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

Usare la conoscenza per ragionare

Ragionare vuol dire rendere esplicita della conoscenza che prima non lo era

- Se so che:
  - Tutti i cetacei vivono in mare
  - Tutte le balene sono cetacei

Cosa posso concludere che sia ugualmente vero e che ora non è esplicito?

Usare la conoscenza per ragionare

- Se so che:
  - Tutti i cetacei vivono in mare
  - Tutte le balene sono cetacei
- Posso concludere che:
  - Tutte le balene vivono in mare

Devo conoscere tutti i cetacei del mondo ed avere verificato che vivono tutti in mare?

No. Mi basta **assumere** che le due affermazioni iniziali siano vere per trarre la conclusione

Il ragionamento è automatizzabile

- Se so che:
  - Tutti gli A hanno la proprietà P
  - Tutti i B sono degli A
- Posso concludere che:
  - Tutti i B hanno la proprietà P

Il ragionamento lavora sulla **forma delle affermazioni**, come se fossero **schemi**, **assumendone la verità**

Il ragionamento è automatizzabile

- Ho bisogno di:
  - strutturare le affermazioni in modo standard → mi serve un linguaggio per rappresentare la conoscenza
  - Codificare le regole del ragionamento → quando posso dire che una certa affermazione è vera?
  - Implementare un algoritmo che sa usare la conoscenza rappresentata e le regole di ragionamento

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

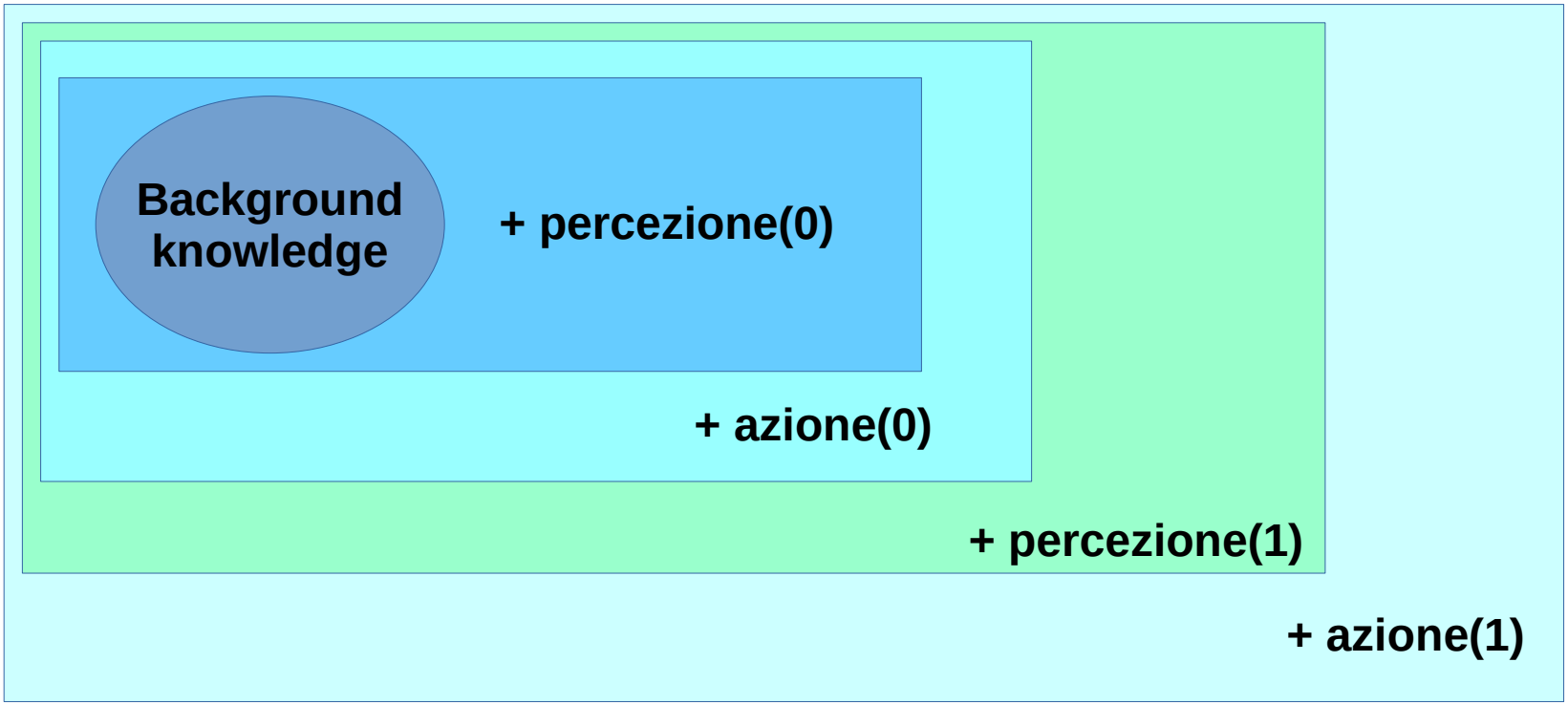
Schema dell'agente, versione 2

```
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
}
```

