UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Intelligenza Artificiale

Agenti Intelligenti

a.a. 2022/2023

Outline

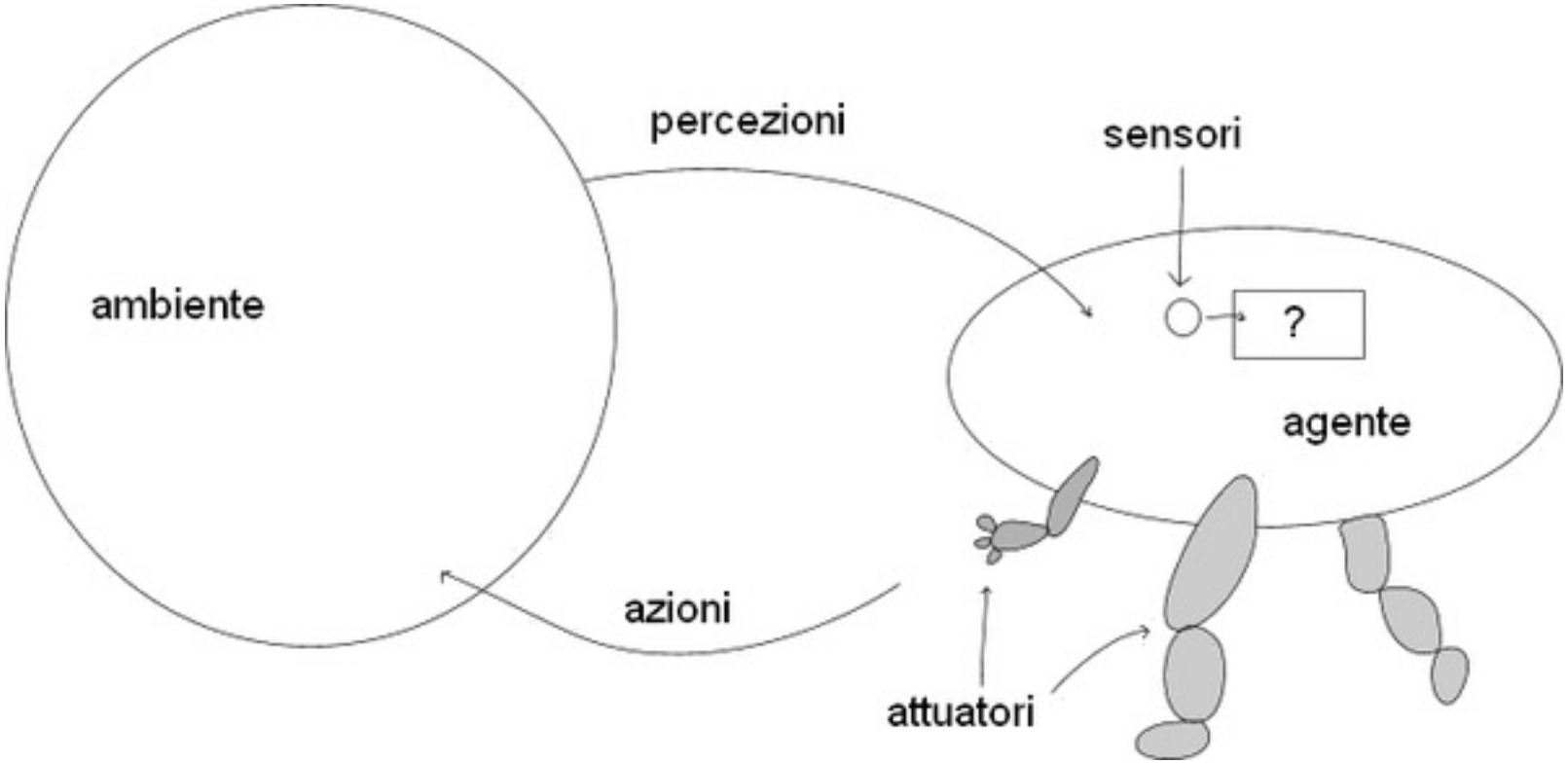
- Agenti ed ambienti
- Razionalità
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Tipi di ambienti
- Tipi di agenti

Agenti

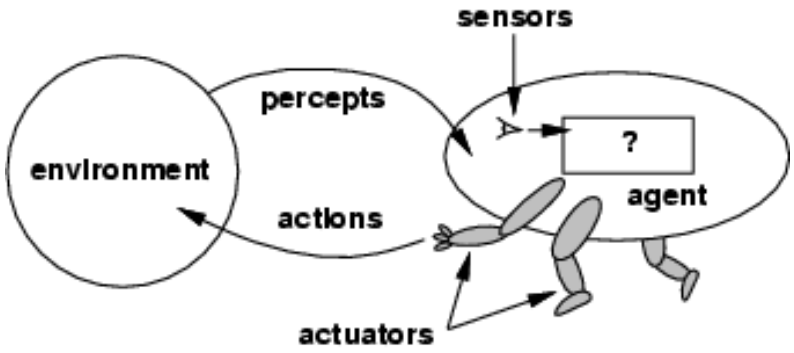
- Un ***agente*** è qualcosa che percepisce ed agisce in un ambiente, che fa qualcosa relazionandosi con l'ambiente.
  - Agenti umani
  - Agenti robotici
  - Agenti software

- Il concetto di ***agente*** è solo uno strumento adatto all'analisi dei sistemi e non una caratterizzazione assoluta che divide il mondo in agenti e non agenti!

Un agente (concetto) è uno strumento adatto all'analisi di sistemi



Agenti ed Ambienti



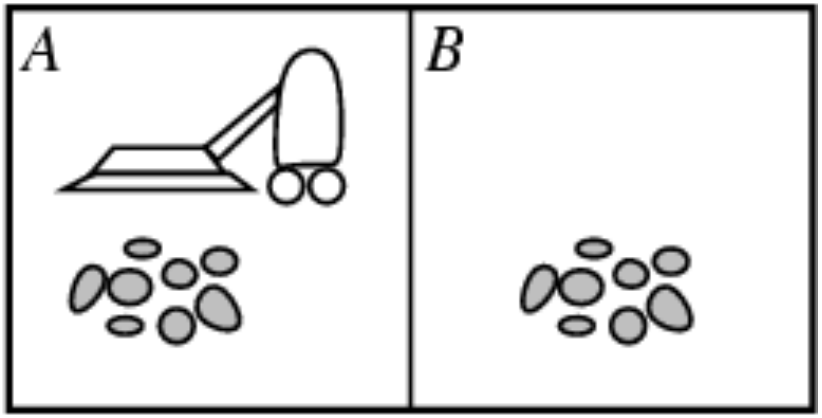
- La funzione agente mappa la storia delle percezioni in azioni:

$$f: \mathcal{P}^{\star} \rightarrow \mathcal{A}$$

- Il programma agente è in esecuzione sull'architettura per implementare *f*

agente = architettura + programma

L'aspirapolvere



- Percezioni: posizione e contenuto, ad. es., [A, Sporco]
- Azioni: *Sinistra*, *Destra*, *Aspira*, *NoOp*

Un agente aspirapolvere

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Quale è la giusta funzione?

