Intelligenza Artificiale 2022-2023
Mario Verdicchio
Università degli Studi di Bergamo
Ingegneria Informatica

Lezione 2 (27 febbraio 2023)

Logica proposizionale

Nella logica proposizionale, come suggerisce il nome, proposizioni sono collegate da operatori logici. L'affermazione "la strada è bagnata" è una proposizione, così come "piove". Queste due proposizioni possono essere collegate per formare la nuova proposizione

se piove la strada è bagnata.

Scritto in modo più formale:

piove $\Rightarrow$ *la strada è bagnata*.

Questa notazione ha il vantaggio che le proposizioni elementari appaiono di nuovo in forma inalterata. Affinché possiamo lavorare precisamente con la logica proposizionale, inizieremo con una definizione dell'insieme di tutte le formule della logica proposizionale.

2.1 Sintassi

Definizione
Siano Op = {$\neg$; $\wedge$; $\vee$; $\Rightarrow$; $\Leftrightarrow$; (;)} l'insieme di operatori logici e Σ un insieme di simboli. Gli insiemi Op, Σ e {*t*; *f*} sono disgiunti a due a due. Σ è chiamata la firma (*signature*) e i suoi elementi sono le variabili proposizionali. L'insieme delle formule della logica proposizionale è ora definito in modo ricorsivo:
• *t* e *f* sono formule (atomiche).
• Tutte le variabili proposizionali, cioè tutti gli elementi di Σ, sono formule (atomiche).
• Se A e B sono formule, allora $\neg$A, (A), A$\wedge$B, A$\vee$B, A$\Rightarrow$B, A$\Leftrightarrow$B sono anche loro formule.

I simboli e gli operatori si leggono come segue:

t: "vero"
f: "falso"
$\neg$A: "non A" (negazione)
A$\wedge$B: "A e B" (congiunzione)
A$\vee$B: "A o B" (disgiunzione)
A$\Rightarrow$B: "se A allora B" (condizionale)
A $\Leftrightarrow$ B: "A se e solo se B" (doppio condizionale)

Le formule così definite sono finora costruzioni puramente sintattiche prive di significato. Ci manca ancora la semantica.

2.2 Semantica

Nella logica proposizionale ci sono due valori di verità: *t* per "vero" e *f* per "falso". Cominciamo con un esempio e ci chiediamo se la formula A ∧ B sia vera. La risposta è: dipende dal fatto che le variabili A e B siano vere. Ad esempio, se A sta per "Oggi piove" e B per "Oggi fa freddo" e queste sono entrambe vere, allora A ∧ B è vero. Se, tuttavia, B rappresenta "Fa caldo oggi" (e questo è falso), allora A ∧ B è falso.
Dobbiamo ovviamente assegnare valori di verità che riflettono lo stato del mondo alle variabili di proposizione. Quindi definiamo:

Definizione
Un assegnamento $I : \Sigma \rightarrow \{t ; f\}$, che assegna un valore di verità a ogni simbolo, si chiama "interpretazione".

Poiché ogni variabile proposizionale può assumere due valori di verità, ogni formula proposizionale con n variabili differenti ha 2^n interpretazioni differenti. Definiamo i valori di verità per le operazioni di base mostrando tutte le possibili interpretazioni in una cosiddetta tavola della verità:

A	B	(A)	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
t	t	t	f	t	t	t	t
t	f	t	f	f	t	f	f
f	t	f	t	f	t	t	f
f	f	f	t	f	f	t	t

La formula vuota è vera per tutte le interpretazioni.
Per determinare il valore di verità per formule complesse, dobbiamo anche definire l'ordine delle operazioni per gli operatori logici. Se le espressioni sono tra parentesi, il termine tra parentesi viene valutato per primo. Per le formule senza parentesi, le priorità sono ordinate come segue, a partire dal legame più forte: ¬, ∧, ∨, ⇒, ⇔.

Definizione
Due formule F e G sono chiamate (semanticamente) equivalenti se assumono lo stesso valore di verità per tutte le interpretazioni. Scriviamo F ≡ G.

L'equivalenza semantica serve soprattutto per poter usare il meta-linguaggio, cioè il linguaggio naturale, per parlare del linguaggio oggetto, cioè la logica. La formula "A ≡ B" indica che le due formule A e B sono semanticamente equivalenti. La formula "A ⇔ B" invece è un oggetto sintattico del linguaggio formale della logica proposizionale.

A seconda del numero di interpretazioni in cui una formula è vera, possiamo dividere le formule nelle seguenti classi:

Definizione
Una formula si dice
• *soddisfacibile* se è vera per almeno un'interpretazione (nota anche come *contingenza*).
• (logicamente) *valida* se è vera per tutte le interpretazioni. Formula valide sono anche chiamate *tautologie*. $(A \vee \neg A)$
• *insoddisfacibile* se non è vera per alcuna interpretazione. Formule insoddisfacibili si dicono anche *contraddizioni*. $(A \wedge \neg A)$
Ogni interpretazione che soddisfa una formula è chiamata *modello* della formula.

Chiaramente la negazione di ogni formula generalmente valida è insoddisfacibile. La negazione di una formula F soddisfacibile, ma non valida in generale, è soddisfacibile.
Ora siamo in grado di creare tavole di verità per formule complesse per accertare i loro valori di verità. Lo facciamo utilizzando equivalenze di formule utili.

Teorema
Gli operatori $\wedge$ e $\vee$ sono commutativi e associativi e sono valide le seguenti equivalenze:
$\neg A \vee B \equiv A \Rightarrow B$
$A \Rightarrow B \equiv \neg B \Rightarrow \neg A$ (legge di contrapposizione)
$(A \Rightarrow B) \wedge (B \Rightarrow A) \equiv (A \Leftrightarrow B)$
$\neg(A \wedge B) \equiv \neg A \vee \neg B$ (legge di DeMorgan 1)
$\neg(A \vee B) \equiv \neg A \wedge \neg B$ (legge di DeMorgan 2)
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ (distributività di $\vee$ su $\wedge$)
$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ (distributività di $\wedge$ su $\vee$)
$A \vee \neg A \equiv t$ (tautologia)
$A \wedge \neg A \equiv f$ (contraddizione)
$A \vee f \equiv A$
$A \vee t \equiv t$
$A \wedge f \equiv f$
$A \wedge t \equiv A$

Dimostrazione (per $\neg A \vee B \equiv A \Rightarrow B$)
Calcoliamo la tavola di verità per $\neg A \vee B$ e $A \Rightarrow B$ e vediamo che i valori di verità per entrambe le formule sono gli stessi per tutte le interpretazioni. Le formule sono quindi equivalenti e quindi tutti i valori dell'ultima colonna sono *t*.