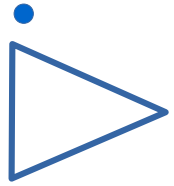
1. Add corrisponde a tell: arricchisce la KB
2. Modify può sia aggiungere che rimuovere elementi dalla KB
3. Esempio: se KB contiene il fatto "il bicchiere è vuoto" l'azione "riempi il bicchiere" cancellerà questo fatto ed aggiungerà "bicchiere pieno"

KB modificata dalla storia



Azioni e successive percezioni modificano la KB

Percezione (dati), informazione e conoscenza



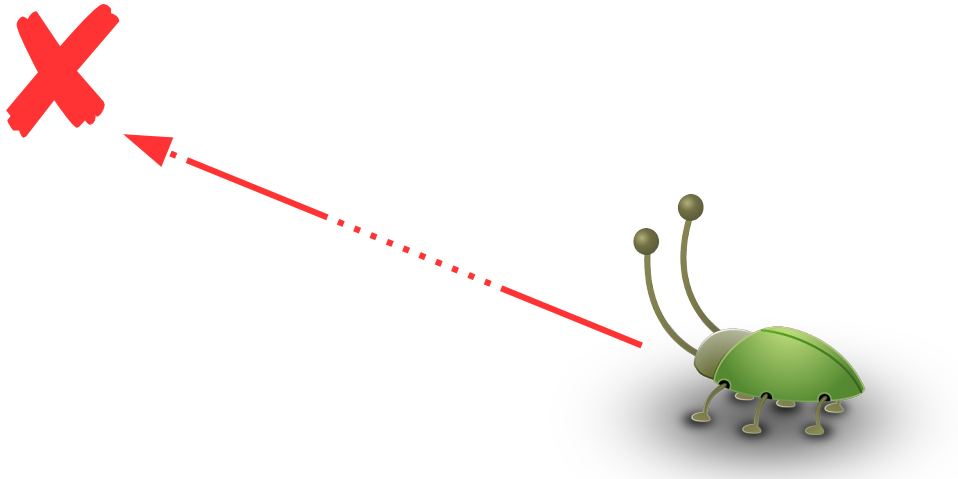
Dato: il disegno che vedete qui a fianco è il risultato della vostra percezione sensoriale. Non ha un significato.

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

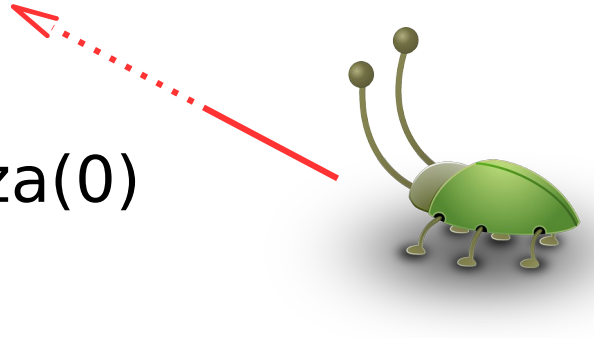
- Semplice agente con conoscenza solo locale dell'ambiente
- In grado di procedere in linea retta e di aggirare un ostacolo applicando un comportamento di wall following (ruota a caso verso destra o sinistra e procede tenendosi parallelo al muro)
- In robotica realizzabile usando solo sensori di contatto
- 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)