Razionalità

- Viene fissata una misura di prestazione che valuta la sequenza di stati dell'ambiente (percezioni)
  - +1 per ogni spazio pulito in tempo T ?
  - +1 per ogni spazio pulito per istante di tempo, -1 per spostamento ?
  - penalizzazione se ci sono >k riquadri sporchi?
- Un agente razionale sceglie una qualunque azione che massimizza il valore atteso della misura di prestazione data la sequenza di percezioni ottenuta fino all'istante corrente
- Razionalità ≠ Onniscienza
  - le percezioni potrebbero non fornire tutte le informazioni rilevanti
- Razionalità ≠ Chiaroveggenza
  - l'effetto delle azioni potrebbe essere diverso da quello che mi aspetto
- Razionalità ≠ Successo
- Razionalità ⇒ esplorazione, apprendimento, autonomia

Agenti

Un ***agente intelligente*** è un sistema che

- agisce intelligentemente/razionalmente,
- fa ciò che è appropriato per la situazione e per i suoi obiettivi,
- è flessibile al variare dell'ambiente e degli obiettivi,
- eventualmente impara dall'esperienza e fa scelte appropriate date le sue percezioni (limitate) ed i limiti della capacità computazionale.

Agire razionalmente

Agire razionalmente significa agire:

- per raggiungere i propri obiettivi,
- date le proprie conoscenze,
- in funzione della sequenza di percezioni ricevute,
- date le azioni che si è in grado di compiere.

Le nuove percezioni non influenzano la descrizione (rappresentazione) del mondo dell'agente

Agire razionalmente (2)

- Un agente razionale fa la *cosa giusta*, ovvero quell'azione che procurerà all'agente il *maggior successo* massimizzando la misura di prestazione attesa

- *Misura di prestazione* (valutare – *come e quando* – il successo atteso, considerato ciò che è stato percepito).

PEAS

- PEAS: Performance measure, Environment, Actuators, Sensors
- La specifica del PEAS è il primo passo nella progettazione di un agente intelligente

- Consideriamo ad esempio la progettazione di un pilota automatico per taxi:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS

- Consideriamo ad esempio la progettazione di un **pilota automatico per taxi**:
  - Performance measure:

    sicuro, veloce, ligio alla legge, viaggio confortevole, profitti massimi
  - Environment

    strada, altri veicoli nel traffico, pedoni, clienti
  - Actuators

    sterzo, acceleratore, freni, frecce, clacson
  - Sensors

    telecamere, sonar, tachimetro, GPS, contachilometri, accelerometro, sensori sullo stato del motore

PEAS

- Agent: **Sistema di diagnosi medica**

- Performance measure: paziente sano, minimizzare i costi e le denunce
- Environment: Paziente, ospedale, staff medico
- Actuators: Schermo per visualizzare domande, test, diagnosi, trattamenti, documentazioni
- Sensors: Tastiera per l'inserimento dei sintomi, dei risultati dei test, delle risposte del paziente

PEAS

- Agente: **Robot selezionatori di parti meccaniche**

- Performance measure: percentuale di pezzi inseriti nei contenitori giusti
- Environment: nastro trasportatore con parti meccaniche, contenitori
- Actuators: braccio meccanico con manipolatore
- Sensors: Camera, sensori di posizionamento del braccio meccanico

PEAS

- Agent: **tutor interattivo per lo studio dell'inglese**

- Performance measure: massimizzare il punteggio del test dello studente
- Environment: insieme di studenti, scuola
- Actuators: display per proporre esercizi, fornire suggerimenti e correzioni
- Sensors: tastiera

PEAS

- Agent: **internet shopping**

- Performance measure
  - prezzo, qualità, appropriatezza, efficienza
- Environment:
  - siti web, venditori, compratori
- Actuators:
  - display per l'utente, seguire URL, riempire form
- Sensors:
  - pagine HTML (testo, grafica, scripts)

Agenti ed ambienti

- Indipendentemente dal tipo di agente, la natura della sua connessione con l'ambiente è di un unico tipo:

le azioni sono fatte dall'agente sull'ambiente, che a sua volta fornisce percezioni all'agente

- Lo stato dell'agente dipende dalle sue percezioni
- Lo stato dell'ambiente dipende dalle azioni di ogni singolo agente

Osservabilità

- Completamente osservabile: se i sensori dell'agente misurano tutti gli aspetti rilevanti per la scelta dell'azione
  - La rilevanza dipende dalla misura di prestazione
- Parzialmente osservabile: sensori inaccurati, mancanza di alcuni sensori, rumore
- Non osservabile: non ci sono sensori

Agente singolo o multiagente

- Esempio: agente che risolve un cruciverba vs. agente che gioca a scacchi
- Distinzione tra altro agente e oggetto parte dell'ambiente
- B agente se massimizza una misura di prestazione il cui valore dipende dal comportamento di A
- Scacchi: ambiente multi-agente competitivo
- Traffico: ambiente multi-agente parzialmente cooperativo (evitare gli incidenti massimizza la misura di prestazione di tutti, trovare un parcheggio libero no)

Ambiente deterministico/stocastico

- Deterministico: lo stato successivo dell'ambiente è determinato dallo stato corrente e dall'azione dell'agente
- Completamente osservabile e deterministico: non c'è incertezza
- Incerto: se non completamente osservabile o non deterministico
- Stocastico: incertezza descritta tramite probabilità
- Non deterministico: azioni caratterizzate da risultati possibili, ma senza uso di probabilità

Ambiente episodico/sequenziale

- Episodico: la scelta di una azione non dipende dalle scelte fatte in ”episodi” precedenti
- Agente che identifica i pezzi difettosi in una catena di montaggio
- Ambiente sequenziale: una decisione può influenzare le successive
- Taxi, scacchi, ecc.
- In un ambiente sequenziale, l’agente deve ”pensare in avanti”

Ambiente dinamico/statico

- Dinamico: se l'ambiente può cambiare mentre l'agente pensa a che azione eseguire
- Semi-dinamico: l'ambiente non cambia, ma la valutazione delle prestazioni dell'agente si
- Guidare un taxi è dinamico, scacchi con orologio semi-dinamico, cruciverba statico

Altre caratteristiche di un ambiente

- Discreto/continuo: Stato dell'ambiente, gestione del tempo, percezioni e azioni (scacchi discreto, taxi continuo)
- Noto/ignoto: stato di conoscenza dell'agente delle leggi dell'ambiente (può essere noto ma parzialmente osservabile, es. solitario, oppure ignoto ma completamente osservabile, es. nuovo videogioco)
- Caso più difficile: parzialmente osservabile, multi-agente, stocastico, sequenziale, dinamico, continuo, ignoto (es. guidare taxi).

