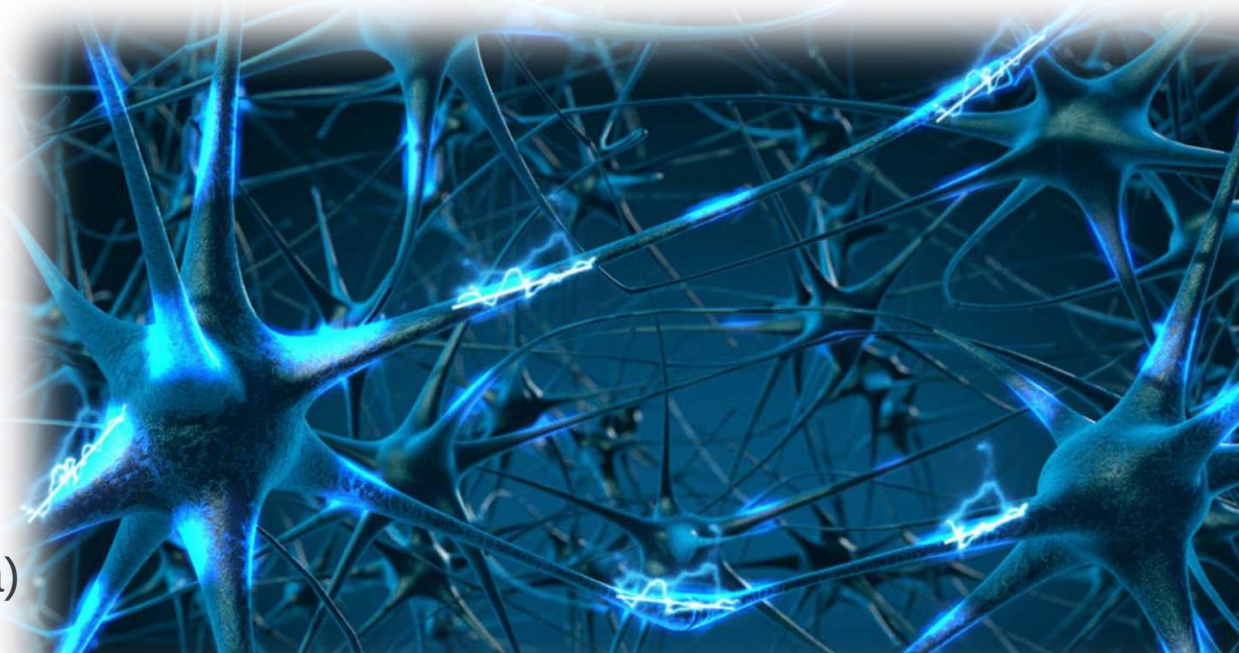
INTELLIGENZA ARTIFICIALE

AGENTI E RAPPRESENTAZIONE DELLA CONOSCENZA (*)

Corsi di Laurea in Informatica, Ing. Gestionale, Ing. Informatica, Ing. di Internet

(a.a. 2021-2022)

Roberto Basili



(*) alcune *slides* sono di
Maria Simi (Univ. Pisa)

Overview

- Il ruolo della conoscenza nella soluzione dei problemi degli agenti
  - Il mondo di Wumpus
  - La intenzionalità
- Formalizzazione della conoscenza in logica
  - Vantaggi
  - Espressività e Complessità
- Aspetti computazionali nell'uso della logica
  - Rappresentazione, Grounding e Ragionamento
  - Gestione della non monotonia
  - Il ruolo della ontologia

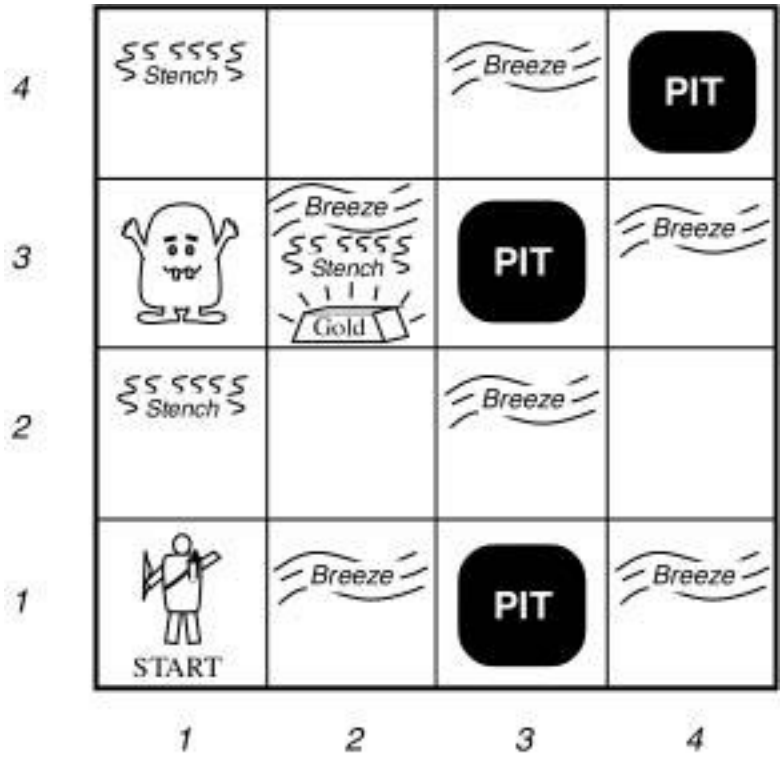
Che cosa abbiamo fatto fin'ora …

- Abbiamo trattato:
  - agenti con stato e con obiettivo, più razionali rispetto ad agenti reattivi
  - stati e azioni descrivibili in maniera semplice
  - enfasi sul processo di ricerca
- Vogliamo adesso migliorare le capacità razionali dei nostri agenti dotandoli di rappresentazioni di mondi più complessi, non descrivibili semplicemente
- Agenti *basati su conoscenza*, con conoscenza espressa in maniera esplicita e dichiarativa (non cablata)

Perché?

