Vero o falso?

- Si consideri la formula:

Piove ∧ Vento

- È vera oppure falsa?

Vero o falso?

- Si consideri la formula:

Piove ∧ Vento

- È vera oppure falsa?

Piove	Vento
T	T
T	F
F	T
F	F

?

Semantica della logica proposizionale

- Premesse:
  - La semantica definisce le regole con cui si calcolano i valori di verità di tutte le formule
  - Nella logica proposizionale i valori di verità delle formule sono calcolati a partire da un modello e un modello è un assegnamento di un valore di verità a ciascuno dei simboli proposizionali specificati
  - N simboli proposizionali, che possono valere vero o falso, possono produrre 2^N *diversi modelli*
  - La semantica è calcolata ricorsivamente

Regole per il calcolo della semantica

- Le formule atomiche
  - **True** e **False** sono rispettivamente vera e falsa in ogni modello
  - I valori di verità di tutti gli altri **simboli proposizionali** vanno specificati esplicitamente dal modello
- Formule complesse
  Siano P e Q due formule della logica proposizionale:
  - ¬ Q: è vera se e solo se Q è falsa nel modello
  - (Q ∧ P): è vera se e solo se sia P che Q sono vere nel modello
  - (Q ∨ P): è vera se e solo se o P o Q è vera nel modello
  - (Q ⇒ P): è sempre vera a meno che P sia vera e Q falsa nel modello
  - (Q ⇔ P): è vera se P e Q sono entrambe vere o entrambe false nel modello
- Per le formule complesse si possono usare le tabelle di verità degli operatori

Tabelle di verità degli operatori

P	Q	¬P	P∧Q	P∨Q	P⇒Q	P⇔Q
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	V	T	T	T

Implicazione

P⇒Q

E' equivalente a ¬ P ∨ Q
Va interpretata nel seguente modo:

- se P è falsa, non mi interessa affermare il valore di Q;
- se P è vera, affermo che anche Q è vera
- Quindi Q deve essere vera nei casi in cui P lo è. Quando P non lo è la verità di Q non interessa.

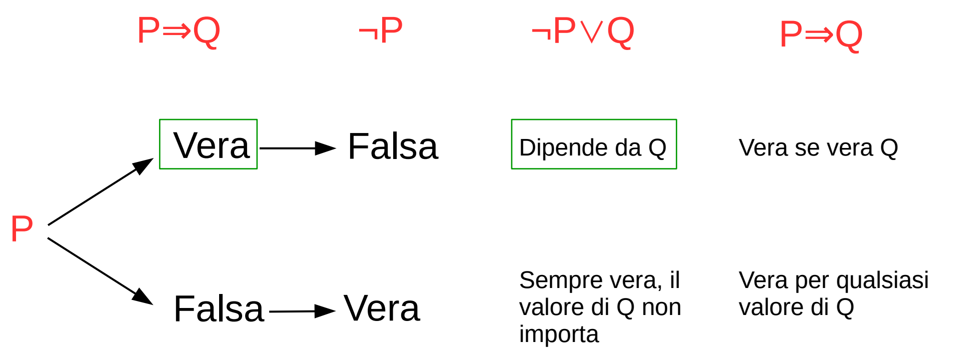
P	Q	⇒
F	**F**	**T**
F	**T**	**T**
T	F	F
T	**T**	**T**

Esempio: background knowledge proposizionale

R1) Piove ⇒ Atmosfera_umida (¬ Piove ∨ Atmosfera_umida)

Piove	Atmosfera_umida	⇒
T	T	T
T	F	F
F	T	T
F	F	T

Spiegazione dell'implicazione



Serve per catturare quelle situazioni in cui il conseguente è vero ogni volta che lo è l'antecedente

Altri tipi di implicazione

- Esistono molti tipi di implicazione
- L' **implicazione logica** è un tipo di relazione che dipende dalle leggi della logica (non dipende dal significato delle parole)
- **Altri tipi di implicazione** dipendono dal significato delle parole:
  - Fido è un cane → fido è un mammifero
  - John ha vinto la partita → John ha giocato la partita
  - John è stato condannato per furto → il furto è un crimine

Tanti tipi di implicazione

- Esistono molti tipi di implicazione
- L' implicazione logica è un tipo di relazione che dipende dalle leggi della logica (non dipende dal significato delle parole)
- Altri tipi di implicazione dipendono dal significato delle parole:
  - Fido è un cane → fido è un mammifero
    ragionamento ontologico
  - John ha vinto la partita → John ha giocato la partita
    ragionamento temporale
  - John è stato condannato per furto → il furto è un crimine
    ragionamento causale