A	B	$\neg A$	$\neg A \vee B$	$A \Rightarrow B$	$(\neg A \vee B) \Leftrightarrow (A \Rightarrow B)$
t	t	f	t	t	t
t	f	f	f	f	t
f	t	t	t	t	t
f	f	t	t	t	t

2.3 Sistemi di prova

Nell'intelligenza artificiale siamo interessati a prendere conoscenza preesistente, e da quella derivare nuova conoscenza, o usarla per rispondere a domande. Nella logica proposizionale questo significa mostrare che da una base di conoscenza (KB, Knowledge Base), cioè una formula di logica proposizionale (possibilmente estesa, ossia un certo numero di formule messe in congiunzione tra loro), segue una formula Q (Q come "query", chiamata anche "tesi" in ambito della logica). Quindi, definiamo prima il termine "implicazione".

Definizione
Una formula KB implica una formula Q (o Q segue da KB) se ogni modello di KB è anche un modello di Q. Scriviamo KB $\models$ Q.

In altre parole, in ogni interpretazione in cui KB è vera, anche Q è vera. Più succintamente, ogni volta che KB è vero, anche Q è vero. Poiché per nel concetto di implicazione vengono introdotte interpretazioni di variabili, si tratta di un concetto semantico.
Ogni formula non valida (ossia non sempre vera) sceglie, per così dire, un sottoinsieme dell'insieme di tutte le interpretazioni come proprio modello. Tautologie come $A \vee \neg A$, ad esempio, non limitano il numero di interpretazioni soddisfacenti perché la loro proposizione è vuota (ossia non dicono nulla). La formula vuota è quindi vera in tutte le interpretazioni. Per ogni tautologia T allora: $\varnothing \models T$. Intuitivamente questo significa che le tautologie sono sempre vere, senza restrizione delle interpretazioni di una formula. In breve scriviamo $\models T$. Ora mostriamo un'importante connessione tra il concetto semantico di implicazione e il condizionale sintattico.

Teorema (Teorema di deduzione)

$$A \models B \text{ se e solo se } \models A \Rightarrow B.$$

Dimostrazione
Osservando la tavola della verità per il condizionale:

A	B	$A \Rightarrow B$
t	t	t
t	f	f
f	t	t
f	f	t

Un condizionale arbitrario $A \Rightarrow B$ è chiaramente sempre vero tranne che con l'interpretazione $A \mapsto t$, $B \mapsto f$. Supponiamo che $A \models B$ valga. Ciò significa che per ogni interpretazione che rende A vero, anche B. La seconda riga critica della tabella di verità non si applica nemmeno in quel caso. Quindi $A \Rightarrow B$ è vero, il che significa che $A \Rightarrow B$ è una tautologia. Così è stata mostrata una direzione del teorema (da sinistra a destra).

Ora supponiamo che $A \Rightarrow B$ sia valida. Così anche la seconda riga critica della tabella della verità è bloccata. Ogni modello di A è allora è anche il modello di B. Quindi $A \models B$. □

Se vogliamo dimostrare che KB implica Q, possiamo anche dimostrare mediante il metodo della tavola di verità che KB ⇒ Q è una tautologia. Così abbiamo il nostro primo sistema di prova per la logica proposizionale, facilmente automatizzabile. Lo svantaggio di questo metodo è il tempo di calcolo molto lungo nel peggiore dei casi. Nello specifico, nel caso peggiore con n variabili di proposizione, per tutte le 2^n interpretazioni delle variabili deve essere valutata la formula KB ⇒ Q. Il tempo di calcolo cresce quindi in modo esponenziale con il numero di variabili. Pertanto questo processo è inutilizzabile per conteggi di variabili di grandi dimensioni, almeno nel caso peggiore.

Se una formula KB implica una formula Q, allora per il teorema di deduzione KB ⇒ Q è una tautologia. Quindi la negazione ¬ (KB ⇒ Q) è insoddisfacibile. Abbiamo

$$\neg (KB \Rightarrow Q) \equiv \neg (\neg KB \vee Q) \equiv KB \wedge \neg Q$$

Pertanto, anche KB ∧ ¬ Q è insoddisfacibile. Formuliamo questa semplice ma importante conseguenza del teorema di deduzione come teorema.

Teorema (Dimostrazione per contraddizione):
KB $\models$ Q se e solo se KB ∧ ¬Q è insoddisfacibile.

Per mostrare che la query Q segue dalla Knowledge Base KB, possiamo aggiungere la query negata ¬Q alla knowledge base e derivare una contraddizione. A causa dell'equivalenza $A \wedge \neg A \equiv f$ sappiamo che una contraddizione è insoddisfacibile. In un modello di KB, ossia in una interpretazione in cui KB è vera, abbiamo quindi che KB ∧ ¬Q non può essere vera, quindi si deduce la verità di Q. Questa procedura, che è frequentemente utilizzata in matematica, viene utilizzata anche in vari calcoli di prova automatici come il calcolo della risoluzione, che vedremo tra poco.

Un modo per evitare di dover testare tutte le interpretazioni con il metodo della tavola di verità è la manipolazione sintattica delle formule KB e Q mediante l'applicazione di regole di inferenza con l'obiettivo di semplificarle notevolmente, in modo tale che alla fine possiamo immediatamente vedere che KB $\models$ Q. Chiamiamo questo processo *derivazione sintattica* e scriviamo

$$KB \vdash Q.$$

Tali sistemi di dimostrazione sintattica sono chiamati calcoli. Per garantire che un calcolo non generi errori, definiamo due proprietà fondamentali dei calcoli.