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile			
Deterministico			
Episodico			
Statico			
Discrete			
Single agent			

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile	Si	Si	No
Deterministico			
Episodico			
Statico			
Discrete			
Single agent			

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile	Si	Si	No
Deterministico	Si	Si	No
Episodico			
Statico			
Discrete			
Single agent			

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile	Si	Si	No
Deterministico	Si	Si	No
Episodico	No	No	No
Statico			
Discrete			
Single agent			

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile	Si	Si	No
Deterministico	Si	Si	No
Episodico	No	No	No
Statico	Semi	Si	No
Discrete			
Single agent			

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile	Si	Si	No
Deterministico	Si	Si	No
Episodico	No	No	No
Statico	Semi	Si	No
Discreto	Si	Si	No
Single agent			

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

Tipi di ambienti

	Scacchi con tempo	Scacchi senza tempo	Guida Taxi
Osservabile	Si	Si	No
Deterministico	Si	Si	No
Episodico	No	No	No
Statico	Semi	Si	No
Discrete	Si	Si	No
Single agent	No	No	No

- Il tipo di ambiente determina come progettare l'agente
- Il mondo reale è (ovviamente) **parzialmente osservabile**, **stocastico**, **sequenziale**, **dinamico**, **continuo**, **multi-agente**

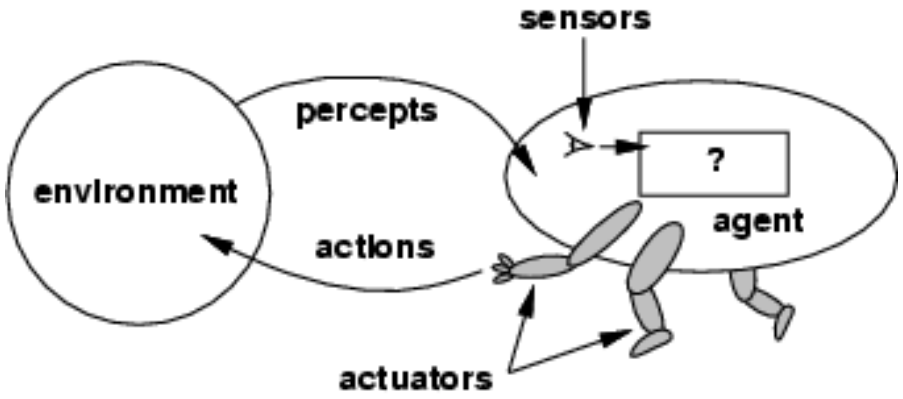
Tipi di ambienti

	Internet shopping	Giocare a calcio	Fare un salto in alto
Osservabile	No	No	Si
Deterministico	Parzialmente	No	No
Episodico	No	No	No
Statico	Semi	No	Si
Discrete	Si	No	No
Single agent	Si (eccetto aste)	No	Si

Progettazione di un agente

- Progettare il *programma* di un agente significa implementare la *funzione* corrispondenza dalle percezioni alle azioni all'interno di un ambiente e con prestazioni predefinite

Agente = architettura + programma



Progettazione di un agente

- La ***funzione agente*** è una descrizione matematica astratta, formale
- Il ***programma agente*** è una sua implementazione concreta in esecuzione sull'architettura dell'agente
- L'***architettura*** (*meccanismo di calcolo*) fornisce le percezioni al programma, esegue il programma e trasmette agli attuatori le azioni scelte dal programma.

Progettazione di un agente

Prerequisiti:

Conoscenza dettagliata di:

- tutte le *percezioni* e le possibili *azioni conseguenti*
- *ambiente* (= il problema di cui l'agente costituisce la soluzione) per scegliere tra tutte le possibili azioni quelle che saranno applicabili
- *misura delle prestazioni* (definita a priori per non alterare la valutazione)
- *obiettivi* (per selezionare/valutare percezioni ed azioni)

Programma di un agente

Schema:

- A fronte della conoscenza precedente e delle percezioni provenienti dall'ambiente [che "aggiornano" eventualmente la conoscenza <> *incrementalità*] vengono generate delle azioni congruenti i risultati delle "procedure di decisione".

- L'obiettivo e la misura delle prestazioni vengono applicate esternamente all'agente.

Schema di agente (semplice)

```
function SKELETON-AGENT(percept) returns action
  static: memory, the agent's memory of the world

  memory ← UPDATE-MEMORY(memory, percept)
  action ← CHOOSE-BEST-ACTION(memory)
  memory ← UPDATE-MEMORY(memory, action)
  return action
```

- Mantiene memoria della sequenza di percezioni (*incrementalità*)
- Si possono definire sistemi ad agenti cooperanti per applicazioni più complesse

Agente basato su tabella

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

- Tabella totalmente e precedentemente specificata
- Agente tiene conto della sequenza delle percezioni consultando la tabella
- La tabella non viene aggiornata *(in quanto totalmente e precedentemente specificata)*

Agente basato su tabella

Vantaggi:

- Progettazione semplice, completa, efficace

Svantaggi:

- Dimensioni sempre crescenti (funzione della complessità dell'applicazione)
  - Per giocare a scacchi tabella con 35^{100} righe!!
- Time-consuming per il progettista
- Nessuna autonomia è prevista per l'agente (non necessaria a causa della totale definizione delle corrispondenze)
- Di difficile aggiornamento, apprendimento complesso.