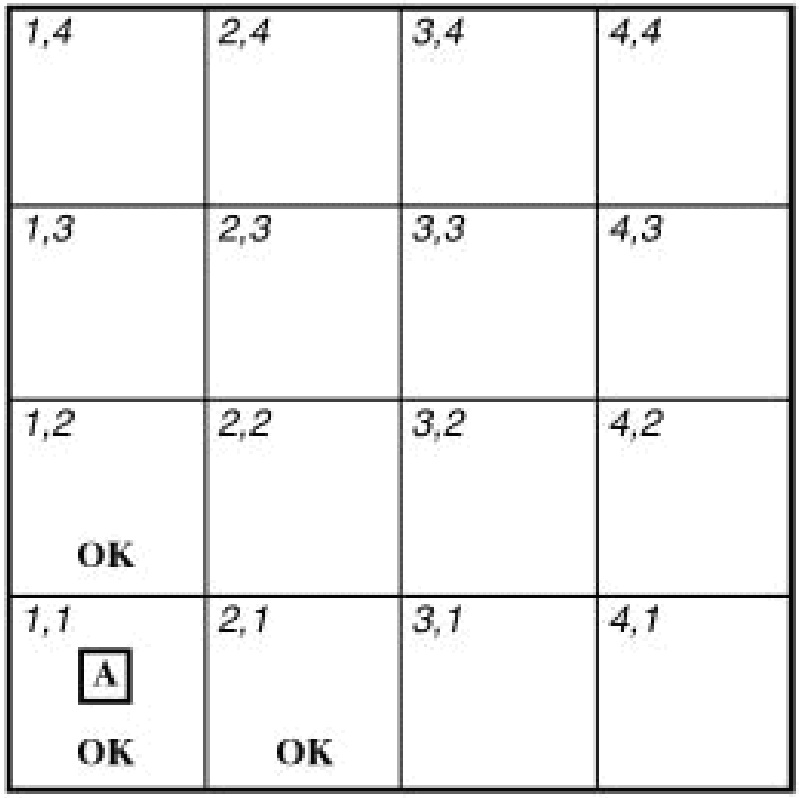
- Il mondo è tipicamente complesso: ci serve una rappresentazione *parziale* e *incompleta* di una astrazione del mondo utile agli scopi dell'agente
- Maggiore conoscenza del mondo è cruciale per risolvere problemi in cui la osservazione è parziale ed il mondo evolve dinamicamente
- Quindi, per ambienti parzialmente osservabili ad esempio, ci servono linguaggi di rappresentazione della conoscenza *più espressivi* e *capacità inferenziali*
- La maggior parte dei problemi di I.A. sono "*knowledge intensive*" tanto che *Sistemi Basati sulla Conoscenza* è quasi sinonimo di sistemi di I.A.

Il mondo del Wumpus: un esempio



- *Misura delle prestazioni*:
  - +1000 se trova l'oro, torna in [1,1] e esce;
  - -1000 se muore;
  - -1 per ogni azione;
  - -10 se usa la freccia.
- Percezioni:
  - *puzzo* nelle caselle adiacenti al Wumpus;
  - *brezza* nelle caselle adiacenti alle buche;
  - *luccichio* nelle caselle con l'oro;
  - *bump* se sbatte in un muro;
  - *urlo* se il Wumpus viene ucciso.
  - L'agente non percepisce la sua locazione.
- *Azioni*:
  - *avanti*
  - *a destra* di 90º, *a sinistra* di 90º
  - *afferra* un oggetto
  - *scaglia la freccia* (solo una)
  - Esce
- Ambienti generati a caso ([1,1] sempre *safe*)

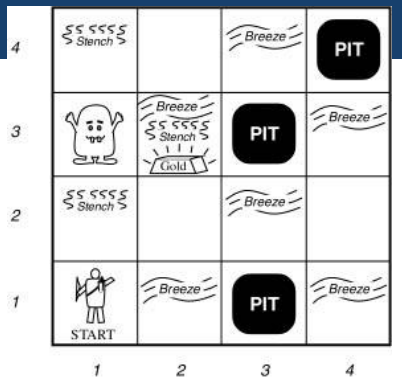
Il mondo del Wumpus: uno scenario



(a)

- Né *Brezza* né *Puzzo* in [1,1], quindi [1,2] e [2,1] sono sicure.
- L'agente decide di spostarsi in [2,1] …

Il mondo del Wumpus: uno scenario

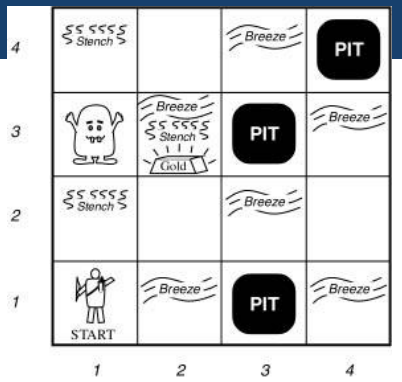


1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

(b)

- L'agente percepisce una brezza. Quindi c'è una buca in [2,2] o [3,1]
- L'agente torna in [1,1] e poi si sposta in [1,2]
- …

Il mondo del Wumpus: uno scenario



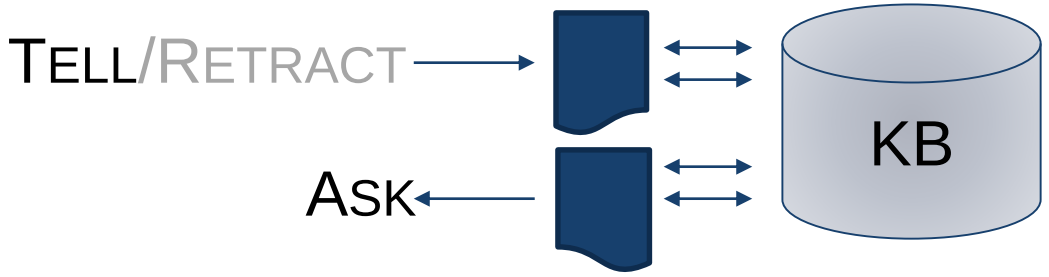
1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(b)

- L'agente si sposta in [2,2] e poi in [2,3]
- Qui percepisce un luccichio, afferra l'oro e torna sui suoi passi, percorrendo caselle OK

Agente basato su conoscenza

- Un agente basato su conoscenza mantiene una *base di conoscenza* (KB): un insieme di *enunciati* espressi in un linguaggio di rappresentazione
- Interagisce con la KB mediante una interfaccia funzionale *Tell-Ask*:
  - *Tell*: per aggiungere nuovi fatti a KB
  - *Ask*: per interrogare la KB
  - … forse *Retract*



- Le risposte α devono essere tali che α è una conseguenza della KB (è *conseguenza logica* di KB)

Il problema da risolvere

- *Il problema*: data una base di conoscenza KB, contenente una rappresentazione dei fatti che si ritengono veri, vorrei sapere se un certo fatto α è vero di conseguenza

$$KB \models \alpha$$ (*conseguenza logica*)

Programma di un agente B.C.