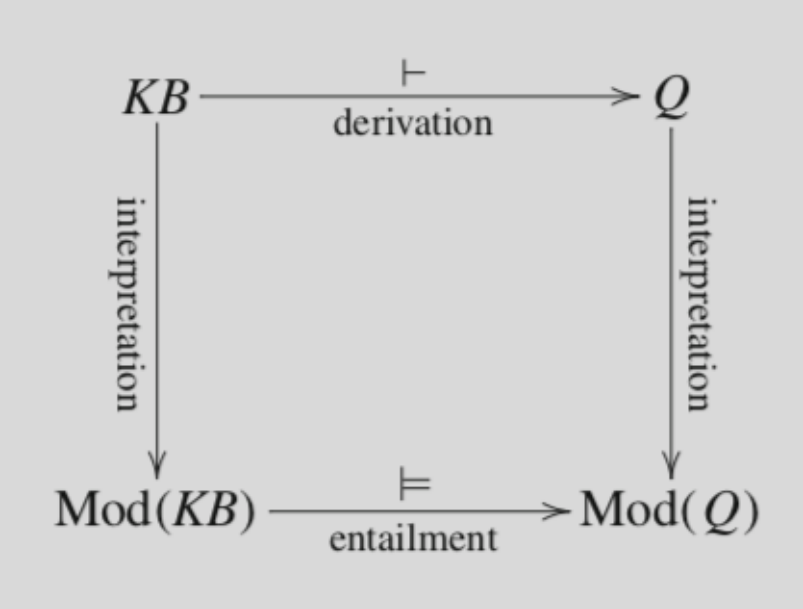
Definizione
Un calcolo è chiamato corretto ("sound" in inglese) se ogni proposizione derivata per via sintattica con il calcolo è una conseguenza anche dal punto di vista semantico. Cioè, vale per le formule KB e Q quanto segue:

se KB $\vdash$ Q allora KB $\models$ Q.

Definizione
Un calcolo è detto completo se tutte le conseguenze semantiche possono essere derivate con il calcolo. Cioè, per le formule KB e Q vale quanto segue:

se KB $\models$ Q allora KB $\vdash$ Q.

La correttezza di un calcolo garantisce che tutte le formule derivate siano di fatto conseguenze semantiche della base di conoscenza. Il calcolo non produce "false conseguenze". La completezza di un calcolo, d'altra parte, assicura che il calcolo non trascuri nulla. Un calcolo completo trova sempre una prova se la formula da dimostrare segue semanticamente dalla base di conoscenza. Se un calcolo è corretto e completo, la derivazione sintattica e la conseguenza semantica sono due relazioni equivalenti (vedi figura: in alto c'è il livello sintattico, in basso quello semantico).



Per mantenere i sistemi di prova automatici i più semplici possibili, essi sono solitamente disegnati per operare su formule in *forma normale congiuntiva*.

Definizione
Una formula è in forma normale congiuntiva ("conjunctive normal form", CNF) se e solo se consiste in una congiunzione $K_1 \wedge K_2 \wedge \ldots \wedge K_m$ di clausole.

Una clausola K_i consiste in una disgiunzione $(L_{i1} \vee L_{i2} \vee \ldots \vee L_{in})$ di letterali.
Un letterale è una variabile proposizionale (A, letterale positivo) o una variabile proposizionale negata (¬B, letterale negativo).

La formula $(A \vee B \vee \neg C) \wedge (A \vee B) \wedge (\neg B \vee \neg C)$ è in forma normale congiuntiva. La forma normale congiuntiva non pone restrizioni sull'insieme di formule perché:

Teorema
Ogni formula della logica proposizionale può essere trasformata in una forma normale congiuntiva equivalente.

Esempio (non è una dimostrazione!)

$$
\begin{aligned}
& A \vee B \Rightarrow C \wedge D \\
& \equiv \neg(A \vee B) \vee (C \wedge D) && \text{(implication)} \\
& \equiv (\neg A \wedge \neg B) \vee (C \wedge D) && \text{(de Morgan)} \\
& \equiv (\neg A \vee (C \wedge D)) \wedge (\neg B \vee (C \wedge D)) && \text{(distributive law)} \\
& \equiv ((\neg A \vee C) \wedge (\neg A \vee D)) \wedge ((\neg B \vee C) \wedge (\neg B \vee D)) && \text{(distributive law)} \\
& \equiv (\neg A \vee C) \wedge (\neg A \vee D) \wedge (\neg B \vee C) \wedge (\neg B \vee D) && \text{(associative law)}
\end{aligned}
$$

Il modo di procedere più comune è:
1. eliminazione di ⇔ trasformandolo in congiunzione di ⇒
2. eliminazione di ⇒ trasformandolo in negazione e disgiunzione
3. usare De Morgan per spostare le negazioni "verso l'interno", in modo da avere letterali negativi anziché intere formule negate
4. applicare distributività e associatività per ottenere congiunzioni di disgiunzioni

Ora ci manca solo un calcolo per la prova sintattica delle formule di logica proposizionale. Partiamo dal modus ponens, una regola di inferenza semplice e intuitiva, che, dalla validità di A e A ⇒ B, consente la derivazione di B. In maniera più formale:

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

Questa notazione significa che possiamo derivare le formule sotto la riga dalle formule separate da virgole sopra la riga. Il modus ponens da solo, come regola di inferenza, è corretto ma non è completo. Ad esempio, da A non riusciamo con il modus ponens a ottenere A∨B, nonostante questa sia una conseguenza semantica di A. Se aggiungiamo regole aggiuntive possiamo creare un calcolo completo, che, tuttavia, non è lo scopo di questa lezione. Invece esamineremo la regola della *risoluzione* come alternativa al modus ponens.

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

Le formule in input sono *clausole*, ossia disgiunzioni di letterali.
La formula derivata si chiama *risolvente*. Se mettiamo *f* al posto di A, e trasformiamo ¬B∨C nella sua equivalente B ⇒ C, si nota come la risoluzione sia una forma più generale del modus ponens (anche se ancora non riusciamo a derivare A∨B a partire da A).
Se omettiamo A e C, abbiamo B e ¬B come input e la risolvente è vuota. La clausola vuota si indica solitamente con { }, oppure [].