L'implicazione non è una relazione causale

$P \Rightarrow Q$

Esempi:

l'implicazione può essere vera in casi poco intuitivi perché istintivamente le attribuiamo una valenza causale, quindi ci sembra sensato:

piove ⇒ strada_bagnata

Mentre ci sembra assurdo:

Torino è in Lombardia ⇒ Giulio Cesare governò Roma

Invece questa implicazione è vera perché la premessa è falsa

Biimplicazione

$P \Leftrightarrow Q$

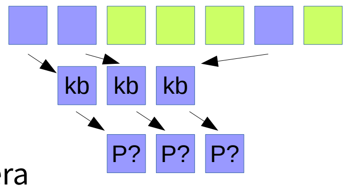
Viene anche letta "se e solo se" perché è vera quando P e Q hanno lo stesso valore

Conseguenza logica?

Come dimostrare che KB ⊨ P?

1) Model Checking:

- Enumero i possibili modelli
- Seleziono quelli in cui KB è vera
- Verifico che in tutti questi P sia vera
- **Costoso**: dati N simboli proposizionali esistono 2^N modelli



2) Theorem proving:

- Permette di usare regole di inferenza per cercare una derivazione, senza costruire i modelli (più efficiente perché ignora le proposizioni irrilevanti, che possono essere numerose)

Theorem Proving: KB ⊨ P?

- È possibile grazie a due risultati fondamentali:
  - **Teorema di deduzione**
    permette di rispondere vero se si dimostra l' equivalenza
    (KB ⇒ P) ≡ True
  - **Dimostrazione per refutazione**
    permette di rispondere vero se si dimostra l' equivalenza
    (KB ∧ ¬ P) ≡ False

Teorema di deduzione

Date due formule R e Q, **(R ⊨ Q)** se e solo se **(R ⇒ Q) è valida**

- cioè: "Q è conseguenza logica di R" **se e solo se** "R implica Q" è valida (cioè è una tautologia)
- Quindi per verificare che KB ⊨ P:
  1) Posso dimostrare che KB ⇒ P è una *formula valida*, cioè vera in ogni modello, enumerando i modelli (costoso)
  2) Equivalentemente, per definizione di tautologia, posso dimostrare per **inferenza sintattica** (cioè manipolando la forma delle formule, senza costruire i modelli) che (KB ⇒ P) ≡ True
     - qual è il procedimento?

Validità e soddisfacibilità

- **Validità** e **soddisfacibilità** sono concetti collegati dalla negazione, in particolare:
  - **A è valida se e solo se ¬ A è insoddisfacibile**
  - A è soddisfacibile se e solo se ¬ A non è valida

Dimostreremo la validità (1) applicando la negazione e (2) dimostrando l' insoddisfacibilità della formula ottenuta

- (R ⇒ Q) equivale a (¬ R ∨ Q)
- negata diventa ¬ (¬ R ∨ Q)
- che è equivalente a (R ∧ ¬ Q) per le leggi di De Morgan

Dimostrazione per refutazione

Date due formule R e Q, (R ⊨ Q) se e solo se (R ∧ ¬ Q) è insoddisfacibile

- È stata ottenuta ricordando che una contraddizione è la negazione di una tautologia
- Corrisponde a una *dimostrazione per refutazione (o per assurdo o per contraddizione):*

Per verificare che KB ⊨ P:

1) Assumo (per assurdo) ¬ P
2) dimostro che KB ∧ ¬ P è *insoddisfacibile,* cioè che ¬ P è in contraddizione con gli assiomi noti, cioè che partendo da KB ∧ ¬ P si dimostra False
3) La dimostrazione è del tutto analoga a una ricerca in uno spazio degli stati

Inferenza e dimostrazioni

- Dalle premesse, applicare una sequenza di passi per raggiungere una determinata conclusione
- **Formulazione come problema di ricerca**:
  - **Stato iniziale**: background knowledge
  - **Azioni**: regole di inferenza
  - **Goal**: stato che contiene la formula da dimostrare
- Importante: ci focalizziamo su **logiche monotóne**, cioè tali che:
  - **Se (KB ⊨ P) allora (KB ∧ Q ⊨ P)**
    cioè l'aggiunta di informazione (Q) non invalida mai le conclusioni precedenti, in altri termini l'insieme delle formule conseguenti può solo crescere con l'aggiunta di informazione

Regole di inferenza

- Abbiamo già citato modus ponens ed eliminazione dei congiunti. Altre regole di inferenza sono date dalle seguenti equivalenze logiche:

$(\alpha \lor \beta)$	≡	$(\beta \lor \alpha)$	Commutatività ∨
$(\alpha \land \beta)$	≡	$(\beta \land \alpha)$	Commutatività ∧
$((\alpha \land \beta) \land \gamma)$	≡	$(\alpha \land (\beta \land \gamma))$	Associatività ∧
$((\alpha \lor \beta) \lor \gamma)$	≡	$(\alpha \lor (\beta \lor \gamma))$	Associatività ∨
$\neg(\neg\alpha)$	≡	α	Elim. doppio negato
$(\alpha \Rightarrow \beta)$	≡	$(\neg\beta \Rightarrow \neg\alpha)$	contrapposizione
$(\alpha \Rightarrow \beta)$	≡	$(\neg\alpha \lor \beta)$	Elim. implicazione
$\neg(\alpha \land \beta)$	≡	$(\neg\alpha \lor \neg\beta)$	De Morgan
$\neg(\alpha \lor \beta)$	≡	$(\neg\alpha \land \neg\beta)$	De Morgan
$(\alpha \land (\beta \lor \gamma))$	≡	$((\alpha \land \beta) \lor (\alpha \land \gamma))$	distributività
$(\alpha \lor (\beta \land \gamma))$	≡	$((\alpha \lor \beta) \land (\alpha \lor \gamma))$	distributività
$(\alpha \Leftrightarrow \beta)$	≡	$((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$	Elim. bicondizionale

Correttezza e completezza

- La dimostrazione avviene coniugando:
  1) un algoritmo di inferenza
  2) e un insieme di regole di inferenza
- L'insieme di tutte le regole di inferenza viste è corretto (derivano solo conseguenze vere) e completo (deriva tutte le conseguenze logiche)
- Sottoinsiemi di queste regole possono non riuscire a produrre tutte le inferenze lecite; in questi casi il metodo non sarà completo
  - Es. se non metto la regola del doppio negato non posso derivare: P ⊢ ¬¬ P (e quindi che P ⊨ ¬¬ P)

Regola di risoluzione

- **Risoluzione**:
  - regola di inferenza che
  - unita a qualsiasi algoritmo di ricerca completo (cioè tale per cui se esistono soluzioni ne trova una)
  - produce un algoritmo di inferenza corretto (cioè tale per cui se KB ⊢ P allora KB ⊨ P) e completo (cioè tale per cui se KB ⊨ P allora KB ⊢ P)

Regola di risoluzione (o Resolution)

- La resolution permette di realizzare dimostrazioni per refutazione sia in logica proposizionale che in logica del prim' ordine
- **Regola di risoluzione:**

$$\frac{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_i \vee P_{i+1} \vee \ldots \vee P_n \qquad Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_j \vee Q_{j+1} \vee \ldots \vee Q_m}{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_{i+1} \vee \ldots \vee P_n \vee Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_{j+1} \vee \ldots \vee Q_m}$$

Resolution

- Regola di resolution:

NB: i due letterali P_i e Q_j sono complementari (uno è la negazione dell'altro)

$$\frac{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_i \vee P_{i+1} \vee \ldots \vee P_n \qquad Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_j \vee Q_{j+1} \vee \ldots \vee Q_m}{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_{i+1} \vee \ldots \vee P_n \vee Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_{j+1} \vee \ldots \vee Q_m}$$

La formula derivata è detta **resolvent**, in essa *ogni letterale compare una volta sola*

Tutte le formule convolte sono **clausole**, cioè sono disgiunzioni di letterali

Resolution

- Regola di resolution:

NB: i due letterali P_i e Q_j sono complementari (uno è la negazione dell'altro)

$$\frac{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_i \vee P_{i+1} \vee \ldots \vee P_n \qquad Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_j \vee Q_{j+1} \vee \ldots \vee Q_m}{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_{i+1} \vee \ldots \vee P_n \vee Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_{j+1} \vee \ldots \vee Q_m}$$

La formula derivata è detta **resolvent**, in essa *ogni letterale compare una volta sola (**FATTORIZZAZIONE**), esempio:*

A ∨ A diventa A

Esempio

Letterali complementari

$$\frac{A \vee B \vee \neg C \qquad C \vee A}{A \vee B}$$

La rimozione dei letterali complementari produce il risolvente: A ∨ B ∨ A che si semplifica nelle'equivalente A ∨ B applicando la fattorizzazione