Function Agente-KB (*percezione*) **returns** *un'azione*

persistent: *KB*, una base di conoscenza

t, un contatore, inizialmente a 0, che indica il tempo

TELL(*KB*, Costruisci_Formula_Percezione(*percezione*, *t*))

azione ← ASK(*KB*, Costruisci_Query_Azione(*t*))

TELL(*KB*, Costruisci_Formula_Azione(*azione*, *t*))

t ← *t* + 1

return *azione*

Agente basato su conoscenza

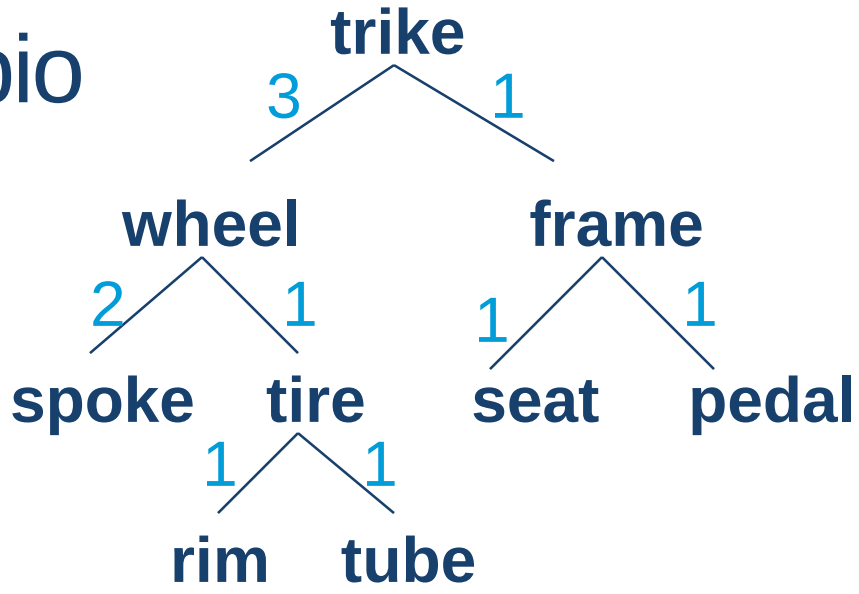
- Approccio dichiarativo vs approccio procedurale
- La differenza principale è che la KB racchiude tutta la conoscenza necessaria a decidere l'azione da compiere in forma *dichiarativa*
- L'alternativa (*approccio procedurale*) è scrivere un programma in cui il processo decisionale è cablato, una volta per tutte.
- Più flessibile: più semplice acquisire conoscenza incrementalmente e modificare il comportamento con l'esperienza

Che relazione c’è con le BdD?

Cosa una Base di Dati non può fare

- SQL-92 non può esprimere alcune interrogazioni utili:
  - Siamo a corto di alcune parti che compongono una Ferrari ZX600?
  - Qual'e' il costo attuale (totale) delle component e dell'assemblaggio di una ZX600?
- Si può estendere Il linguaggio SQL per poter esprimere tali interrogazioni? Come?
  - Si, introducendo la ricorsione.

Esempio



part	subpart	number
trike	wheel	3
trike	frame	1
frame	seat	1
frame	pedal	1
wheel	spoke	2
wheel	tire	1
tire	rim	1
tire	tube	1

istanza di Assembly

- Trovare le component di un triciclo?
- Le interrogazioni in Algebra Relazionale ci consentono di calcolare la risposta su una **certa istanza di Assembly.**
- ma … non esiste alcuna interrogazione in AR (o SQL-92) che calcoli le risposte **in ogni istanza di *Assembly***.

Limiti dell'algebra relazionale (SQL-92)

- Intuitivamente, dobbiamo computare una *join* di Assembly con se stessa per derivare che un triciclo contiene un *spoke* ed un *tire*.
  - Ma questo ci porta più in basso di un livello nella gerarchia di Assembly.
  - Per trovare le componenti che sono più in basso di un secondo livello (e.g., rim) abbiamo bisogno di un'altra *join*.
  - Per tutte le componenti, abbiamo bisogno di tante *join* quanti sono i livelli nella gerarchia espresso dalla istanza corrente!

- (Per ogni interrogazione relazionale, è facile trovare una istanza che lasci fuori da tale interrogazione alcune componenti: basta inserirle ad un livello più basso del numero di *join* nella espressione)

Le forme della Rappresentazione in Logica

```
printColor(snow) :- !, write("It's white.").
printColor(grass) :- !, write("It's green.").
printColor(sky) :- !, write("It's yellow.").
printColor(X) :- write("Beats me.").
```

ecco una alternativa

```
printColor(X) :- color(X,Y), !,
        write("It's "), write(Y), write(".").
printColor(X) :- write("Beats me.").
color(snow,white).
color(sky,yellow).
color(X,Y) :- madeof(X,Z), color(Z,Y).
madeof(grass,vegetation).
color(vegetation,green).
```

Le forme della Rappresentazione in Logica

```
printColor(X) :- color(X,Y), !,
       write("It's "), write(Y), write(".").
printColor(X) :- write("Beats me.").
color(snow,white).
color(sky,yellow).
color(X,Y) :- madeof(X,Z), color(Z,Y).
madeof(grass,vegetation).
color(vegetation,green).
```

- Perché il secondo dei programmi Prolog è una rappresentazione **più adeguata** della procedura rispetto alla prima? Perché è:
  - Trasparente epistemologicamente
  - Fornisce una distinzine chiara tra *ciò che si conosce* e *ciò che deve essere eseguito*
  - Non dipende da ciò che il sistema crede (o sa) riguardo al printing ma piuttosto da *ciò che il sistema conosce riguardo sulla neve ed i colori (cioè sull'ambiente)*