2.4 Risoluzione

Ora generalizziamo nuovamente la regola di risoluzione consentendo clausole con un numero arbitrario di letterali. Con i letterali A_1, ..., A_m, B, C_1, ..., C_n la regola di risoluzione generale è come segue:

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), \quad (\neg B \vee C_1 \vee \cdots \vee C_n)}{(A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n)}$$

Chiamiamo i letterali B e ¬B complementari. La regola di risoluzione elimina una coppia di letterali complementari dalle due clausole e combina il resto dei letterali in una nuova clausola.
Per dimostrare che da una knowledge base KB segue una query Q, eseguiamo una dimostrazione per contraddizione. Dobbiamo mostrare che una contraddizione può essere derivata da KB ∧ ¬Q. Nelle formule in forma congiuntiva normale, se c'è una contraddizione essa compare nella forma di due clausole (A) e (¬A), che portano alla clausola vuota come loro risolvente. Il teorema che segue ci assicura che questo processo funzioni davvero come desiderato.

Affinché il calcolo sia completo, abbiamo bisogno di una piccola aggiunta, come mostrato dal seguente esempio. Sia data la formula (A ∨ A) come nostra base di conoscenza. Per mostrare che con la regola di risoluzione da lì possiamo derivare (A ∧ A), dobbiamo mostrare che la clausola vuota può essere derivata da (A ∨ A) ∧ (¬A ∨ ¬A). Con la sola regola di risoluzione, questo è impossibile. Con la *fattorizzazione*, che consente la cancellazione di copie di letterali dalle clausole, questo problema viene eliminato. Nell'esempio, una doppia applicazione della fattorizzazione porta a (A) ∧ (¬A) e un passaggio di risoluzione porta alla clausola vuota.

Teorema
Il calcolo della risoluzione per la dimostrazione di insoddisfacibilità di formule in forma congiuntiva normale è valido e completo.

Se è completo, allora dovrebbe riuscire a derivare A ∨ B da A. Visto che la dimostrazione è di insoddisfacibilità, noi usiamo la negazione della formula che vogliamo derivare, la aggiungiamo alla KB e procediamo con la risoluzione, in cerca della clausola vuota.
Tesi: A ∨ B
KB: A
Tesi negata: ¬(A ∨ B) ≡ ¬A ∧ ¬B
KB + tesi negata: {A, ¬A, ¬B} (le formule in CNF vengono separate in clausole nella KB)
Da A e ¬A con la risoluzione ottengo subito la clausola vuota, dimostrando che nella KB + tesi negata c'è una contraddizione, quindi la tesi originale (quella non negata) è sempre vera quando la KB è vera, ossia è una sua conseguenza semantica.

Poiché è compito del calcolo della risoluzione derivare una contraddizione da KB ∧ ¬Q, è fondamentale che la base di conoscenza KB sia consistente

Definizione
Una formula KB si dice consistente se non è possibile derivarne una contraddizione, cioè una formula della forma $\phi \wedge \neg\phi$.

Altrimenti qualsiasi cosa può essere derivata da una KB inconsistente. Ad esempio, se una contraddizione come avere sia A sia ¬A in una KB è ammessa, allora ¬B (formula scelta

arbitrariamente nell'esempio precedente) può essere aggiunta alla KB senza problemi, e così via: con aggiunte arbitrarie la KB si "arricchisce" di formule a caso, tutte derivabili col calcolo. Questo è vero non solo per la risoluzione, ma anche per molti altri calcoli.
Tra i calcoli per la deduzione automatica, la risoluzione gioca un ruolo molto importante. A differenza di altri calcoli, la risoluzione ha solo due regole di inferenza e funziona con le formule in forma normale congiuntiva. Ciò rende la sua implementazione più semplice. Un ulteriore vantaggio rispetto a molti calcoli sta nella sua riduzione del numero di possibilità di applicazione delle regole di inferenza in ogni fase della dimostrazione, per cui lo spazio di ricerca è ridotto e il tempo di calcolo diminuito.

2.5 Clausole di Horn

Una clausola in forma normale congiuntiva contiene letterali positivi e negativi e può essere rappresentata nella forma

$$\left(\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n\right)$$

con tutti i letterali negativi a sinistra e quelli positivi a destra. Usando l'equivalenza $\neg A \vee B \equiv A \Rightarrow B$, possiamo trasformare la clausola nella formula

$$A_1 \wedge \cdots \wedge A_m \Rightarrow B_1 \vee \cdots \vee B_n.$$

In un esempio della vita di tutti giorni, una formula del genere potrebbe essere la rappresentazione formale di una frase come la seguente:

Se il tempo è bello e c'è neve, allora andrò a sciare oppure andrò in ufficio.

Non una frase particolarmente informativa. Molto più chiara è una frase come questa:

Se il tempo è bello e c'è neve e il mio capo è d'accordo, allora andrò a sciare.

Questa frase, se formalizzata, avrebbe la forma:

$$A_1 \wedge \cdots \wedge A_m \Rightarrow B_1$$

ossia, in forma di clausola:

$$\neg A_1 \vee \cdots \vee \neg A_m \vee B_1$$

Questa è una clausola con un solo letterale positivo. Questo tipo di clausole ha il vantaggio di consentire solo una conclusione (es. "andrò a sciare", anziché "andrò a sciare oppure andrò in ufficio") e sono quindi decisamente più semplici da interpretare. Molte relazioni possono essere descritte da clausole di questo tipo. Clausole con al massimo un letterale positivo si chiamano *clausole definite* o *clausole di Horn* (dal nome di chi le ha inventate e definite).

Definizione
Clausole con al massimo un letterale positivo nella forma
$\neg A_1 \vee \ldots \vee \neg A_m \vee B$ oppure $\neg A_1 \vee \ldots \vee \neg A_m$ oppure B
o, equivalentemente,
$(A_1 \wedge \ldots \wedge A_m) \Rightarrow B$ oppure $A_1 \wedge \ldots \wedge A_m \Rightarrow f$ oppure B

sono chiamate clausole di Horn.
Una clausola con un singolo letterale positivo è chiamata *fatto*. Nelle clausole con letterali negativi e un letterale positivo, il letterale positivo è chiamato *testa*.

