Rappresentazione della conoscenza

Si studia come rappresentare la conoscenza in modo tale che sia possibile applicarvi dei processi di ragionamento automatici (inferenze) per derivare informazioni, nuova conoscenza e per prendere decisione – in particolare per decidere quale azione eseguire

Agenti basati sulla conoscenza

Sono caratterizzati nel seguente modo:

- Knowledge base (KB): un insieme di formule espresse in un linguaggio per la rappresentazione della conoscenza possedute dall' agente. Può cambiare nel tempo. La conoscenza iniziale è detta *background knowledge*
- Tell (o assert):Un meccanismo per aggiungere nuove formule
- Ask (o query): Un meccanismo per effettuare interrogazioni
- Sia ask che tell possono attivare *processi di inferenza* e devono soddisfare la proprietà:
  - *Ogni risposta ad una ask deve deve essere una conseguenza delle asserzioni (tell) fatte e della conoscenza di background*
  - *Esempio:*
    - *KB: so che quando piove la strada è bagnata*
    - *tell(piove)*
    - *ask(strada bagnata)?*
    - *La risposta deve essere Yes.*

Schema dell'agente

```
Agente ha: KB
Tempo = 0

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

1. Si suppone che l'agente sia dotato di una KB iniziale
2. La prima tell aggiorna la KB con la percezione corrente (tempo è inizializzato a 0 e permette di mantenere una storia). La percezione è tradotta in formula da costruisci-formulaP
3. Ask interroga la KB per ottenere l 'azione da eseguire
4. La seconda tell aggiorna la KB con l'azione eseguita
5. La funzione restituisce l'azione

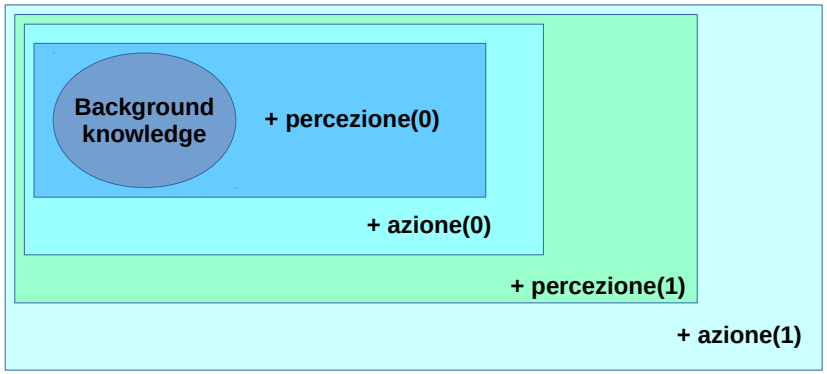
Schema dell'agente, versione 2

```
Agente ha: KB
Tempo = 0

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

1. Add corrisponde a tell: arricchisce la KB
2. Modify può sia aggiungere che rimuovere elementi dalla KB
3. Esempio: se KB contiene il fatto "il bicchiere è vuoto" l'azione "riempi il bicchiere" cancellerà questo fatto ed aggiungerà "bicchiere pieno"

KB modificata dalla storia



Azioni e successive percezioni modificano la KB

Percezione (dati), informazione e conoscenza

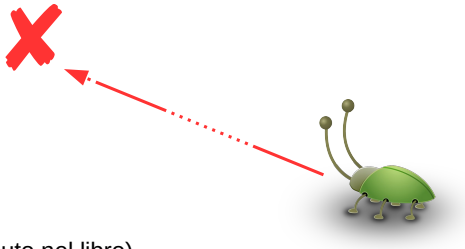
Dato: il disegno che vedete qui a fianco è il risultato della vostra percezione sensoriale. Non ha un significato.

Informazione: è ciò che il dato rappresenta.
Per esempio quel simbolo è la lettera 'u' (pronunciata lunga) dell'alfabeto degli Inuit. L'informazione in parte è legata allo scopo per cui si percepisce.

Conoscenza: cattura relazioni. Quella lettera può essere composta con altre dello stesso alfabeto, secondo determinate regole per produrre parole e frasi.