Le forme della Rappresentazione in Logica

- Solo il secondo dei programma ha una rappresentazione esplicita della conoscenza riguardo al fatto che “la neve è bianca"
- Il secondo programma procede con la sua stampa del colore della neve proprio a causa di questa conoscenza. Se **`colour(snow,white)`** fosse rimosso dale sue conoscenze, non stamperebbe più il colore corretto per la neve.
- Ciò che rende il programma basato sulla conoscenza NON E’:
  - L’uso di un linguaggio basato sulla logica come il Prolog
  - La proprietà che i fatti in esso rappresentati siano very (notiamo che **`colour(sky, yellow)`** è falso)
  - La numerosità dei fatti memorizzati o la complessità strutturale
- piuttosto, *è la proprietà di possedere una esplicita rappresentazione delle conoscenze che sono usate nel processo di calcolo attuato dal programma.*

Intentional stance

Imagine, for example, playing a game of chess against a complex chess-playing program. In looking at one of its moves, we might say to ourselves something like this: "It moved this way because it believed its queen was vulnerable, but still wanted to attack the rook." In terms of how the chess-playing program is actually constructed, we might have said something more like, "It moved this way because evaluation procedure *P* using static evaluation function *Q* returned a value of +7 after an alpha-beta minimax search to depth 4." The problem is that this second description, although perhaps quite accurate, is at the wrong level of detail, and does not help us determine what chess move we should make in response. Much more useful is to understand the behavior of the program in terms of the immediate goals being pursued relative to its beliefs, long-term intentions, and so on. This is what the philosopher Daniel Dennett calls taking an *intentional stance* toward the chess-playing system.

- Dennet, 1991:
- http://human-nature.pbworks.com/f/Dennett,+Intentional+Systems.pdf
- https://ase.tufts.edu/cogstud/dennett/papers/intentionalsystems.pdf

I vantaggi della Rappresentazione in Logica

- Il secondo programma Prolog E' PIU' ADEGUATO come rappresentazione della conoscenza riguardo alla sottostante procedura
- **VANTAGGI**
  - **Modularità:** è utile al RIUSO della connoscenza per altri task
  - **Messa a punto e Raffinamento di strategie** nella stampa dei colori (e.g. gestione di nuove credenza)
  - **Manutenzione semplificata** del software (se qualcosa va starto, e.g. «l'erba non è verde» , è minimo l'insieme dei fatti da modificare)
  - **Trasparenza epistemologica:** Il sistema sembra essere in grado di **spiegare** ciò che fa

Agenti basati su conoscenza

- I sistemi basati su conoscenza (o *Knowledge-based*) sono sistemi per i quali le posizioni intenzionali sono giustificate/espresse/radicate nella rappresentazione in simboli (sin dal loro *design*)
- La espressione simbolica della conoscenza è detta Base di Conosecnza.

- La logica consente una implementazione possibile:
  - **Rappresentazione**: come l'insieme delle frasi in PROP o first order logic (FOL)
  - **Ragionamento**: la capacità di dedurre le conseguenze logiche dei fatti/assiomi

Knowledge-based Systems

This is what the philosopher Brian Smith calls the *Knowledge Representation Hypothesis:*

> Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantic attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.

In other words, the Knowledge Representation Hypothesis implies that we will want to construct systems for which the intentional stance is grounded by design in symbolic representations. We will call such systems *knowledge-based systems* and the symbolic representations involved their *knowledge bases* (KBs).

Base di conoscenza …

- *Base di conoscenza*: una rappresentazione esplicita, parziale e compatta, in un linguaggio simbolico, che contiene:
  - fatti di tipo specifico (Es. *Socrate è un uomo*)
  - fatti di tipo generale, o regole (Es. *Tutti gli uomini sono mortali*)

- Quello che caratterizza una B.C. è la *capacità di inferire* nuovi fatti da quelli memorizzati esplicitamente (Es. *Socrate è mortale*)