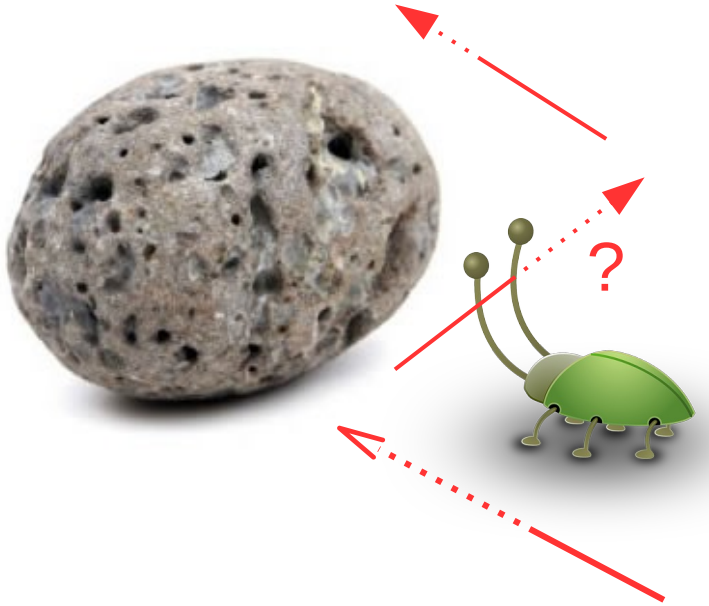
Esempio, bug algorithm

- 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

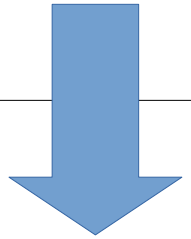
- **NB**: paradigma ben diverso da quello procedurale dove tutto è codificato da procedimenti specifici per il dominio!!

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

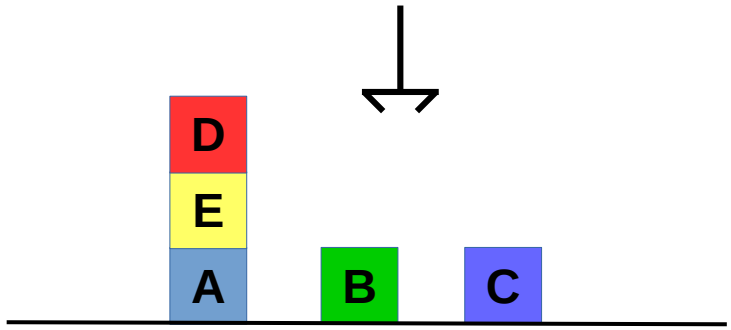
`(10)*111(10)+`

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

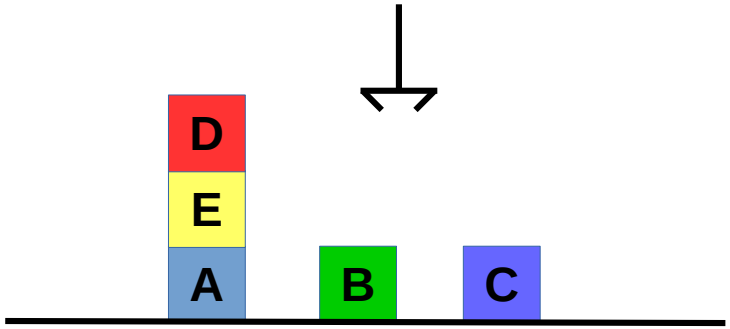
10101011110	**SÌ**
111	**NO**

Mondo dei blocchi

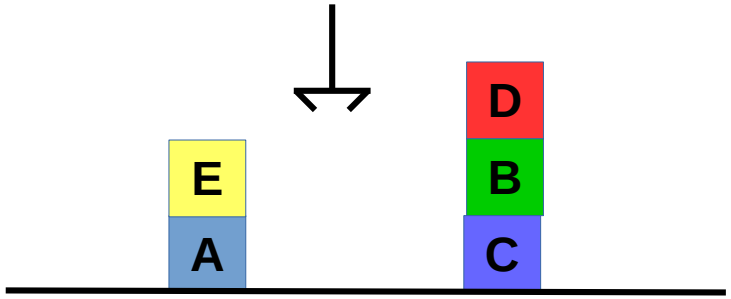
- Su un tavolo sono posizionati blocchi di legno (in generale di diversa forma e dimensione, per noi cubi di identiche dimensioni).
- L'obiettivo è costruire una o più torri posizionandoli uno sull'altro.
- Solo un blocco per volta può essere mosso o posizionandolo o sopra al tavolo o sopra a un altro 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)