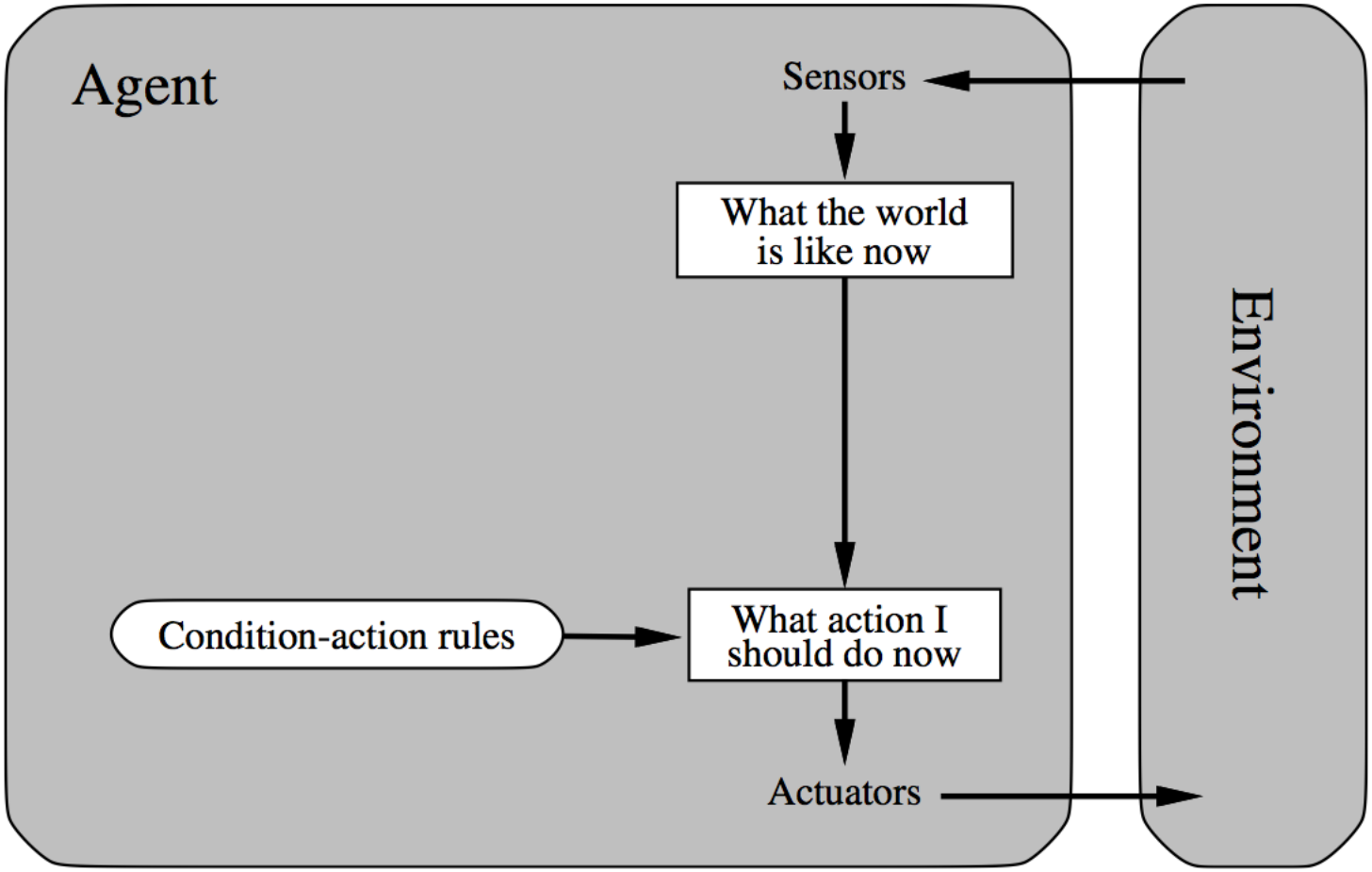
In ambiti limitati funzionano bene (ruolo dell'ambiente)

Tipologie di programmi di agente

- *Agente reattivo semplice* (rispondono alle percezioni)
- *Agente reattivo basato su modello* (considerano l'evolversi del mondo circostante)
- *Agenti basati su obiettivi* (agiscono per raggiungere i propri obiettivi)
- *Agenti basati su utilità* (cercano di massimizzare le propria utilità)

- Sono agenti specializzati, ovvero ottimizzati rispetto a ciò che vogliamo misurare

Agente reattivo semplice



regole di produzione ***if-then-else***

Agente reattivo semplice

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

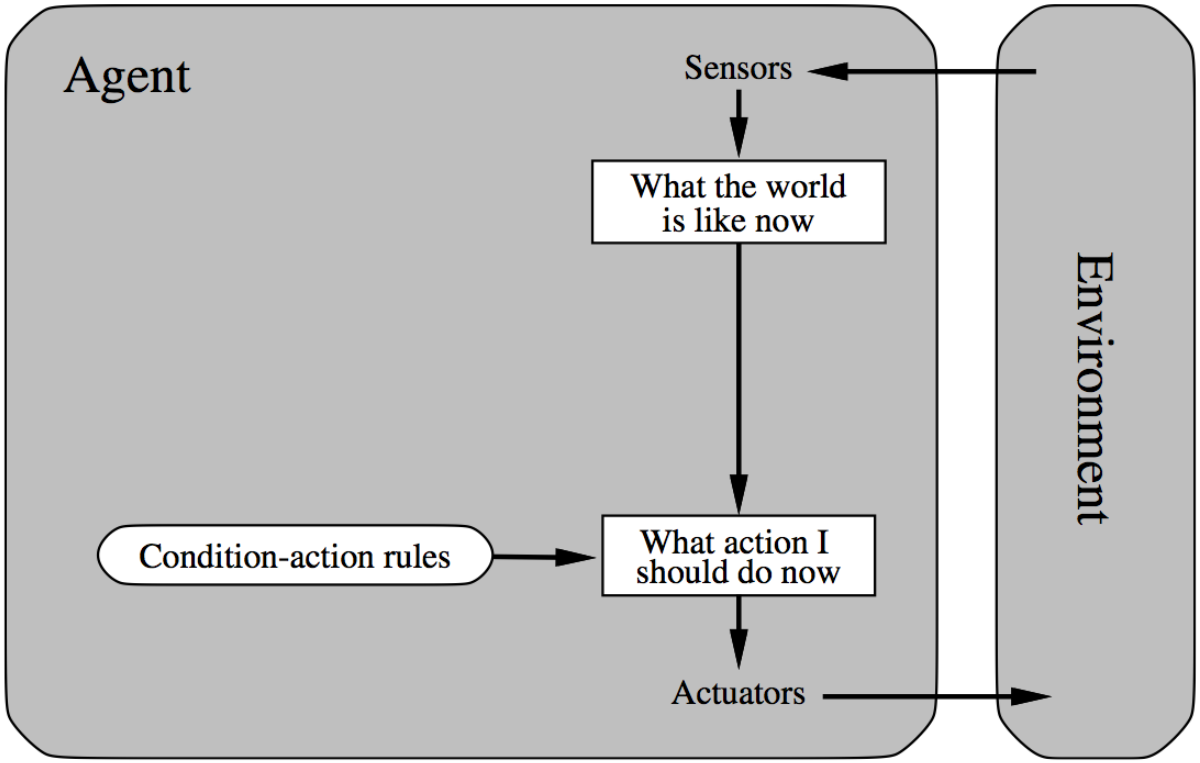
- Trova la prima regola di produzione in accordo con la situazione corrente (definita da percezione in un ambiente osservabile). Compie azione associata.
- Cambiando le regole, cambia il tipo di agente.