Le clausole di Horn sono più facili da gestire non solo nella vita quotidiana, ma anche nel ragionamento formale, come possiamo vedere nell'esempio seguente. Sia la base di conoscenza composta dalle seguenti clausole (la "∧" che lega le clausole è tralasciata):

$$(nice\_weather)_1$$
$$(snowfall)_2$$
$$(snowfall \Rightarrow snow)_3$$
$$(nice\_weather \wedge snow \Rightarrow skiing)_4$$

Se ora vogliamo sapere se "*skiing*" è vera, questo può essere facilmente dedotto. Un modus ponens leggermente generalizzato è sufficiente qui come regola di inferenza:

$$\frac{A_1 \wedge \cdots \wedge A_m, \quad A_1 \wedge \cdots \wedge A_m \Rightarrow B}{B}$$

La prova di "skiing" ha la seguente forma (MP (i_1, ..., i_k) rappresenta l'applicazione del modus ponens sulle clausole da i_1 a i_k:

$$\text{MP}(2,3): \quad (snow)_5$$
$$\text{MP}(1,5,4): \quad (skiing)_6$$

Con il modus ponens otteniamo un calcolo completo per formule che consistono in clausole di Horn della logica proposizionale. Nel caso di grandi basi di conoscenza, tuttavia, il modus ponens può derivare molte formule non necessarie se si inizia con le clausole sbagliate. Pertanto, in molti casi è meglio utilizzare un calcolo che inizi con la query e funzioni a ritroso fino a raggiungere i fatti. Tali sistemi sono chiamati *backward chaining* (concatenamento all'indietro), in contrasto con i sistemi di *forward chaining* (concatenamento in avanti), che iniziano con i fatti e infine derivano la tesi, come nell'esempio precedente con il modus ponens.
Per il concatenamento all'indietro delle clausole Horn, viene utilizzata la risoluzione SLD. SLD sta per "Selection-rule-driven Linear resolution for Definite clauses" (risoluzione lineare basata su regole di selezione per clausole definite). Nell'esempio sopra, a cui aggiugiamo la tesi negata (skiing ⇒ *f*),

$$(nice\_weather)_1$$
$$(snowfall)_2$$
$$(snowfall \Rightarrow snow)_3$$
$$(nice\_weather \wedge snow \Rightarrow skiing)_4$$
$$(skiing \Rightarrow f)_5$$

eseguiamo la risoluzione SLD a partire dai passi di risoluzione che seguono dalla clausola della tesi negata:

$$\begin{aligned}
&\text{Res}(5,4): && (nice_weather \wedge snow \Rightarrow f)_6\\
&\text{Res}(6,1): && (snow \Rightarrow f)_7\\
&\text{Res}(7,3): && (snowfall \Rightarrow f)_8\\
&\text{Res}(8,2): && ()
\end{aligned}$$

e derivare una contraddizione con la clausola vuota. Qui possiamo facilmente vedere che cosa vuol dire "risoluzione lineare": il passo successivo viene sempre eseguito sulla clausola appena derivata. Questo porta ad una grande riduzione dello spazio di ricerca.

I letterali della clausola corrente vengono sempre elaborati in un ordine fisso (ad esempio, da destra a sinistra). I letterali della clausola corrente sono chiamati sotto-obiettivo (*subgoal*). I letterali della query negata sono gli obiettivi (*goal*). La regola di inferenza per un singolo passaggio ha la seguente forma:

$$\frac{A_1 \wedge \cdots \wedge A_m \Rightarrow B_1, \quad B_1 \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow f}{A_1 \wedge \cdots \wedge A_m \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow f}$$

Prima dell'applicazione della regola di inferenza, B_1, B_2, ..., B_n (gli attuali sotto-obiettivi) devono essere dimostrati.
Dopo la applicazione della regola, il sotto-obiettivo B1 è sostituito dal nuovo sotto-obiettivo $A_1 \wedge \ldots \wedge A_m$.
Per mostrare che B_1 è vero, dobbiamo ora mostrare che $A_1 \wedge \ldots \wedge A_m$ è vero.
Questo processo continua fino a quando l'elenco dei sotto-obiettivi delle clausole correnti (il cosiddetto stack degli obiettivi) è vuoto. Con ciò, è stata trovata una contraddizione. Se, per un obiettivo secondario $\neg B_i$, non esiste una clausola con il letterale complementare B_i come testa della clausola, la dimostrazione termina e non si può trovare alcuna contraddizione. La tesi, in tal caso, non è dimostrabile.
La risoluzione SLD gioca un ruolo importante nella pratica perché i programmi nel linguaggio di programmazione logica PROLOG sono costituiti da clausole di Horn in logica predicativa e la loro elaborazione è ottenuta mediante risoluzione SLD.


Situazione indicata con proposizione P.


Serve una nuova proposizione Q.

Nella formalizzazione con la logica proposizionale P e Q sono distinte come lo sono due proposizioni qualunque A e B. Per una modellizzazione migliore occorre introdurre la capacità di fare riferimento a entità e non solo a situazioni.

Logica del prim'ordine

Siano:

V un insieme di variabili
K un insieme di costanti
F un insieme di simboli funzionali

Definiamo l'insieme dei termini come segue:
- tutte le variabili e tutte le costanti sono termini;
- se t_1, t_2, …, t_k sono termini e f è un simbolo funzionale a k posti, allora $f(t_1,t_2, …, t_k)$ è un termine.

In più, sia P l'insieme dei simboli di predicato, con ciascun simbolo caratterizzato da una "arità", Ossia il numero di termini a cui il simbolo si applica.

Definizione delle formule della logica del prim'ordine.
- Se $t_1,…,t_n$ sono n termini e P è un simbolo di predicato n-ario, allora $P(t_1,…,t_n)$ è una formula.
- Se A e B sono formule, allora lo sono anche $\neg A$, $A \vee B$, $A \wedge B$, $A \Rightarrow B$, $A \Leftrightarrow B$
- Se x è una variabile e A è una formula, allora anche $\forall xA$ e $\exists xA$ lo sono.

