina}(\text{Roberta}), \neg\text{Femmina}(\text{Roberta})}{\text{F}}$$

Per risoluzione unitaria derivo la clausola vuota!

Vantaggi della Logica

- I vantaggi principali della logica sono:
  - ***Precisione***, la logica ha una semantica chiara e definita per la quale esistono metodi standardizzati per determinare il significato di una espressione.
  - ***Flessibilità***, la logica rappresenta la conoscenza in modo dichiarativo, quindi in modo indipendente dal suo uso.
  - ***Modularità***, le asserzioni della logica possono entrare in una base di conoscenza in modo indipendente le une dalle altre.

Limiti della Logica

- I limiti principali della logica sono:
  - ***Inadeguatezza espressiva***, non può rappresentare mondi dinamici, non può rappresentare conoscenze probabilistiche.
  - ***Monotonicità,*** l'aggiunta di un teorema non provoca la cancellazione di nessun teorema precedente.
  - ***Inefficienza***, la dimostrazione automatica basata esclusivamente sulla logica si fonda su processi di ricerca, la ricerca è inerentemente esponenziale ed eventuali euristiche non ne cambiano la natura combinatoria.

Ingegneria della conoscenza nella logica del primo ordine

- Sviluppare una base di conoscenza in logica del primo ordine richiede
  - un attento processo di analisi del dominio
  - scelta di un vocabolario
  - codifica degli assiomi necessari a supportare le inferenze richieste
- ***Knowledge engineer*** è una persona che studia un particolare dominio, impara quali concetti sono importanti in quel dominio e crea una rappresentazione formale degli oggetti e delle relazioni nel dominio.

The knowledge-engineering process

1. Identificare il compito della base di conoscenza (task)
2. Raccogliere la conoscenza rilevante
3. Definire un vocabolario di predicati, funzioni e costanti *(ontologia di dominio)*
4. Codificare la conoscenza generale riguardante il dominio *(scrivere gli assiomi per tutti gli elementi)*
5. Codificare una descrizione della specifica istanza del problema (semplici formule atomiche di istanze di concetti)
6. Interrogare la procedura di inferenza e ottenere da essa risposte
7. Fare il debugging della base di conoscenza

The knowledge-engineering process

1. Identificare il compito della base di conoscenza (task)

- Si deve delineare la gamma di domande che la base di conoscenza dovrà supportare e i tipi di fatti che saranno disponibili per ogni specifica istanza di problema.
  - Per esempio, la base di conoscenza del mondo del wumpus deve essere in grado di scegliere le azioni o deve rispondere solo a domande sui contenuti dell'ambiente?
- Il compito determinerà quale conoscenza deve essere rappresentata per collegare le istanze del problema alle risposte.
- Questa fase è analoga al processo PEAS per la progettazione di agenti.

The knowledge-engineering process

2. Raccogliere la conoscenza rilevante (**knowledge acquisition**)

- L'ingegnere della conoscenza potrebbe essere già un esperto del dominio, o potrebbe aver bisogno di lavorare con esperti reali per estrarre ciò che sanno.
- In questa fase, la conoscenza non è rappresentata formalmente.
- L'idea è quella di comprendere l'ambito della base di conoscenza, come determinato dal compito, e di capire come funziona effettivamente il dominio.

The knowledge-engineering process

3. Definire un vocabolario di predicati, funzioni e costanti *(ontologia di dominio)*

- Ovvero, tradurre i concetti importanti a livello di dominio in nomi a livello logico.
- E' un processo creativo che dipende dallo stile dell'ingegnere della conoscenza
- La parola ontologia indica una particolare teoria sulla natura dell'essere o dell'esistenza. L'ontologia determina quali tipi di cose esistono, ma non determina le loro proprietà specifiche e le loro interrelazioni.

The knowledge-engineering process

4. Codificare la conoscenza generale riguardante il dominio

- Scrivere gli assiomi per tutti gli elementi.
- In questo modo si fissa (per quanto possibile) il significato dei termini, consentendo all'esperto di verificare il contenuto.
- Spesso questa fase rivela concetti errati o lacune nel vocabolario che devono essere risolte tornando alla fase 3 e iterando il processo.

The knowledge-engineering process

5. Codificare una descrizione della specifica istanza del problema

- Se l'ontologia è ben pensata, questo passo sarà facile.
- Si tratta di scrivere semplici frasi atomiche su istanze di concetti che fanno già parte dell'ontologia.

The knowledge-engineering process

6. Interrogare la procedura di inferenza e ottenere da essa risposte

- È qui che si trova la ricompensa:
- possiamo lasciare che la procedura di inferenza operi sugli assiomi e sui fatti specifici del problema per ricavare i fatti che ci interessa conoscere. In questo modo, evitiamo di dover scrivere un algoritmo di soluzione specifico per l'applicazione.

The knowledge-engineering process

7. Fare il debugging della base di conoscenza

- Le risposte del passo precedente saranno corrette per la base di conoscenza così come è stata scritta, assumendo che la procedura di inferenza sia corretta, ma potrebbero non essere quelle che l'utente si aspetta.
- Ad esempio, se manca un assioma, alcune interrogazioni non potranno essere soddisfatte dalla base di conoscenza.

Prolog

- Un programma PROLOG ha le seguenti caratteristiche:
  - Un programma è una sequenza di formule
  - Sono ammesse solo clausole di Horn
  - I termini possono essere simboli di costanti, variabili, termini funzionali
  - Le interrogazioni includono congiunzioni, disgiunzioni, variabili e termini funzionali
  - Non ci sono antecedenti negati, ma un goal *not P* si considera dimostrato se il sistema fallisce nel dimostrare *P*.

Prolog

Per provare …

http://www.swi-prolog.org/