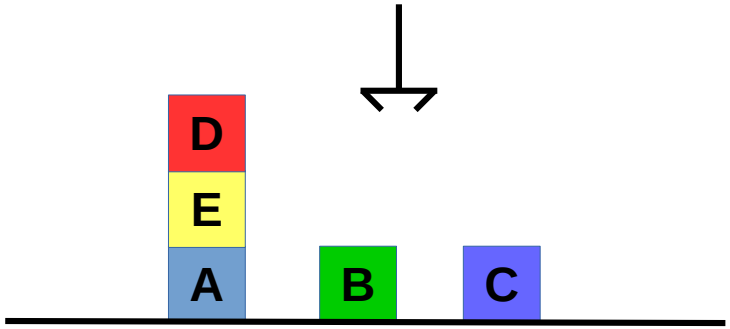


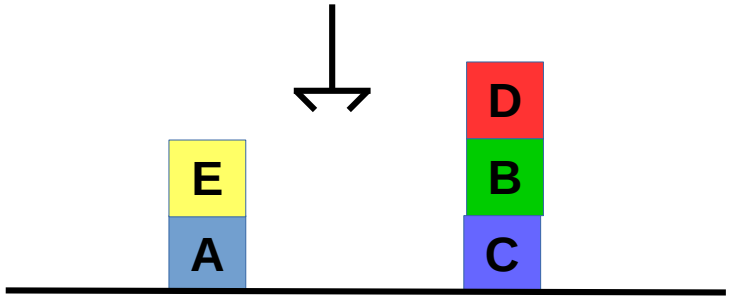
Descrizione della situazione obiettivo:
ontable(A), ontable(C),
handempty, on(B,C), on(D,B),
clear(D), clear(E)

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)

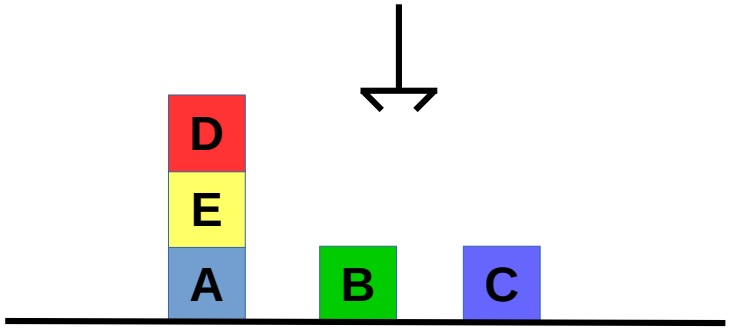
Descrizione della situazione obiettivo:
ontable(A), ontable(C),
handempty, on(B,C), on(D,B),
clear(D), clear(E)

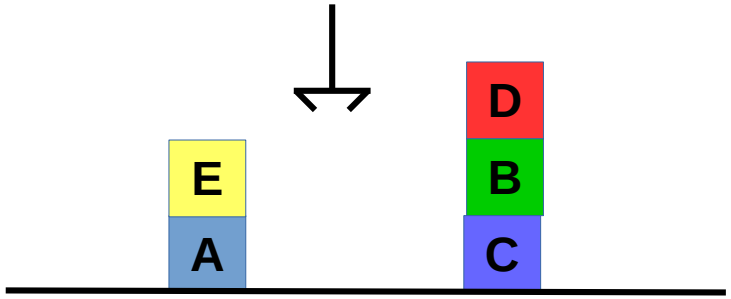
KB: azioni con condizioni di applicabilità ed effetti, esempio

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

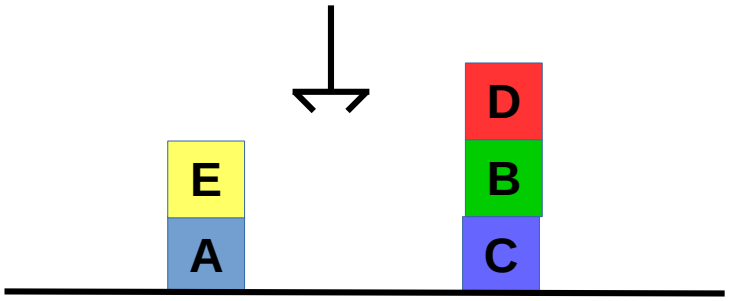
Mondo dei blocchi

- 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



on(E,A) and clear(B): **FALSA**

clear(E) or clear(C): **VERA**

Logica

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