Agente per l'aspirapolvere

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

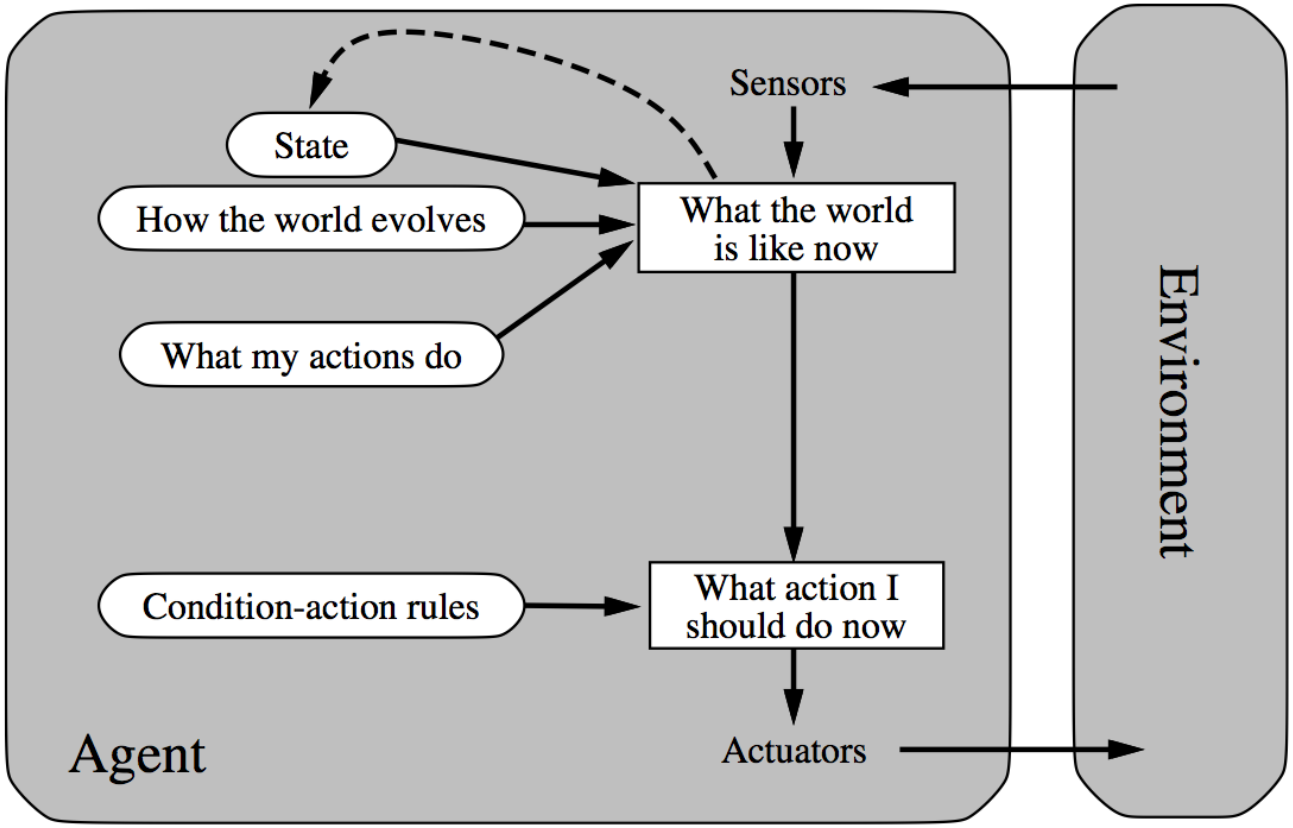
if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Agente reattivo semplice



- Problema: se l'ambiente è parzialmente osservabile può fallire (randomizzare)
- Soluzione migliore: tenere traccia delle percezioni passate → stato interno

Agente basato su modello



- La conoscenza che l'agente ha del mondo (*state*), di come esso evolve (*how…)* e di cosa fanno le proprie azioni *(what…),* si chiama *modello del mondo*.