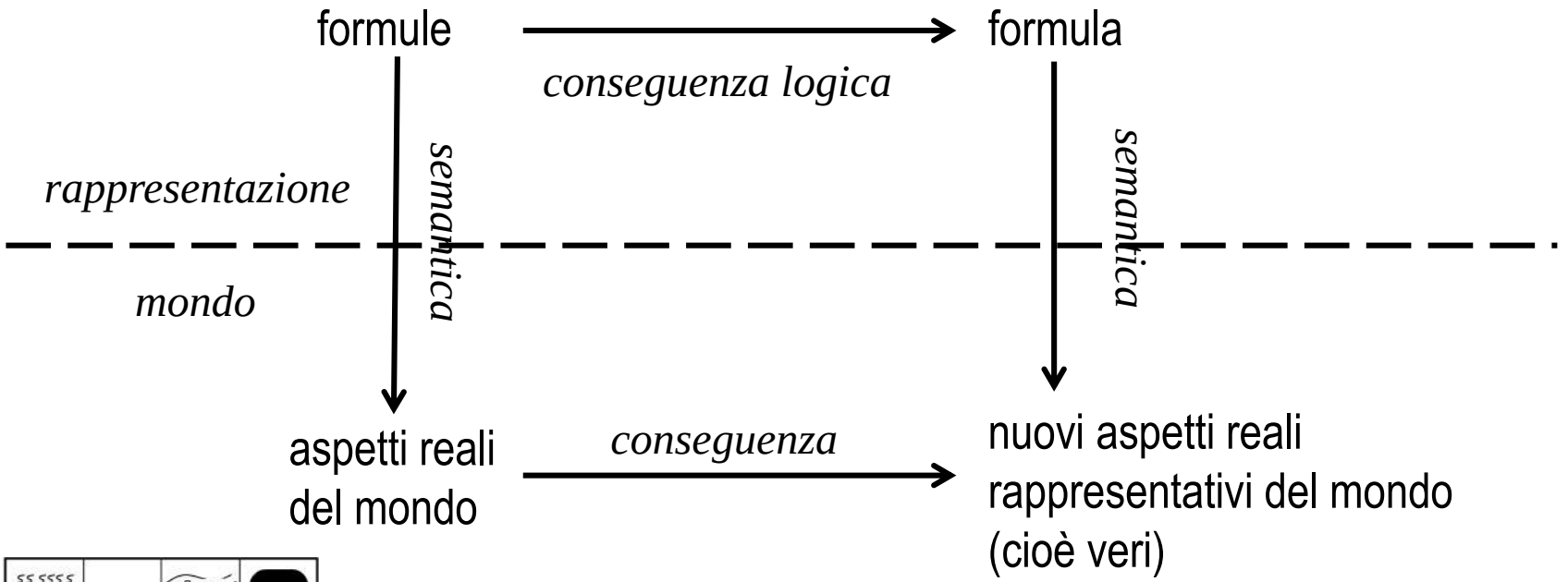
… e base di dati

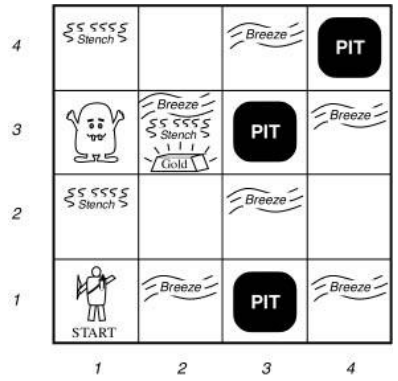
… e base di dati

- Nelle basi di dati solo fatti specifici e positivi
  - (‘Rossi’,’Paolo’, CFN00MM, ‘Via della Ricerca’, Roma)
  - (‘Verdi’,’Mario’, CFN19lO, ‘Via di TV’, Roma)
- Le basi di dati assumono una conoscenza completa del mondo (Closed World Assumption)
  - "Verdi Paolo" non esiste (!)
- Nessuna capacità inferenziale:
  - I vincoli di integrità gestiscono la *consistenza* dei fatti inseriti
  - Non è prevista la generazione di nuovi fatti

Rappresentazione e mondo:

… generare nuovi fatti





Il *trade-off* fondamentale della R.C.

- Il problema ‘fondamentale’ in R.C. è trovare il giusto compromesso tra:
  - *Espressività* del linguaggio di rappresentazione;
  - *Complessità* del meccanismo inferenziale
- Vogliamo linguaggi *espressivi,* ma anche *efficienti*.
- Questi due obiettivi sono in contrasto e si tratta di mediare tra queste due esigenze

Espressività come *imprecisione*

- Cosa vuol dire *espressivo*? … e perchè l'espressività è in contrasto con l'efficienza?
- Un linguaggio più espressivo ci consente di essere vaghi, imprecisi, di esprimere conoscenze parziali, di omettere dettagli che non si conoscono …
- L'espressività determina non tanto quello che può essere detto ma quello che può essere *lasciato non detto*

Espressività come *imprecisione* : esempi

Nelle Basi di Dati quello che possiamo esprimere sono solo fatti specifici e positivi:

1. M*oglie*(*Rossi*, *Paola*)

Con linguaggi più espressivi …

2. $\exists x$ *Moglie*(*Rossi*, x) *Rossi ha una moglie*
3. $\neg$*Operaio*(*Rossi*) *Rossi non è un operaio*
4. *Moglie*(*Rossi*, *Anna*) $\vee$ *Moglie*(*Rossi*, *Paola*)

   Rossi ha una moglie; si chiama Anna o Paola
5. $\forall y \exists x$ *Moglie*(y, x) $\Rightarrow$ *Coniugato*(y)

   Coloro che hanno una moglie sono coniugati

Espressività e complessità inferenziale

- Nelle basi di dati nessuna deduzione è possibile, solo *recupero*. Si assume una descrizione *completa* del mondo.
- Invece: dai fatti
  - $\exists$ *x Moglie*(*Rossi*, x)
  - $\forall$y $\exists$ x *Moglie*(y, x) $\Rightarrow$ *Coniugato*(y)

  è possibile dedurre: *Coniugato*(*Rossi*)
- Come pure dai fatti
  - M*oglie*(*Rossi*, *Anna*) $\vee$ *Moglie*(*Rossi*, *Paola*)
  - $\forall$y $\exists$ x *Moglie*(y, x) $\Rightarrow$ *Coniugato*(y)

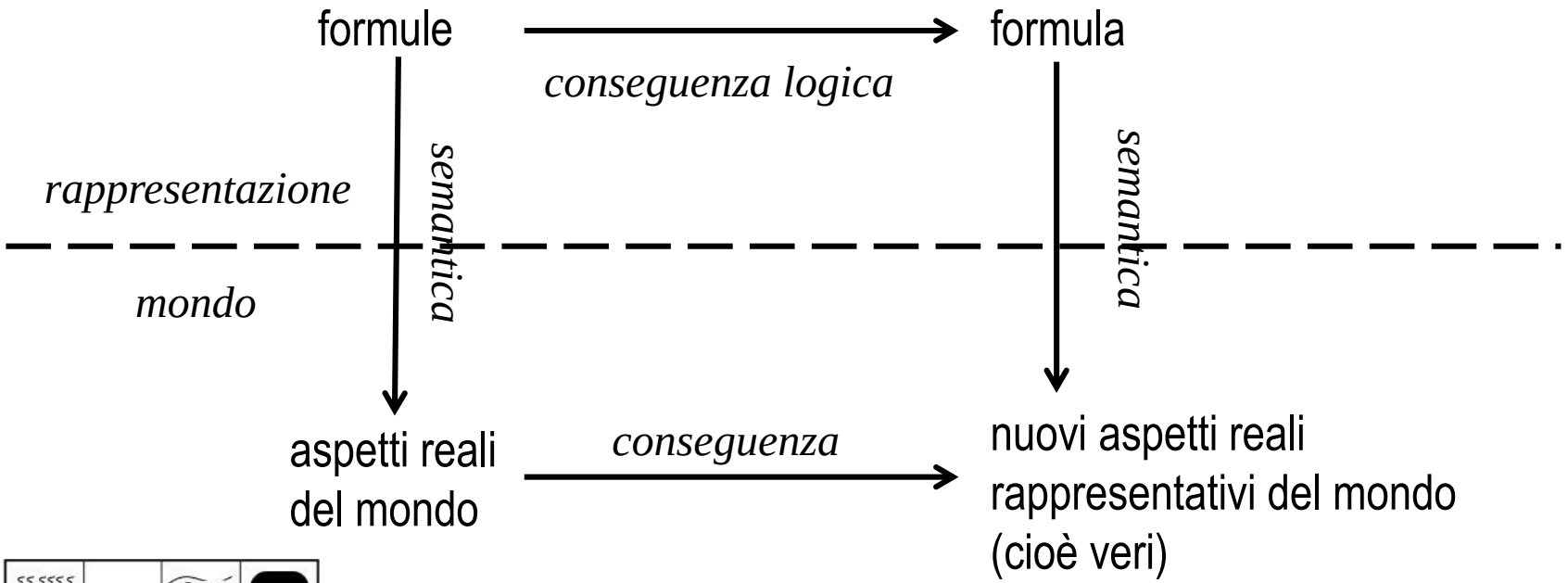
  è possibile dedurre *Coniugato*(*Rossi*) ma è più complicato (richiede un ragionamento per casi)