Table 3.1 Examples of formulas in first-order predicate logic. Please note that *mother* here is a function symbol

Formula	Description
$\forall x\ frog(x) \Rightarrow green(x)$	All frogs are green
$\forall x\ frog(x) \wedge brown(x) \Rightarrow big(x)$	All brown frogs are big
$\forall x\ likes(x, cake)$	Everyone likes cake
$\neg\forall x\ likes(x, cake)$	Not everyone likes cake
$\neg\exists x\ likes(x, cake)$	No one likes cake
$\exists x\ \forall y\ likes(y, x)$	There is something that everyone likes
$\exists x\ \forall y\ likes(x, y)$	There is someone who likes everything
$\forall x\ \exists y\ likes(y, x)$	Everything is loved by someone
$\forall x\ \exists y\ likes(x, y)$	Everyone likes something
$\forall x\ customer(x) \Rightarrow likes(bob, x)$	Bob likes every customer
$\exists x\ customer(x) \wedge likes(x, bob)$	There is a customer whom bob likes
$\exists x\ baker(x) \wedge \forall y\ customer(y) \Rightarrow likes(x, y)$	There is a baker who likes all of his customers
$\forall x\ older(mother(x), x)$	Every mother is older than her child
$\forall x\ older(mother(mother(x)), x)$	Every grandmother is older than her daughter's child
$\forall x\ \forall y\ \forall z\ rel(x, y) \wedge rel(y, z) \Rightarrow rel(x, z)$	*rel* is a transitive relation

Intepretazione

Definizione: una interpretazione è
- una corrispondenza tra l'insieme unione delle variabili e le costanti e un insieme di nomi di oggetti nel mondo
- una corrispondenza tra l'insieme dei simboli funzionali e l'insieme delle funzioni nel mondo (con le arità rispettate)
- una corrispondenza tra l'insieme dei simboli di predicato e le relazioni nel mondo (con le arità rispettate).
Una formula elementare P(t) è vera in una interpretazione I quando a t corrisponde un oggetto nel mondo che gode della proprietà corrispondente a P.
Una formula $\forall xA$ è vera in una interpretazione I quando A è vera per ogni interpretazione I' che è identica a I tranne che per la corrispondenza di x.

Una formula $\exists xA$ è vera in una interpretazione I quando esiste una interpretazione I' che è identica a I tranne che per la corrispondenza di x in cui A è vera.

Teorema
I teoremi "teorema della deduzione" e "dimostrazione per contraddizione", visti nel contesto della logica proposizionale valgono in modo analogo per la logica del prim'ordine.

Definizione
Le formule in cui ogni variabile rientra nell'ambito di un quantificatore sono chiamate formule *chiuse*. Le variabili che non rientrano nell'ambito di un quantificatore sono chiamate variabili *libere*.

Esempi:
$\forall x(P(x) \Leftrightarrow Q(x))$ è una formula chiusa.
$\forall x((R(x) \Rightarrow S(x)) \wedge T(y))$ non è una formula chiusa e in essa y è una variabile libera.

Definizione
Le definizioni di "forma normale congiuntiva" (CNF) e di clausole di Horn valgono per le formule della logica del prim'ordine in maniera analoga a quanto visto nella logica proposizionale.

Definizione
Scriviamo $\varphi[x/t]$ per indicare la formula che risulta quando sostituiamo ogni ricorrenza libera della variabile x nella formula φ con il termine t.
Il termine t non può contenere alcuna variabile che sia quantificata in φ.

Esempio:
$\forall x(x = y)$ è una formula in cui y compare come variabile libera e può quindi essere sostituita.
Se la sostituissimo con il termine x + 1 avremmo
$\forall x(x = x + 1)$, ossia una formula con una semantica ben diversa da quella originale.
Questa sostituzione è illegittima perché il termine sostituente contiene la variabile x, già presente e quantificata nella formula. Una sostituzione legittima è [y / y+1] che dà luogo alla formula
$\forall x(x = y + 1)$

Una sostituzione in cui una variabile viene sostituita da un'altra variabile (es. [x / y]) si chiama anche "ridenominazione".

Quantificatori e forme normali

$\forall xP(x)$ è vera se e solo se è vera per tutte le interpretazioni della variabile x. Con questa definizione semantica in mente, possiamo scrivere, invece della formula con il quantificatore, una formula così:
$P(a_1) \wedge \ldots \wedge P(a_n)$
per tutte le costanti $a_1 \ldots a_n$ nell'insieme delle costanti K.
Al posto di $\exists xP(x)$, invece, si potrebbe scrivere
$P(a_1) \vee \ldots \vee P(a_n)$

Prendendo in considerazione le leggi di DeMorgan ($\neg(P(x) \wedge Q(x)) \equiv (\neg P(x) \vee \neg Q(x))$ e il duale) e la regola di inferenza sulla doppia negazione, notiamo che
$P(a_1) \wedge \ldots \wedge P(a_n) \equiv \neg\neg(P(a_1) \wedge \ldots \wedge P(a_n)) \equiv \neg(\neg P(a_1) \vee \ldots \vee \neg P(a_n))$
ossia che

$\forall x P(x) \equiv \neg\exists x \neg P(x)$
e anche
$\exists x P(x) \equiv \neg\forall x \neg P(x)$
Esempio:
La frase “tutte le persone vogliono essere amate” e la frase “non esiste alcuna persona che non voglia essere amata” sono logicamente equivalenti.

I quantificatori aumentano notevolmente l’espressività delle formule del linguaggio della logica del prim’ordine, ma ne rendono anche più complessa la trattazione per mezzo di strumenti automatici di calcolo. Per ovviare a questo inconveniente, cerchiamo di trasformare le formule, mantenendone la semantica, in modo da ridurre i loro quantificatori.

Definizione
Una formula della logica dei predicati φ è in “forma normale prenessa” quando si ha che
• φ ha la forma $Q_1x_1 \ldots Q_nx_n\ \psi$
• ψ è una formula senza quantificatori
• $Q_i \in \{\forall,\exists\}$ con i = 1, ..., n

Teorema
Ogni formula della logica del prim’ordine può essere trasformata in una formula equivalente in forma normale prenessa.