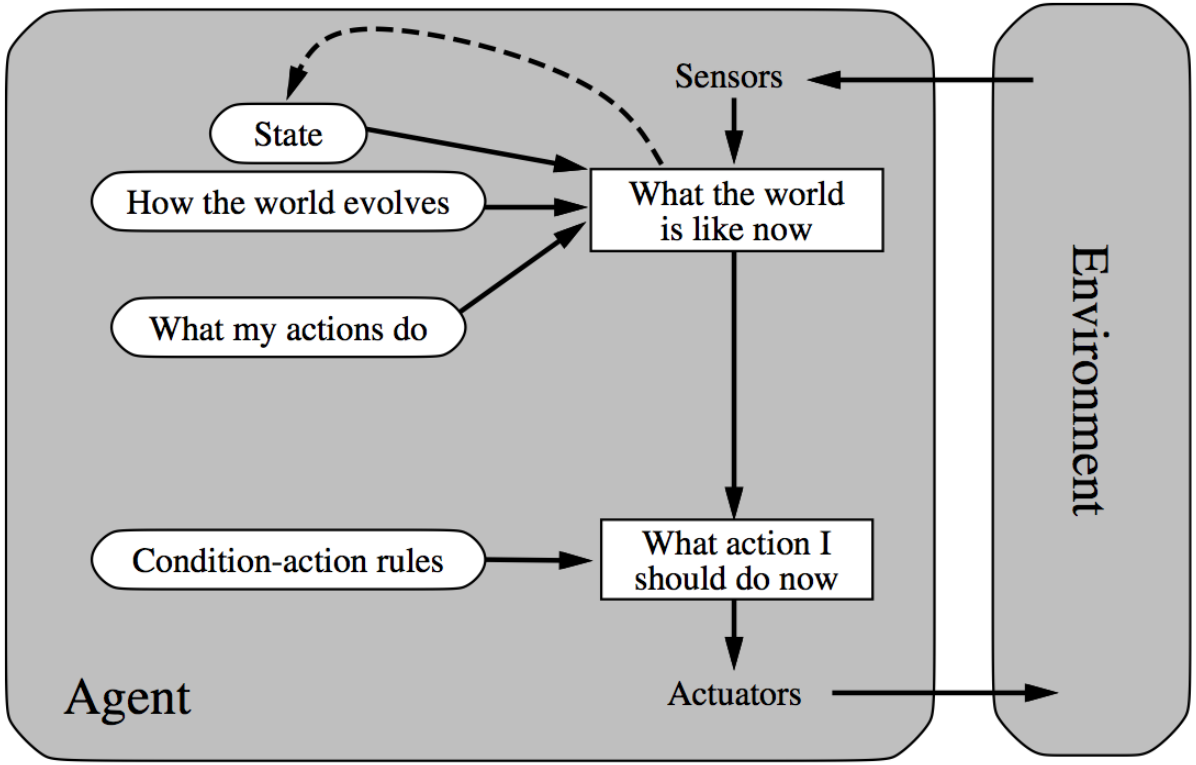
Agente basato su modello

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

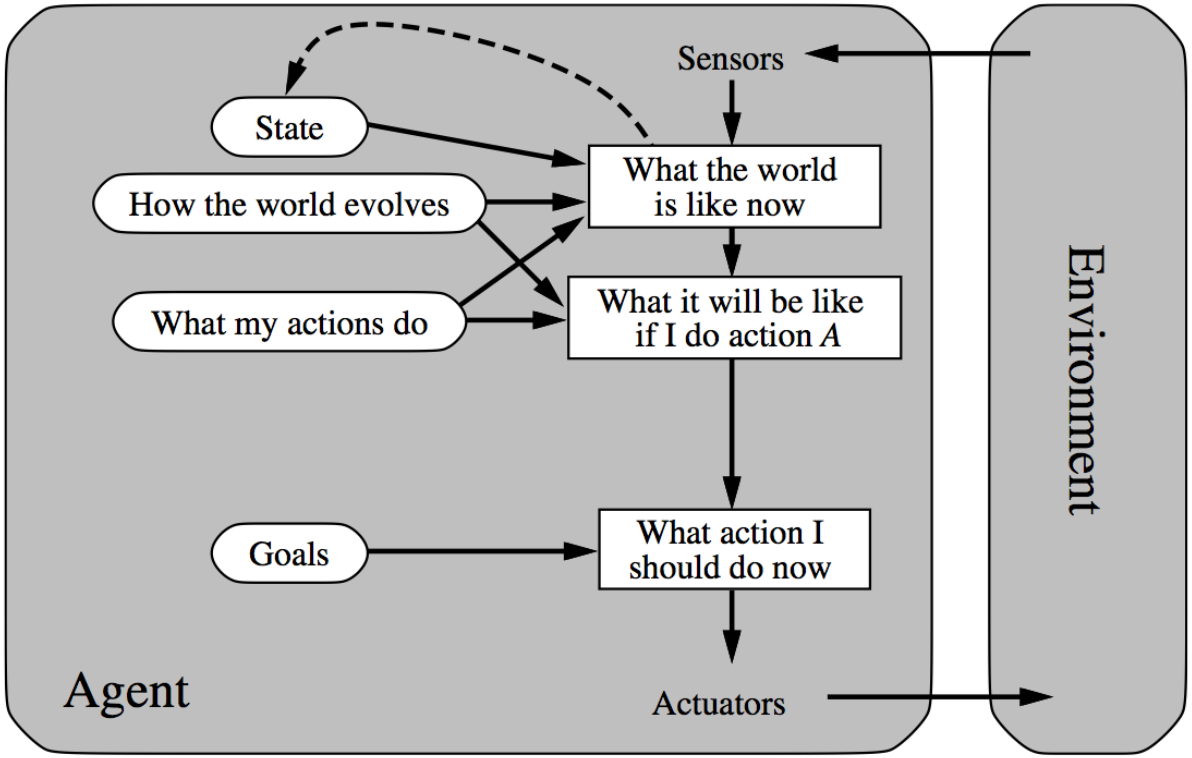
- Trova regola in accordo situazione corrente (definita da percezioni e stato interno)
- Compie azione associata
- Aggiorna lo stato del mondo

Agente basato su modello



- Problema: poco flessibile perché il comportamento è codificato direttamente nelle regole!
- Soluzione: introduzione di goal

Agenti basati su obiettivi (goal) / Pianificatori



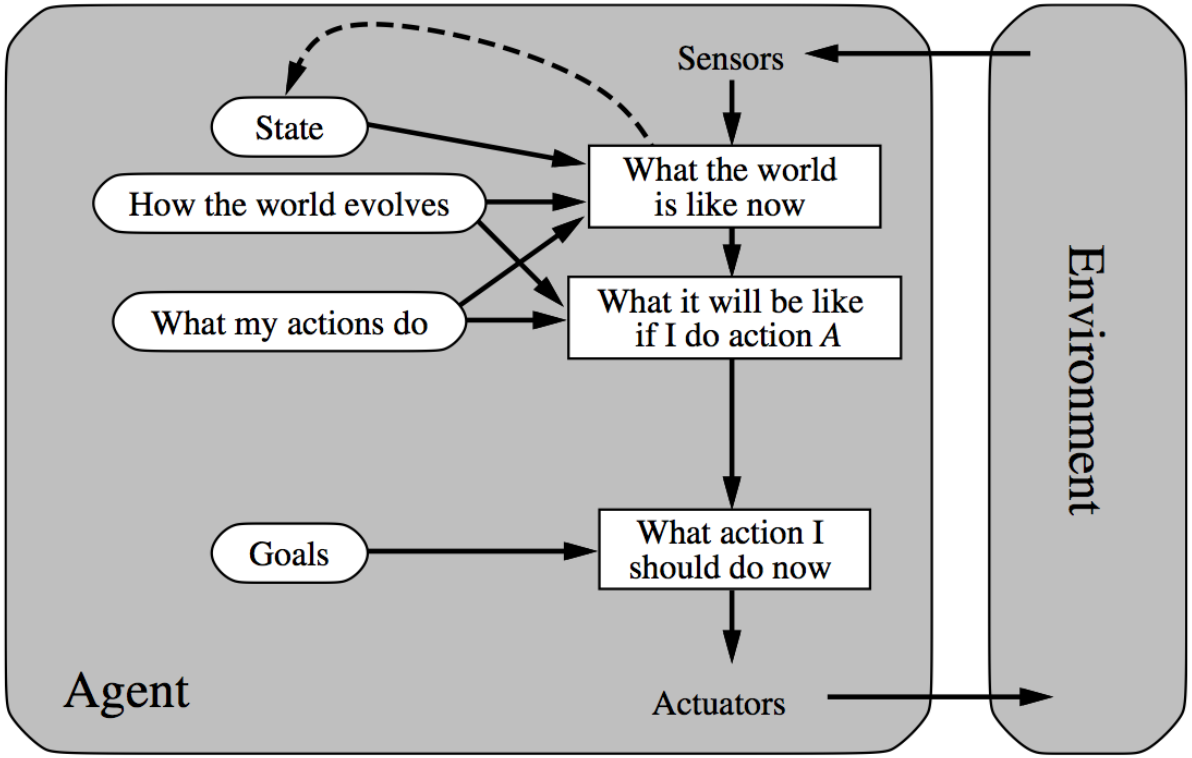
- La ricerca di una soluzione e la pianificazione permetteranno di identificare la sequenza di azioni che mettono in grado un agente di raggiungere i propri obiettivi