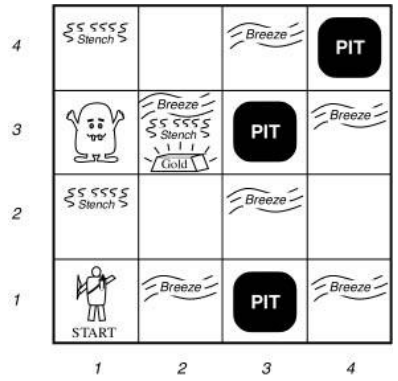
Formalismi per la R.C.

Un formalismo per la rappresentazione della conoscenza ha tre componenti:

1. Una ***sintassi***: un linguaggio composto da un vocabolario e regole per la formazione delle frasi (*enunciati*)
2. Una ***semantica***: che stabilisce una corrispondenza tra gli enunciati e fatti del mondo; se un agente ha un enunciato α nella sua KB, crede che il fatto corrispondente sia vero nel mondo
3. Un ***meccanismo inferenziale*** (codificato o meno tramite regole di inferenza come nella logica) che ci consente di inferire nuovi fatti (conseguenza logica).

Rappresentazione e mondo



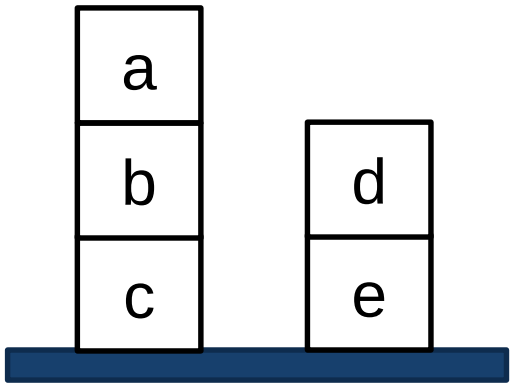


Grounding (radicamento)

- Come sappiamo che la KB è vera nel mondo reale? Come l'agente forma le sue credenze?
- Attraverso i sensori si crea una connessione con il mondo; le credenze sono il risultato di percezioni.
- Non solo: le regole sono il risultato di un processo di apprendimento, che può essere fallibile (es. ragionamento induttivo).

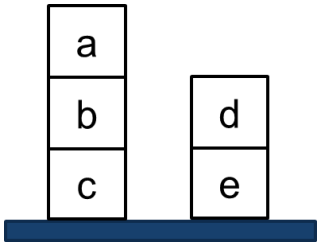
Un esempio motivante

- Ci interessano i blocchi e alcune loro relazioni spaziali
- Dominio: *{a, b, c, d, e}*
- Le funzioni: si individuano le funzioni rilevanti che servono anch'esse per identificare oggetti.
  - Es. *Hat* la funzione unaria che dato un blocco identifica il blocco che ci sta sopra;
    - *Hat(b)=a*
- Le relazioni: si individuano le relazioni interessanti. Ad es.:
  - *On/2= {<a, b>, <b, c>, <d, e>}*
  - *Clear/1= {a, d}*
  - *Table/1= {c, e}*
  - *Block/1= {a, b, c, d, e}*



Grounding (esempio)

- Attraverso i sensori si crea una connessione con il mondo



- Le credenze sono il risultato di percezioni
  - Accettiamo fatti come *Clear(a), On(d,e), Table(c)*
  - … ma non altri fatti come *Clear(c), On(c,a), Table(b)*

- Non solo: le regole sono il risultato di un processo di apprendimento, per esempio il ricordo di precedenti osservazioni (vedi LRTA*)
- L'apprendimento però può essere fallibile (es. ragionamento induttivo): ad es. generalizzazioni sbagliate o rumore nelle osservazioni

Logica come linguaggio per la R.C.

- I linguaggi logici, calcolo proposizionale (PROP)
- e logica dei predicati (FOL), sono adatti per la rappresentazione della conoscenza?
  - **PROP**: $A, B, \neg B, (A \wedge \neg B\ ) \Rightarrow C \ldots$
  - **FOL**: $\forall y \exists x\ Moglie(y, x) \Rightarrow Coniugato(y)$
- Qual è la complessità computazionale del problema KB $\models \alpha$?
  - In PROP il problema è decidibile, ma intrattabile (NP)
  - FOL è un linguaggio espressivo, con una semantica ben definita, ma ha un problema: il meccanismo inferenziale non è decidibile

    In FOL il problema KB $\models \alpha$ è semidecidibile

Linguaggi per la R.C.: efficienza

1. Superamento del FOL verso linguaggi ad inferenza limitata: *contrazioni* del FOL alla ricerca di proprietà computazionali migliori (es. i linguaggi di *programmazione logica*, le logiche *descrittive*)
2. Linguaggi di rappresentazione che propongono meccanismi di strutturazione della conoscenza per guadagnare efficienza su forme particolari di inferenza (es. *reti semantiche* e connettività, *frame* e aggregazione, *ereditarietà*). FOL per la semantica.