Esempio, bug algorithm

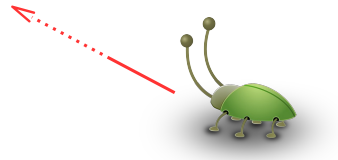
- Semplice agente con conoscenza solo locale dell' ambiente
- In grado di procedere in linea retta e di aggirare un ostacolo applicando un comportamento di wall following (ruota a caso verso destra o sinistra e procede tenendosi parallelo al muro)
- In robotica realizzabile usando solo sensori di contatto
- Opzionalmente l' agente potrebbe aver una destinazione-obiettivo, ma ciò richiede di equipaggiarlo con la capacità di allinearsi con quest' ultimo



(Esempio non contenuto nel libro)

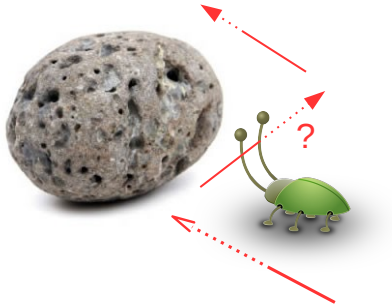
Esempio, bug algorithm

- Percezione ∈ { ostacolo, libero, allineato, arrivato }
- Azioni ∈ { avanza, ruota }
  - KB_0: non arrivato
  - Tell_0: libero(0)
  - Azione: avanza
  - Tell_1: avanza(0)
  - KB_1: non arrivato, libero(0), avanza(0)
  - ...



Esempio, bug algorithm

- Percezione ∈ { ostacolo, libero, allineato, arrivato }
- Azioni ∈ { avanza, ruota }
  - KB_i: non arrivato, ...
  - Tell_i: ostacolo(i)
  - Azione: ruota
  - Tell_{i+1}: ruota(i)
  - KB_1: non arrivato, ..., ruota(i)
  - ...



Programmazione di agenti basati sulla conoscenza

- Si effettua **specificando la KB** che serve
- KB comprende la specifica delle azioni
- La KB è data in **forma dichiarativa** (cosa e non come)
- l' agente è equipaggiato di *meccanismi generali* che permettono di effettuare ask e tell su qualsiasi KB

- **NB**: paradigma ben diverso da quello procedurale dove tutto è codificato da procedimenti specifici per il dominio!!

Programmazione dichiarativa

- La programmazione dichiarativa è un paradigma di programmazione in cui *i programmi esprimono la logica di una computazione senza esprimerne il flusso di controllo*
- Un programma è dichiarativo quando descrive *cosa* la computazione dovrebbe fare ma *non come* effettuarla
- **Esempi**:
  - XML dice come è fatta una pagina web non come visualizzarla (farne il rendering)
  - Una query SQL dice quali informazioni estrarre non come percorrere le tabelle per reperire e combinare i dati
  - un' espressione regolare dice la forma di una sequenza di caratteri non come effettuarne la ricerca in uno stream di input

XML: esempio

```
<note>
      <to>Beppe</to>
      <from>Milo</from>
      <heading>Pro memoria</heading>
      <body>
        Non dimenticarti della riunione di lunedì!
      </body>
</note>
```

Da: Beppe
A: Milo
Oggetto: Pro memoria

```
      Non dimenticarti della riunione di lunedì!
```

SQL query: esempio

```
SELECT * FROM STATION
WHERE LAT_N > 39.7;
```

Viene percorsa la tabella STATION (che contiene descrizioni di stazioni ferroviarie)
alla ricerca di tutti I record per I quali il campo LAT_N (latitudine nord) è maggiore del valore indicato

ID	CITY	STATE	LAT_N	LONG_W
44	Denver	CO	40	105
66	Caribou	ME	47	68

Espressioni regolari: esempio

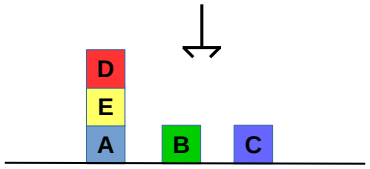
```
(10)*111(10)+
```

Descrive (permette di generare o di riconoscere) sequenze di 0 e 1 che iniziano con una sequenza di coppie 10 eventualmente vuota, contengono una sola occorrenza di 111 e terminano con una sequenza non vuota di coppie 10