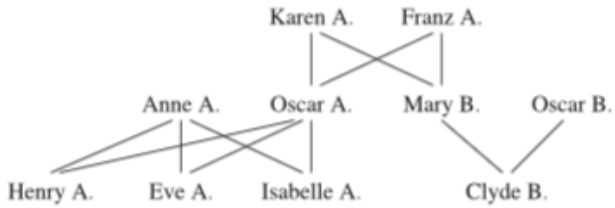
Come procedere:
- Trasformazione in forma normale congiuntiva:
  - Eliminazione delle equivalenze $\Leftrightarrow$
  - Eliminazione dei condizionali $\Rightarrow$
  - Applicazione ripetuta delle leggi di DeMorgan e della legge distributiva
- Ridenominazione delle variabili se necessario
- Spostamento dei quantificatori universali all’inizio della formula
- Skolemizzazione: sostituzione di variabili esistenzialmente quantificate con nuove funzioni di Skolem
- Omissione dei quantificatori universali (con l’assunto che tutte le variabili presenti siano universalmente quantificate)

Skolemizzazione
Inventata dal matematico norvegese Thoralf Skolem (1887 – 1963), è una tecnica per eliminare i quantificatori esistenziali che si trovano nell’ambito di quantificatori universali.
L’idea di base è questa. Data la formula:
$\forall x\ (P(x) \Rightarrow \exists y Q(x,y))$
la interpretiamo come
“*per ogni x, se x ha la proprietà P, allora esiste un y che è nella relazione Q con x.*”
È come se la variabile y esistesse in funzione di x. Esprimiamo questo legame proprio con una funzione matematica (la funzione di Skolem), in cui y è funzione di x, e riscriviamo la formula così:
$\forall x\ (P(x) \Rightarrow Q(x,f(x)))$
Il quantificatore esistenziale non serve più, perché la variable quantificata y è stata sotituita da un termine che dipende da x tramite la funzione di Skolem.
Attenzione: la nuova formula NON è equivalente a quella originale, perché facciamo una selezione di un singolo termine f(x) da mettere in relazione con x, mentre nella formula originale tale relazione poteva esistere tra x e più entità nel dominio, a seconda di come interpretiamo y.

Ad ogni modo, la formula che risulta dalla Skolemizzazione è una conseguenza logica di quella originale, quindi la possiamo usare nelle dimostrazioni per contraddizione (la negazione della formula Skolemizzata crea una contraddizione se la negazione della formula originale crea una contraddizione).

Esercizio:
Trasformare le seguenti frasi in formule della logica del prim'ordine.
a) *Tutti i cani abbaiano di notte*.
b) *Chiunque abbia un gatto non ha alcun topo.*
c) *Chi ha il sonno leggero non ha niente che abbai la notte.*
d) *Luca ha un gatto o un cane.*
e) *Se Luca ha il sonno leggero, allora Luca non ha alcun topo.*

Esercizio:
Ecco un albero genealogico



che, tradotto in logica del prim'ordine, può dare luogo alla seguente KB:

KB ≡ Female(Karen) ∧ Female(Anne) ∧ Female(Mary) ∧ Female(Eve) ∧ Female(Isabelle) ∧
Child(Oscar, Karen, Franz) ∧ Child(Mary, Karen, Franz) ∧ Child(Eve, Anne, Oscar) ∧
Child(Henry, Anne, Oscar) ∧ Child(Isabelle, Anne, Oscar) ∧ Child(Clyde, Mary, OscarB) ∧
(∀x ∀y ∀z Child(x,y,z) ⇒ Child(x,z,y)) ∧
(∀x ∀y Descendant(x, y) ⇔ ∃z Child(x, y, z) ∨ (∃u ∃v Child(x, u, v) ∧ Descendant(u, y)))

Come provare che Child(Eve, Oscar, Anne) è una conseguenza logica di questa KB?

Introduciamo regole di inferenza che si aggiungono a quelle viste in precedenza e caratterizzano la logica del prim'ordine:

Eliminazione di ∀

∀xP(x)

P[x/t]

Eliminazione di ∃

∃xQ(x)

Q[x/K]

con K nome di costante nuovo, non presente nella KB.
Attenzione: ∃ NON deve trovarsi nel campo di applicazione di un ∀.

Introduzione di ∃

Q(K)

∃xQ(x)

Dimostrazione di Child(Eve, Oscar, Anne):

1. Child(Eve, Anne, Oscar) [premessa]
2. ∀x ∀y ∀z Child(x,y,z) ⇒ Child(x,z,y) [premessa]
3. Child(Eve, Anne, Oscar) ⇒ Child(Eve, Oscar, Anne) [2 con [x/Eve, y/Anne, z/Oscar]]
4. Child(Eve, Oscar, Anne) [MP(1,3)]

c.v.d.

Teorema (Teorema di completezza di Gödel)
La logica del prim'ordine è completa, ossia, esiste un calcolo (insieme di regole di inferenza) con il quale ogni proposizione che è una conseguenza logica di una base di conoscenza può essere dimostrata.
Se KB |= α, allora KB ⊢ α.

Teorema (Correttezza)
Esistono calcoli con i quali si possono dimostrare solo formule della logica del prim'ordine vere.
Cioè, se KB ⊢ α vale, allora KB |= α.

Risoluzione nella logica del prim'ordine

Procediamo come nella logica predicativa, con negazione della tesi, trasformazione in clausole, e ricerca della clausola vuota.
Dimostrazione di Child(Eve, Oscar, Anne):

¬Child(Eve, Oscar, Anne) [tesi negata]

Da ∀x ∀y ∀z Child(x,y,z) ⇒ Child(x,z,y) [nella KB]
otteniamo ¬ Child(x,y,z) ∨ Child(x,z,y) [clausola 1]

Child(Eve, Anne, Oscar) [nella KB, già in forma di clausola, clausola 2]

Per procedere, dobbiamo arricchire la risoluzione di una regola che ci permetta di gestire i termini dei predicati (problema che non si poneva per i letterali senza variabili della logica predicativa)

Definizione
Due letterali sono chiamati "unificabili" se esiste una sostituzione σ per tutte le variabili presenti che rende uguali i letterali. Tale sostituzione σ è chiamata "unificatore". Un unificatore è chiamato "l'unificatore più generale" (MGU: *most general unifier*) se tutti gli altri unificatori possono essere ottenuti a partire da esso tramite sostituzione di variabili.