Limiti in espressività del FOL

Molti linguaggi della R.C. sono *estensioni* [di sottoinsiemi] del FOL per superare limiti di espressività nel ragionamento di “senso comune”

Ne possiamo citare tre importanti:

- Atteggiamenti proposizionali
- Ragionamento incerto
- Ragionamento non monotòno

Atteggiamenti proposizionali

- Atteggiamenti epistemici
  - conoscenze, credenze (convinzioni o opinioni)
- Atteggiamenti motivazionali
  - desideri, obiettivi, intenzioni, …
- L'oggetto del discorso sono le proposizioni

  Bel(P) operatori e logiche modali

  Bel(‘P’) reificazione o meta-livello

Ragionamento incerto

- Nella logica classica le proposizioni sono vere (*T*) o false (*F*) (*assunzione epistemologica*)
- Alcune forme della conoscenza non si conformano bene a tale rigidità
  - *" Sicuramente pioverà ", "Aldo è alto"*
  - *" A basse temperature corrispondono pressioni più alte '*

- Il superamento della dicotomia *T*|*F* può avvenire in modi diversi:
  - logiche a più valori (*vero, falso, non so*)
  - ragionamento probabilistico (*a* vera con probabilità *P, P(a)*)
  - vero con *grado di fiducia c*
  - logica *fuzzy* (proprietà sfumate, es. '*alto*' come aspettativa nell'intervallo [1m; 2,30 metri])

Ragionamento non monotòno

Nella logica classica vale la proprietà di *monotonia*:

Monotonia: Se KB $\models \alpha$ allora KB $\cup \{\beta\} \models \alpha$

- Il ragionamento di senso comune è spesso *non monotòno:* si fanno inferenze tentative, anche in mancanza di informazioni complete.

- Esempio 1: ragionamento *default*
  - *Gli uccelli tipicamente volano. Tweety è un uccello. Quindi Tweety vola.*

- Esempio 2: assunzione di mondo chiuso
  - Se un fatto non è presente nella KB si assume che non sia vero (come nelle basi di dati). Quando si aggiunge un nuovo fatto può invalidare le vecchie conclusioni.

https://staff.fnwi.uva.nl/b.bredeweg/pdf/BSc/20032004/KamermansSchmits.pdf

Assunzioni ontologiche

- Ogni linguaggio per la R. C. fa assunzioni diverse su come è fatto il mondo (ontologico ≅ che riguarda ciò che esiste):
- Nel calcolo proposizionale il mondo è visto come popolato di *fatti* veri o falsi (le proposizioni).
- Il calcolo dei predicati fa una assunzione *ontologica* più sofisticata: il mondo è fatto di *oggetti*, che hanno *proprietà* e tra cui sussistono *relazioni*.
- Logiche specializzate assumono *ontologie* più ricche:
  - gli stati e le azioni nel *calcolo di situazioni*
  - il tempo nelle *logiche temporali*
  - concetti o categorie nelle *logiche descrittive*

Ontologia

- *Parte della filosofia che ha come oggetto di studio i caratteri universali dell'essere in quanto tale, a prescindere dalle sue qualità particolari o fenomeniche (Hoepli)*

- *Parte della filosofia che studia il concetto e la struttura dell'essere in generale, e non i fenomeni in cui si concretizza e specifica* | *nella filosofia analitica, riflessione sui problemi di esistenza a partire dal linguaggio; anche, l'insieme di entità che una teoria assume come esistenti (Garzanti)*

- Informatica: ***Rappresentazione formale**, schematizzata, di un campo di interesse; contiene l'insieme dei concetti (entità, attributi ecc.) e delle possibili relazioni tra concetti (Garzanti)*

Rappresentazione formale

- Ogni programma logico

```
printColor(X) :- color(X,Y), !,
      write("It's "), write(Y), write(".").
printColor(X) :- write("Beats me.").
color(snow,white).
color(sky,yellow).
color(X,Y) :- madeof(X,Z), color(Z,Y).
madeof(grass,vegetation).
color(vegetation,green).
```

- Definisce formalmente
  - Ciò che esiste
    - Q. Esiste qualcosa come "snow"? O "grass«? YES
    - Q. e il "wumpus"? NO
  - Ciò che caratterizza ciò che esiste
    - Q. Qual'e' il colore di «sky»? "yellow"
  - Le relazioni tra le cose che esistono
    - Q. Di cosa è fatto il "grass"? "vegetation"
    - Q. Che relazione c'è tra vegetation e grass? MADEOF

Che cosa vedremo

- Rivisitazione di PROP e FOL per la rappresentazione della conoscenza
- … con attenzione agli algoritmi e alla complessità
- Contrazioni ed estensioni
  - Le regole
  - Le logiche descrittive
  - Linguaggi specializzati per la pianificazione

SummarAIzing

- Gli agenti hanno la necessità di rappresentare proprietà complesse dell'ambiente in cui agiscono, credenze riguardo alla sua evoluzione ed agli altri agenti
- La progettazione di agenti in grado di pianificare comportamenti complessi passa per la capacità di dedurre le conseguenze delle proprie azioni come il mondo della esplorazione in Wumpus dimostra
- La deduzione implicita in questi processi suggerisce che il ruolo della **logica** sia fondamentale sia come strumento di **rappresentazione** dei fatti utili per descrivere la relatà che come paradigm computazionale per modellare il **ragionamento** (relazione di **conseguenza logica**, $\models$)
- C'e' bisogno di molta più espressività di quella disponibile dai linguaggi dell'AR (SQL-92), poichè è richiesta la chiusura transitiva delle relazioni in gioco (ad es. PARTOF)
- Con la rapresentazione logica abilitiamo **basi di conoscenza** come strumenti in grado di organizzare la conoscenza degli agenti e di fondare gli strumenti del ragionamento, in modo da progettare **sistemi basati su conoscenza** (**knowledge-based system**).
- Pur non essendo l'unica via alla rappresentazione della conoscenza in Intelligenza Artificiale, i paradigmi logici consentono di progettare sistemi che esibiscono **posizioni intenzionali** (**intentional stance**): ciò è in accordo con la nozione di intelligenza fornita da diverse teorie filosofiche (Dennet)

SummarAIzing (2)

- La rappresentazione consente di mantenere il **grounding** nell'ambiente che quindi **garantisce il significato dei simboli in ogni istante**. Nel caso di rappresentazione di credenze esse mantengono i principi generali dell'ambiente e può essere dotato di istanze non realizzate
- Le rappresentazioni logiche si sono dimostrate interessanti per la comprensione dei fenomeni di incrtezza. Si distinguono approcci quantitativi al ragionamento incerto come quelli delle **logiche a più valori**, i **modelli probabilistici** o **sfumati** (**fuzzy)**
- La rappresentazion logica ha consentito di evidenziare elementi di **non monotonia** nel ragionamento poichè l'insieme di conseguenze logiche di una teoria può non persistere (e quindi diminuire o cambiare) nel tempo
- Questo ha dato luogo ad una linea di ricerca alla solzuione del *frame problem*
- Soluzioni al problema della non monotonia sono tecniche che estendono la logic $\models$ classica come gli **atteggiamenti proposizionali** e le **logiche di default**.
- Ogni rappresentazione logica dipende da una serie di assunzioni sull'**esistenza degli oggetti rappresentati**, codificate in una **Ontologia**. Questa determina **ciò che si può rappresentare e le sue proprietà e relazioni con il resto del mondo.**
- Il calcolo dei predicati assume mondi più complessi del calcolo proposizionali: entià, properietà complesse e relazioni.