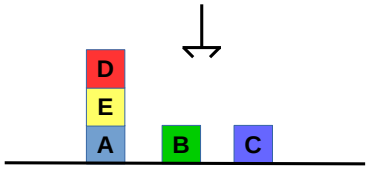
10101011110	**SÌ**
111	**NO**

Mondo dei blocchi

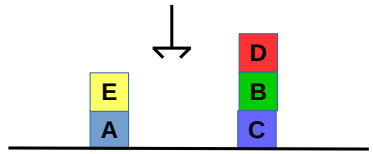
- Su un tavolo sono posizionati blocchi di legno (in generale di diversa forma e dimensione, per noi cubi di identiche dimensioni).
- L'obiettivo è costruire una o più torri posizionandoli uno sull'altro.
- Solo un blocco per volta può essere mosso o posizionandolo o sopra al tavolo o sopra a un altro che non ha altri blocchi sovrapposti su di lui.
- L'ambiente è descritto tramite i predicati: **holding(x), handempty, clear(x), ontable(x), on(x,y)**



Mondo dei blocchi



Descrizione della situazione iniziale:
ontable(A), ontable(B), ontable(C),
handempty, on(E,A), on(D,E),
clear(D), clear(B), clear(C)

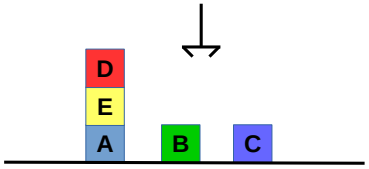


Descrizione della situazione obiettivo:
ontable(A), ontable(C),
handempty, on(B,C), on(D,B),
clear(D), clear(E)

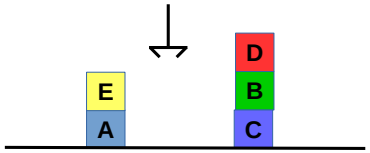
KB: azioni con condizioni di applicabilità ed effetti, esempio

pick (x, y)
Preconditions:
on (x, y) and clear(x) and handempty
Effects:
add(clear(y))
add (holding (x))
delete (on (x, y))
delete (clear(x))
delete (handempty)

Mondo dei blocchi



Descrizione della situazione iniziale:
ontable(A), ontable(B), ontable(C),
handempty, on(E,A), on(D,E),
clear(D), clear(B), clear(C)



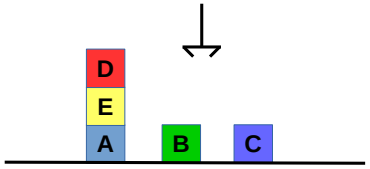
Descrizione della situazione obiettivo:
ontable(A), ontable(C),
handempty, on(B,C), on(D,B),
clear(D), clear(E)

KB: azioni con condizioni di applicabilità ed effetti, esempio

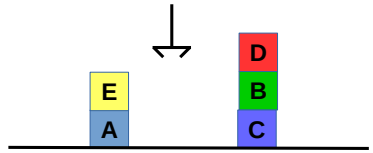
pick (x, y)
Preconditions:
on (x, y) and clear(x) and handempty
Effects:
add(clear(y))
add (holding (x))
delete (on (x, y))
delete (clear(x))
delete (handempty)

caratteristiche degli stati del mondo in cui l'azione pick è eseguibile (NB è una descrizione)

Mondo dei blocchi



Descrizione della situazione iniziale:
ontable(A), ontable(B), ontable(C),
handempty, on(E,A), on(D,E),
clear(D), clear(B), clear(C)



Descrizione della situazione obiettivo:
ontable(A), ontable(C),
handempty, on(B,C), on(D,B),
clear(D), clear(E)

KB: azioni con condizioni di applicabilità ed effetti, esempio

pick (x, y)
Preconditions:
on (x, y) and clear(x) and handempty
Effects:
add(clear(y))
add (holding (x))
delete (on (x, y))
delete (clear(x))
delete (handempty)

Modifiche alla descrizione dello stato del mondo comportate dall'esecuzione dell'azione pick. In questo linguaggio add = tell, delete ha l'effetto opposto a tell, rimuove elementi descrittivi

Mondo dei blocchi

- Background knowledge: descrizione delle azioni
- Percezione: stato corrente (stato iniziale)
- Metodo: applicazione di un criterio generale, cioè non legato allo specifico dominio del discorso, per determinare l' azione successiva

Knowledge Representation tramite formalismi logici

Logica

- **Linguaggio di rappresentazione**:
  è lo strumento che consente di rappresentare la conoscenza in una forma su cui è possibile applicare forme di ragionamento automatico
- Una formula che segue le regole del linguaggio è **ben formata**
- **Semantica del linguaggio**: definisce la verità delle formule rispetto a un mondo possibile