Relazione con il modus ponens

$$\frac{B \vee C \qquad \neg C \vee A}{A \vee B}$$

Il modus ponens è un caso speciale di risoluzione

Lo si evidenzia tramite l'eliminazione dell'implicazione

$$\frac{C \qquad C \Rightarrow A}{A}$$

Modus Ponens

$$\frac{C \qquad \neg C \vee A}{A}$$

Modus ponens dopo l'eliminazione dell'implicazione

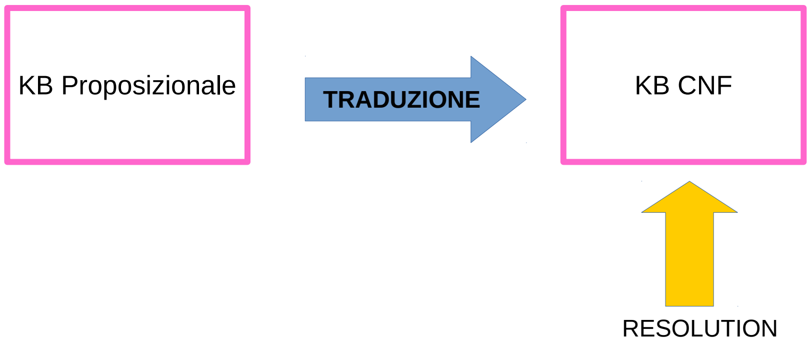
Agente guidato dalla conoscenza e inferenza

```
Agente ha: KB
Tempo = 0

Function KB-Agent(percezione) returns azione
{
1.   tell(KB, costruisci-formulaP(percezione, tempo))
2.   risposta   ask(KB, costruisci-interrogazioneA(tempo))
3.   tell(KB, costruisci-formulaA(azione, tempo))
4.   tempo   tempo + 1
5.   return risposta
}
```

1. L'agente sia dotato di una KB iniziale
2. La KB viene aggiornata con l'aggiunta di fatti che dipendono dalla "percezione" e dalle "azioni" eseguite
3. Ask interroga la KB per ottenere l 'azione da eseguire: questa richiesta attiva un processo di inferenza in cui la query, negata, viene aggiunta alla KB e, applicando iterativamente la resolution, viene ottenuta la risposta

Prerequisito: KB in Conjunctive Normal Form



Formule proposizionali e clausole

- CNF: conjunctive normal form
  data una qualsiasi formula proposizionale esiste una congiunzione di clausole ad essa equivalente

GRAMMATICA DELLE CLAUSOLE

- CNFsentence → Clause ∧ … ∧ Clause
- Clause → Literal ∨ … ∨ Literal
- Literal → Symbol | ¬Symbol
- Symbol → P | Q | …

Esempi e controesempi

- $\neg A \land (B \lor C)$
- $(A \lor B) \land (\neg B \lor C \lor \neg D) \land (D \lor \neg E)$
- $A \lor B$
- $A \land B$

Clausole

- $\neg(B \lor C)$
- $(A \land B) \lor C$
- $A \land (B \lor (D \land E))$.

Formule proposizionali che sembrano ma non sono clausole

https://en.wikipedia.org/wiki/Conjunctive_normal_form

Formule proposizionali e clausole

Algoritmo di traduzione in clausole

1) Eliminare la biimplicazione: ((α ⇒ β) ∧ (β ⇒ α))

2) Eliminare l' implicazione: (¬α ∨ β)

3) Portare il not all' interno (De Morgan ed eliminazione della doppia negazione):

- (¬α ∨ ¬β) oppure (¬α ∧ ¬β)
- α

1) Distribuire l' or sull' and dove possibile: ((α ∨ β) ∧ (α ∨ γ))

Formule proposizionali e clausole

Algoritmo di traduzione in clausole

1) Eliminare la biimplicazione
2) Eliminare l' implicazione
3) Portare il not all' interno (De Morgan ed eliminazione della doppia negazione)
4) Distribuire l' or sull' and dove possibile

- $\neg(B \lor C)$
- $(A \land B) \lor C$
- $A \land (B \lor (D \land E))$.

⇒

¬ B ∧ ¬ C

(A ∨ C) ∧ (B ∨ C)

A ∧ (B ∨ D) ∧ (B ∨ E)

Esempio: da formule a clausole

La KB proposizionale vista tradotta in clausole:

C1) ¬ Piove ⋁ Atmosfera_umida

C2) ¬ Notte ⋁ Vento or Atmosfera_umida

C3a) ¬ Atmosfera_umida ⋁ Prato_bagnato

C3b)¬ Atmosfera_umida ⋁ Strada_Bagnata

C4) ¬ Innaffiatore_on ⋁ Prato_bagnato

C5) ¬ Piove ⋁ Ombrello_aperto

C6) ¬ Sole ⋁ ¬ Vento or Innaffiatore_on

C7) ¬ Sole ⋁ ¬ Vento ⋁ Atmosfera_asciutta

C8) ¬ Sole ⋁ ¬ Notte

C10) ¬ Atmosfera_asciutta ⋁ ¬ Atmosfera_umida

La traduzione in clausola della R8) e della R9) producono esattamente la stessa clausola (in particolare la C8)