Esempio:
Dati i due letterali Child(x,y,z) e Child(Eve, Anne, Oscar), esiste una sostituzione che li rende uguali: σ = [x/Eve, y/Anne, z/Oscar]

Definizione
La risoluzione tra due clausole in forma normale congiuntiva nella logica del prim'ordine si esegue come segue:

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), \quad (\neg B' \vee C_1 \vee \cdots \vee C_n) \quad \sigma(B) = \sigma(B')}{(\sigma(A_1) \vee \cdots \vee \sigma(A_m) \vee \sigma(C_1) \vee \cdots \vee \sigma(C_n))}$$

dove σ è l'unificatore più generale di *B* e *B'*.

Attenzione: la sostituzione che unifica i due letterali va applicata anche agli altri che fanno parte delle clausole con cui si esegue la risoluzione.

Quindi possiamo proseguire con la dimostrazione:
¬Child(Eve, Oscar, Anne) [tesi negata]
¬Child(x,y,z) ∨ Child(x,z,y) [clausola 1]
Child(Eve, Anne, Oscar) [clausola 2]
Child(Eve, Oscar, Anne) [clausola 3, per risoluzione tra 1 e 2, con σ = [x/Eve, y/Anne, z/Oscar]]
{ } [risoluzione tra la tesi negata e la clausola 3] c.v.d.

Teorema
La regola della risoluzione è corretta: la clausola risultante è una conseguenza logica delle due clausole su cui la risoluzione viene eseguita.

Per la completezza serve un'ulteriore precisazione.

Il paradosso del barbiere
Il famoso paradosso di Russell (filosofo e matematico inglese, 1872 – 1970) recita:
"C'è un barbiere che rade tutti quelli che non si radono".
Il paradosso emerge quando ci si chiede se questo barbiere si rade o meno.
Questa affermazione è contraddittoria, nel senso che è insoddisfacibile. Vogliamo dimostrarlo con la risoluzione.

La frase tradotta in formula del prim'ordine diventa:
∀x (Rade(Barbiere,x) ⇔ ¬Rade(x,x))
che, trasformata in clausole, dà il seguente insieme di clausole:
¬Rade(Barbiere,x) ∨ ¬Rade(x,x) [1]
Rade(x,x) ∨ Rade(Barbiere,x) [2]
applicando la risoluzione a queste clausole otteniamo numerose tautologie (es. Rade(x,x) ∨ ¬Rade(x,x)) ma nessuna contraddizione.
Ci serve una regola aggiuntiva.

Definizione
Si chiama "fattorizzazione" la seguente manipolazione sintattica di una clausola:

$$\frac{(A_1 \vee A_2 \vee \cdots \vee A_n) \quad \sigma(A_1) = \sigma(A_2)}{(\sigma(A_2) \vee \cdots \vee \sigma(A_n))}$$

in cui se esistono due letterali che sono unificati da una sostituzione σ, allora possiamo derivare una clausola in cui applichiamo σ a tutti i letterali e un membro della coppia di letterali identici viene omesso.

Quindi, prima di procedere alla risoluzione, eseguiamo la fattorizzazione.
$\neg$Rade(Barbiere,x) $\vee$ $\neg$Rade(x,x) [1]
Rade(x,x) $\vee$ Rade(Barbiere,x) [2]
$\neg$Rade(Barbiere,Barbiere) [clausola 3, per fattorizzazione su 1, con σ = [x/Barbiere]
Rade(Barbiere,Barbiere) [clausola 4, per fattorizzazione su 2, con σ = [x/Barbiere]
{ } [risoluzione tra 3 e 4] c.v.d.

Teorema
La regola di risoluzione insieme alla regola di fattorizzazione è completa per le dimostrazioni per refutazione. Cioè, applicando passaggi di fattorizzazione e risoluzione, la clausola vuota può essere derivata da qualsiasi insieme insoddisfacibile di formule in forma normale congiuntiva.

Strategie di risoluzione

Sebbene la completezza della risoluzione sia importante per l'utente, la ricerca di una prova può essere molto frustrante nella pratica. La ragione di ciò è l'immenso spazio di ricerca combinatoria. Anche se all'inizio ci sono solo pochissime coppie di clausole in KB $\wedge$ $\neg$Q, il dimostratore (umano o automatico) genera una nuova clausola ad ogni passo di risoluzione, il che aumenta il numero di possibili passi di risoluzione nell'iterazione successiva. Pertanto è stato a lungo tentato di ridurre lo spazio di ricerca utilizzando strategie speciali, preferibilmente senza perdere la completezza.
Le strategie più importanti sono le seguenti.

- La risoluzione *unitaria* dà la priorità ai passaggi di risoluzione in cui una delle due clausole è costituita da un solo letterale, chiamato clausola unitaria. Questa strategia preserva la completezza e porta in molti casi, ma non sempre, a una riduzione dello spazio di ricerca. Si tratta quindi di un processo euristico.
- Si ottiene una riduzione garantita dello spazio di ricerca mediante l'applicazione della strategia *dell'insieme di supporto*. Un sottoinsieme di KB $\wedge$ $\neg$Q è definito come insieme di supporto (SOS: *set of support*): ogni passo di risoluzione deve coinvolgere una clausola in SOS e il risultato del passo viene aggiunto al SOS. Questa strategia è incompleta. Diventa completa quando ci si assicura che l'insieme di clausole senza il SOS sia soddisfacibile. La tesi negata $\neg$Q viene spesso utilizzata come SOS iniziale.
- Nella strategia della *risoluzione dell'input*, una clausola del set di input KB $\wedge$ $\neg$Q deve essere coinvolta in ogni passo di risoluzione. Questa strategia riduce anche lo spazio di ricerca, ma a scapito della completezza.
- Con la regola del *letterale puro* tutte le clausole che contengono letterali per i quali non ci sono letterali complementari in altre clausole possono essere eliminate. Questa regola riduce lo spazio di ricerca ed è completa, quindi viene utilizzata praticamente da tutti i programmi di risoluzione.
- Se i letterali di una clausola C1 rappresentano un sottoinsieme dei letterali della clausola C2, è possibile eliminare C2. Ad esempio, la clausola $\neg$A$\vee$B è ridondante se B è già presente nella KB. Questo passaggio di riduzione è chiamato *sussunzione*. Anche la sussunzione è completa.

Esercizio (continuazione da prima)
Negare la formula derivata dalla frase (e) e trasformare tutte le formule risultanti in clausole in forma normale congiuntiva. Dimostrare che l'insieme delle formule è inconsistente, e che quindi…