Logica

- Modello
- Conseguenza
- Inferenza
- Algoritmo di inferenza
- Grounding

Modello

- **Modello = mondo possibile**
- Un modello fissa i valori di verità delle formule, quindi i modelli possibili sono definiti da tutti i modi in cui è possibile assegnare valori agli elementi che determinano il valore di verità delle formule,
- Esempio consideriamo la formula x + y = 4:
  - Vera nei modelli x = 1, y = 3 oppure x = 2, y = 2 oppure x = 0, y = 4, …
  - Falsa nei modelli x = 0, y = 0 oppure x = 1, y = 0 oppure ...
- Quando l' universo di riferimento è fisico (reale), il modello è un' astrazione matematica (simbolica) significativa di quella realtà
- Dati un modello m e una formula α:
  m è un modello di α se α è vera in m
- Indichiamo con M(α) l' insieme dei modelli di α

Conseguenza (implicazione logica)

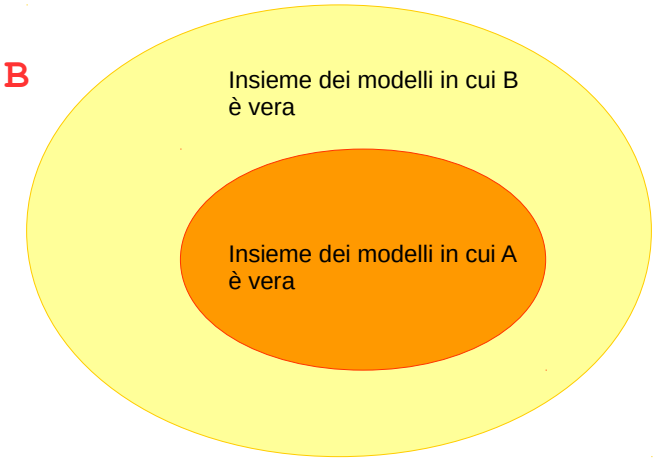
- **Conseguenza logica**: è una relazione fra due formule che dice che in tutti i modelli in cui la prima formula è vera, è vera anche la seconda, il fatto che da A consegue B è denotato: $A \models B$
- Esempio: $(x+y=4) \models (x+y<5)$
- Attenzione al significato di $A \not\models B$ che coinvolge una nozione di direzionalità ...

Conseguenza (implicazione logica)

- **A** $\models$ **B** coinvolge una nozione di direzionalità:
  - il fuoco è posto sui modelli in cui A è vera, in almeno alcuni di questi B è falsa
  - (x+y=4) $\models$ (x<3) infatti:
    - Nel modello x=4, y=0 la prima formula è vera e la seconda è falsa
    - Non è rilevante che esista il modello x=2, y=8 in cui la prima formula è falsa ma la seconda è vera
  - Il fatto che B non sia sempre vera in tutti i modelli in cui è vera A **non vuol dire che B sia sempre falsa**.
    Possono esistere modelli in cui B è vera e A falsa, esempio:
    - (x+y=4) $\models$ (x>4) and (y >0): ad esempio B è vera nel modello x=5, y=11

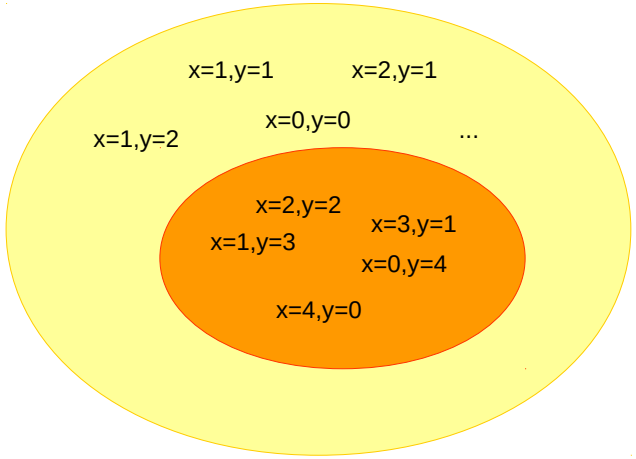
Visione insiemistica

A $\models$ **B**



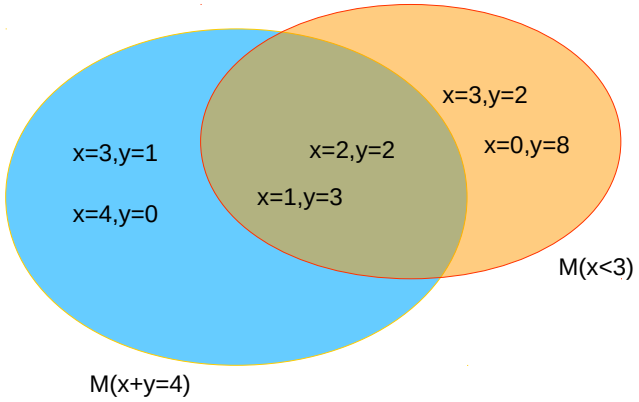
Visione insiemistica

(x+y=4) $\models$ (x+y<5)