Esercizio

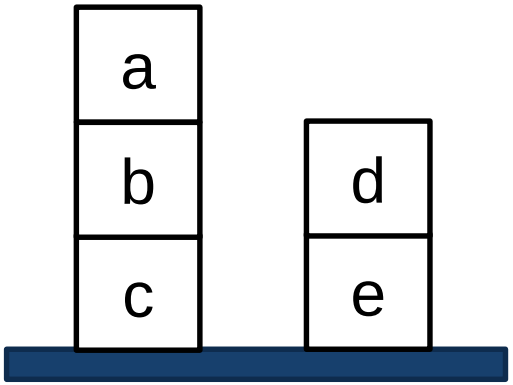
- Scrivere una versione in Prolog della mappa della Romania
  - Grafo:
    - Nodi come nomi delle citta' (in minuscolo costanti Prolog) e
    - Archi come relazioni binarie (predicato **`PArco/2`**) tra nodi
  - Ambiente: come rappresentazione dello stato (ad esempio un fatto dedicato a descrivere dov'è l'agente nella mappa, qual è il costo del suo cammino corrente, …)
  - Azioni come transizioni usando i predicati **`PArco/2`**
  - Piano come sequenza di azioni diventa una lista (ordinata) di predicati **`PArco/2`**)
  - Algoritmo di Search è una query Prolog ….

Esercizio 2

- Descrivere in modo relazionale tramite il Prolog i seguenti giochi
  - Il gioco del 15
  - Il Vacuum Cleaner
  - Le otto regine
- Scrivere la formula Prolog che calcola una delle euristche vista per ciascun gioco di sopra
- Provare a descrivere in Prolog l'esempio di slide 35 rispondendo a domande del tipo
  - Quali blocchi posso prendere?
  - Su quale blocco è posizionato il blocco "A"?

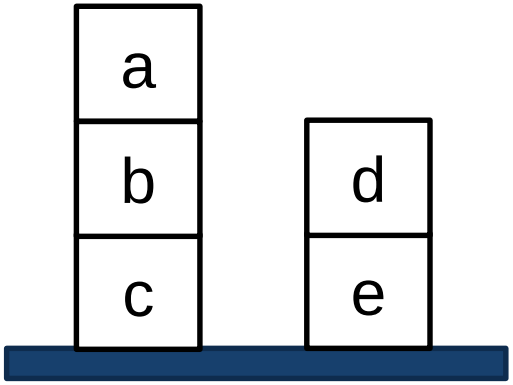
Un esempio motivante in Prolog

- Simboli del dominio, le costanti **`a, b, c, d`** ed **`e`**
- Le relazioni:
  - *On/2 in Prolog i fatti*
  - *Da cui*
    - **`on(a,b).`**
    - **`on(b,c).`**
    - **`on(d,e).`**
  - *Clear/1= {a, d}*
    - **`clear(a).`**
    - **`clear(d).`**
  - *Table/1= {c, e}*
    - **`table(c).`**
    - **`table(e).`**
  - *IsaBlock/1= {a, b, c, d, e}*
    - **`isaBlock(a). isaBlock(b). ... isaBlock(e).`**



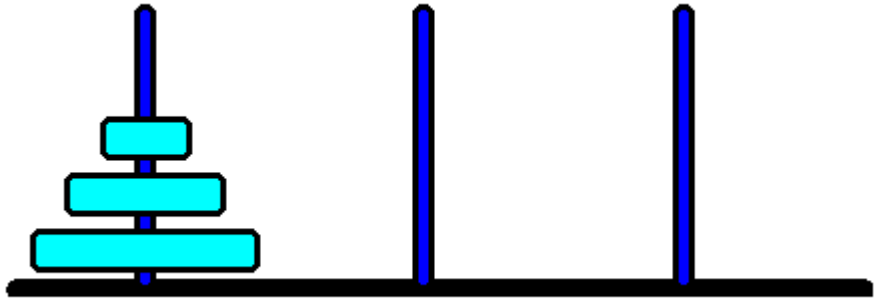
Un esempio motivante in Prolog

- Esempio di slide 35 - domande
  - Quali blocchi posso prendere?
    - Predicato **`takeable/1.`**
    - **`takeable(X) :- clear(X).`**
    - **`?-takeable(X).`**
    - **`Yes X=a ; X=d`**



  - Su quale blocco è posizionato il blocco «A»?
    - **`?-on(a, X).`**
    - **`Yes X=b`**
  - Quali blocchi posso muovere? E come?
    - Predicato **`moveable/2.`**
    - **`movable(Origin, End) :-`**
      `takeable(Origin),`
      `takeable(End).`

Hanoi in Prolog?



```
move(1, X, _, Z, [[X, Z]]).
move(N, X, Y, Z, P) :-  /* To move N disks from X to Z          */
    M is (N - 1),       /* we first must move the smaller N-1 disks */
    move(M, X, Z, Y, P1), /* from X to Y  */
    move(1, X, Y, Z, P2), /* then move the largest disk from X to Z */
    move(M, Y, X, Z, P3), /* then move the smaller N-1 disks form Y to Z */

    append(P1, P2, Q),    /* we get the corresponding solution   */
    append(Q, P3, P).     /* by appending all the steps.   */

hanoi(N, X, Y, Z, Plan) :-
        move(N, X, Y, Z, Plan).

?-hanoi(3, X, Y, Z, Plan).
Yes, Plan = [[X, Z], [X, Y], [Z, Y], [X, Z], [Y, X], [Y, Z], [X, Z]] .
```