Agenti basati su obiettivi (goal) / Pianificatori

function GOAL-BASED-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    goal, a description of the desired goal state
    plan, a sequence of actions to take, initially empty
    action, the most recent action, initially none

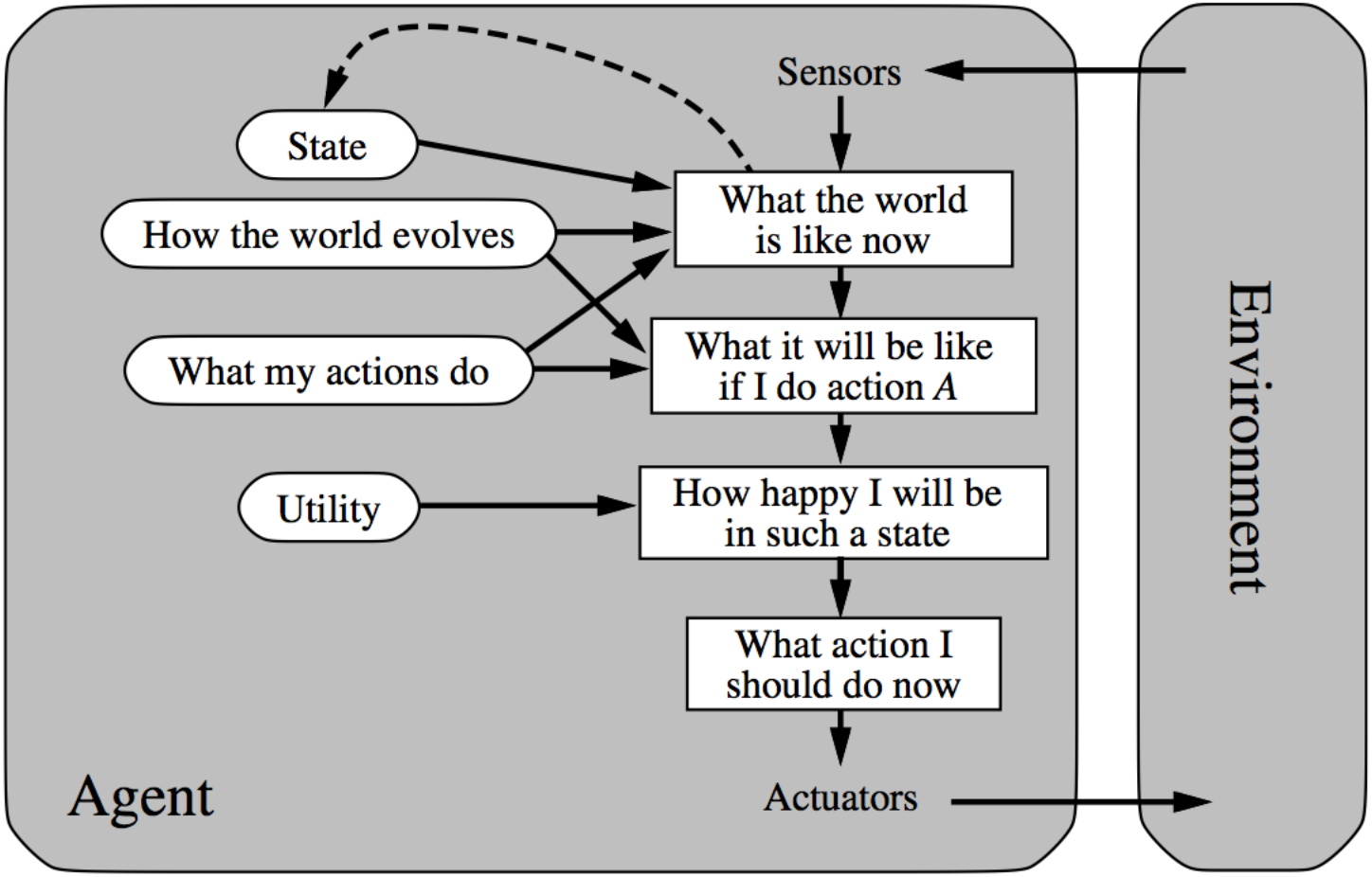
  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  if GOAL-ACHIEVED(*state*,*goal*) **then return** a null action
  if *plan* is empty **then**
plan ← PLAN(*state*,*goal*,*model*)
  action ← FIRST(*plan*)
  plan ← REST(*plan*)
  return *action*

Agenti basati su obiettivi (goal) / Pianificatori



- Problema: se ho più goal in conflitto fra loro o che non riesco a raggiungere pienamente?
- Soluzione: misura di utilità

Agenti basati su utilità
Teoria delle decisioni



Agenti basati su utilità
Teoria delle decisioni

- L'***utilità*** è una funzione predefinita che associa ad uno stato (o sequenza di stati se stiamo misurando l'utilità a lungo termine) dell'agente un numero reale che descrive il grado associato di *"felicità"*

- *Uno stato del mondo preferibile ad un altro ha per l'agente una maggiore utilità.*

- Si definisce *una funzione di utilità.*

Agenti basati su utilità
Teoria delle decisioni

function UTILITY-BASED-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    utility − *function*, a description of the agent's utility function
    plan, a sequence of actions to take, initially empty
    action, the most recent action, initially none