Visione insiemistica

(x+y=4) $\models$ (x<3)



Equivalenza

A ≡ B se e solo se A ⊨ B e B ⊨ A

In altri termini due formule sono equivalenti quando sono vere negli stessi modelli. Insiemisticamente: M(A) = M(B)

Verità delle formule

- Validità
- Insoddisfacibilità
- Soddisfacibilità

Validità (tautologia)

- Una formula P è **valida** se è vera in tutti i modelli
- *True* è una formula valida
- Sono valide tutte le formule P tali che: P ≡ True
- Esempio
  - Q ∨ ¬ Q è valida, si dice anche che è una tautologia

Insoddisfacibilità (contraddizione)

- Una formula P è **insoddisfacibile** se è falsa in tutti i modelli
- *False* è una formula insoddisfacibile
- Sono insoddisfacibili (contraddizioni) tutte le P tali che: P ≡ False
- Esempio: Q ∧ ¬ Q è una contraddizione per come sono definiti gli operatori
- ***NB: P è valida se e solo se ¬ P è insoddisfacibile***

Soddisfacibilità

- Una formula P è **soddisfacibile** se esiste qualche modello in cui è vera
- Quando una formula P è vera nel modello m, si dice che:
  - *m soddisfa P*
  - o anche che *m è un modello di P*
- **Esempio**:
  nei CSP si cerca un modello (assegnamento di valori a variabili) tale per cui i suoi vincoli risultano tutti veri

Inferenza

Inferenza *(dal latino "in ferre", portare dentro): è il processo con il quale da una proposizione, accolta come vera, si passa a una seconda proposizione la cui verità è derivata dalla prima*

***NB**: l'inferenza è sintattica, lavora sulla struttura delle formule secondo il linguaggio di rappresentazione scelto*

Esempio

Supponendo vero che "tutti gli uomini sono mortali" e che "Socrate è un uomo" posso inferire che "Socrate è mortale".

Posso farlo perché "vedo" un collegamento catturabile come regola generale di ragionamento:

$$\frac{\text{uomo}(X) \supset \text{mortale}(X) \qquad \text{uomo}(X)}{\text{mortale}(X)}$$

Regola di inferenza: modus ponens

$$\frac{A \supset B \qquad A}{B}$$

MODUS PONENS:
Da un'implicazione e dalla sua premessa, derivo la conseguenza

Lavora sulla struttura delle formule: cioè $A \supset B$ e A sono supposte vere

È il fondamento del ragionamento deduttivo:

Se piove, allora la strada è bagnata.

Piove.

La strada è bagnata

Regola di inferenza: eliminazione degli and

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

ELIMINAZIONE DEI CONGIUNTI:
Da una congiunzione posso derivare un qualsiasi congiunto

Ancora una volta lavoriamo sulla struttura delle formule: cioè $A \wedge B$ è supposta vera

Se piove e tira vento.

Piove.

Inferenza

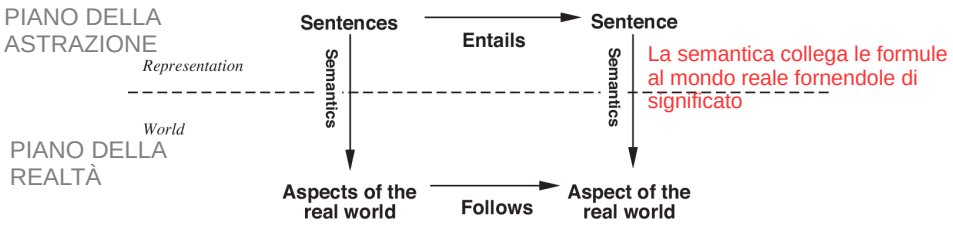
- Sia i un algoritmo di inferenza
- $KB \vdash_i A$
  (A è derivabile da KB utilizzando l'algoritmo i)
  se tale algoritmo permette di produrre una sequenza di passi che, partendo da KB, porta a A
- Si dice anche:
  - *A segue da KB*
  - *A può essere inferito da KB*
  - *Esiste una dimostrazione (una prova) di A a partire da KB*