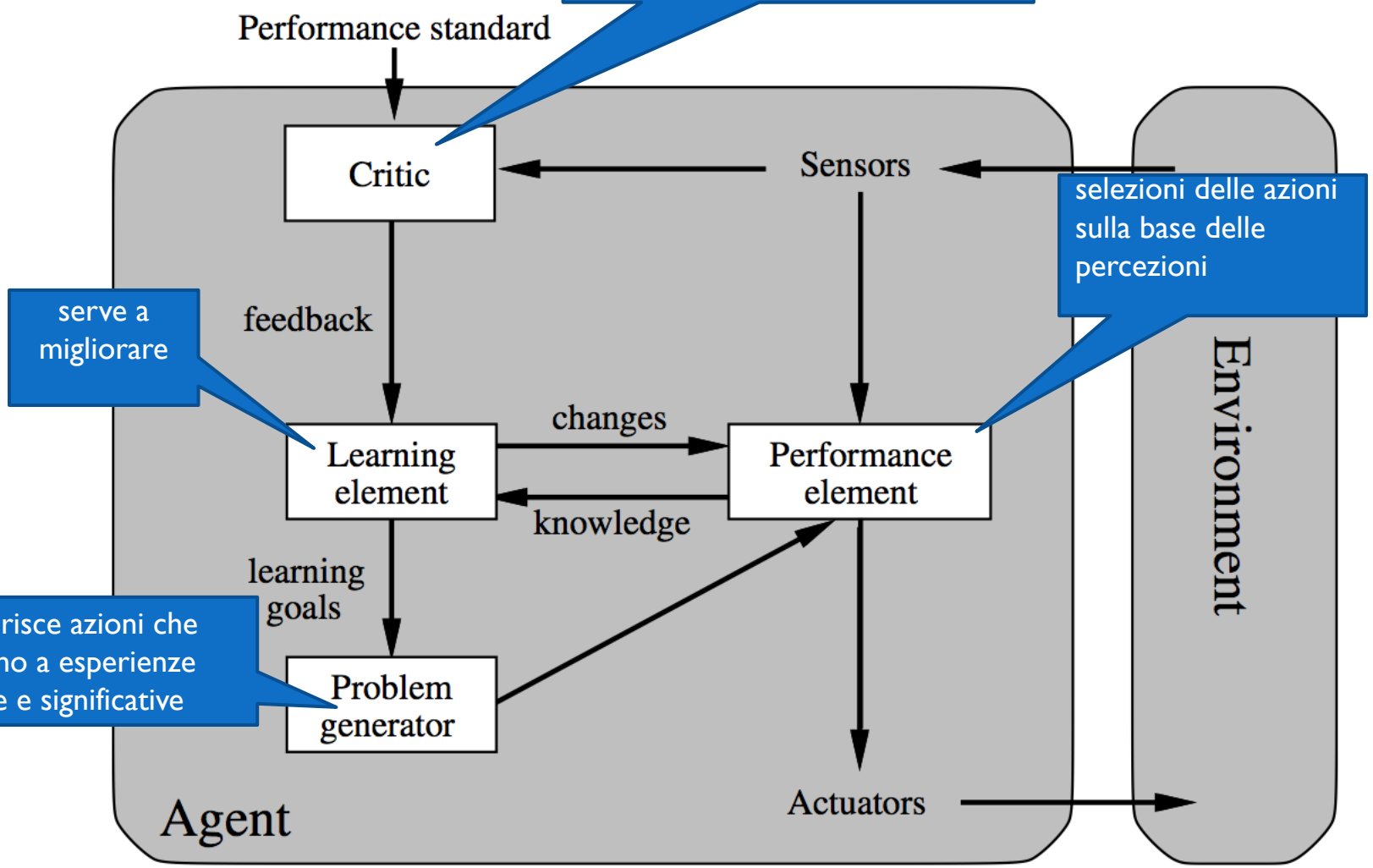
state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
if *plan* is empty **then**
plan ← PLAN(*state*,*utility* − *function*,*model*)
action ← FIRST(*plan*)
plan ← REST(*plan*)
return *action*

Apprendimento

- Tutti gli agenti possono migliorare le loro prestazioni mediante *l'apprendimento*.

- Per apprendimento di un agente intelligente si intende il processo che modifica ogni suo componente affinché si accordi meglio con l'informazione di feedback disponibile, migliorando così le prestazioni globali dell'agente.

Agenti cap ndere

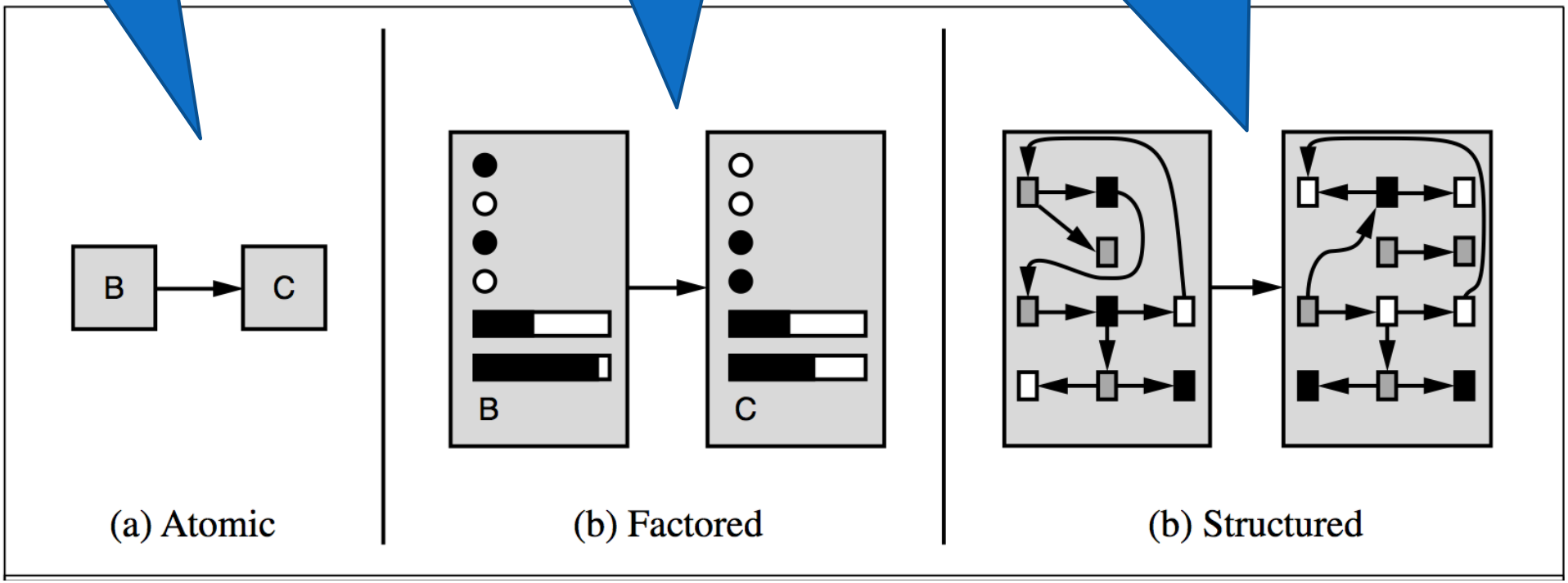


Rappresentare gli stati

Usato da:
Algoritmi di ricerca
HMM

Usato da algoritmi di:
CSP
Pianificazione
Reti bayesiane

Usato da:
logica del primo ordine
Modelli probabilistici
Apprendimento basato sulla conoscenza
NLP



Argomenti trattati in questa lezione

- Definizione di un agente
- Ambiente e razionalità
- Progettazione di un agente
- Diverse tipologie di agenti