Proprietà desiderate dell'algoritmo di inferenza

- Correttezza (soundness):
  l'algoritmo deriva *solo* formule che sono anche conseguenze logiche (preserva la verità):
  - Se $KB \vdash_i A$ allora $KB \models A$
- Completezza (completeness):
  l'algoritmo permette di derivare *tutte* le formule che sono anche conseguenze logiche
  - Se $KB \models A$ allora $KB \vdash_i A$

Inferenza e realtà

Le formule sono astrazioni usate per regionare sul mondo



La semantica collega le formule al mondo reale fornendole di significato

I modelli catturano il mondo reale, insieme alla relazione di conseguenza

NOTE: (1) le logiche devono **sempre garantire la correttezza** (le inferenze devono corrispondere a reali conseguenze nel mondo); (2) **non sempre garantiscono la completezza** (cioè di catturare tutte le conseguenze possibili).

Grounding

- Cattura il legame fra la rappresentazione simbolica, formale e l'ambiente reale che essa rappresenta
- Possiamo immaginare il grounding come derivante dalla percezione
- Esempio:
  - quando piove le strade sono bagnate.
  - Vedo che piove e quindi concludo che nel mondo reale le strade sono bagnate

Logica proposizionale

- È uno dei più semplici tipi di logica. Le formule non includono variabili.
- Ne vedremo:
  1) Sintassi
  2) Semantica
  3) Inferenza
  4) Equivalenza, validità, soddisfacibilità

Logica proposizionale: formule

- **Formule atomiche**:
  - **simboli proposizionali**: ognuno rappresenta una formula che può essere vera o falsa
  - Hanno un nome che inizia con la maiuscola
- **Formule complesse**: sono costruite componendo altre formule tramite gli **operatori della logica:**
  - **Negazione**: il termine letterale indica formule atomiche eventualmente negate
  - **Congiunzione**: le formule composte tramite questo operatore sono dette congiunti
  - **Disgiunzione**: le formule composte tramite questo operatore sono dette disgiunti
  - **Implicazione**: correla una formula detta premessa (o antecedente) a una formula detta conclusione (o conseguente)
  - **Biimplicazione** (o equivalenza)

Sintassi della logica proposizionale

- Grammatica:
  - formula → formulaAtomica | formulaComplessa
  - formulaAtomica → True | False | simbolo
  - simbolo → P | Q | R | ...
  - formulaComplessa → ¬ formula
    | (formula ∧ formula)
    | (formula ∨ formula)
    | (formula ⇒ formula)
    | (formula ⇔ formula)

Sintassi della logica proposizionale

- Grammatica:
  - formula → formulaAtomica | formulaComplessa
  - formulaAtomica → True | False | simbolo
  - simbolo → P | Q | R | ...
  - formulaComplessa → ¬ formula NEGAZIONE
    | (formula ∧ formula) CONGIUNZIONE
    | (formula ∨ formula) DISGIUNZIONE
    | (formula ⇒ formula) IMPLICAZIONE
    | (formula ⇔ formula) EQUIVALENZA

Sintassi della logica proposizionale

- Grammatica:
  - formula → formulaAtomica | formulaComplessa
  - formulaAtomica → True | False | simbolo
  - simbolo → P | Q | R | ...
  - formulaComplessa → ¬ formula
    | (formula ∧ formula)
    | (formula ∨ formula)
    | (formula ⇒ formula)
    | (formula ⇔ formula)

Ordine di precedenza degli operatori dal più forte al più debole, **es: ¬ Q ∨ P equivale a ((¬ Q)∨ P)**

Esempio: background knowledge proposizionale

R1) Piove ⇒ Atmosfera_umida

R2) Notte ∧ (¬ Vento) ⇒ Atmosfera_umida

R3) Atmosfera_umida ⇒ (Prato_bagnato ∧ Strada_bagnata)

R4) Innaffiatore_on ⇒ Prato_bagnato

R5) Piove ⇒ Ombrello_aperto

R6) Sole ∧ Vento ⇒ Innaffiatore_on

R7) Sole ∧ Vento ⇒ Atmosfera_asciutta

R8) Sole ⇒ ¬ Notte

R9) Notte ⇒ ¬ Sole

R10) Atmosfera_asciutta ⇒ ¬ Atmosfera_umida