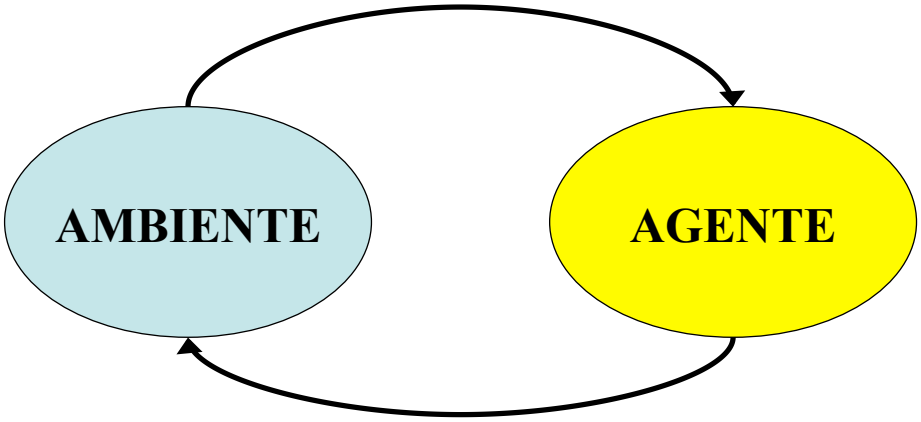
Agente Artificiale Intelligente (razionale)

Percezioni attraverso sensori

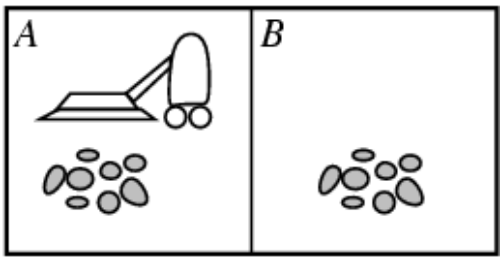


Azioni attraverso "attuatori"
(bracci meccanici, ruote, ecc.)

Quando un Agente Artificiale è Razionale?

- Quando fa la "***cosa giusta***"
- Cosa giusta = ***azione che causa maggior successo***
- Come valutare un agente?
- ***Misura delle prestazioni*** (*grado di successo*)
  - deve essere imposta **dall'esterno**
  - dipende dalla sequenza di **percezione**
  - dipende dalla **conoscenza** dell'agente sul suo ambiente
  - dipende dalle **azioni** che l'agente può eseguire
- E' meglio progettare misure in base all'effetto che si desidera ottenere sull'ambiente, piuttosto che su come si pensa debba comportarsi l'agente

Esempio Robot Aspirapolvere



- **Percezioni**: *posizione e contenuto*, ad es. [A,Sporco]
- **Azioni**: *VaiSinistra*, *VaiDestra*, *Aspira*, *NoOp*
- **Performance**: *quanto ha pulito, quanto ha consumato, quanto tempo ha impiegato, ecc*.

Agente Razionale ***Ideale***

"*Per ogni sequenza di percezioni possibile, un agente ragionale dovrebbe compiere qualsiasi azione che si aspetta massimizzare la sua misura delle prestazioni, sulla base della sequenza di percezione e di qualsiasi conoscenza predefinita dall'agente*"

S. Russell e P. Norvig (1995)

Esempi: Un orologio è un agente intelligente?
Un robot aspirapolvere?
Un robot classificatore?

NB: *Razionalità non vuol dire conoscenza completa!*

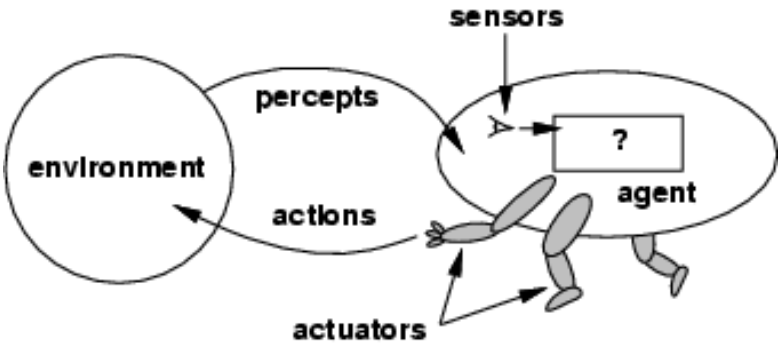
Razionalità ≠ Omniscienza

Un agente è **omnisciente** se conosce perfettamente le conseguenze delle sue azioni.

Esempio: *Un Robot che sa attraversare una strada trafficata senza alcuna collisione con altri veicoli non è omnisciente se non è in grado di anticipare la possibile caduta del portellone di un aereo che transita sopra di lui nel momento in cui attraversa la strada, facendo fallire il suo compito...*

Ma il nostro Robot è comunque razionale!

Struttura Agente Intelligente (Razionale)



- L' agent function mappa sequenze di percezioni in azioni: $f : \mathcal{P}^* \rightarrow \mathcal{A}$
- L'agent program gira su una architettura fisica hardware (con relativo sw) per implementare *f*
- **agente = architettura + programma**

Agent Function & Program

Il comportamento (programma) dell'agente dipende da

1) Sequenza di percezione
2) Conoscenza precostituita
3) Propria esperienza (apprendimento)

Come implementare *f* ?

Può essere impossibile *calcolare e memorizzare f*
(troppo tempo, troppa memoria: esempio scacchi....)

Spesso meglio implementare f con un programma (di uno o più algoritmi)

Inoltre, se l'agente è ***autonomo*** può cambiare *f* attraverso ***forme di apprendimento***

Ambiente Operativo *(PEAS)*

La progettazione di un Agent Program dipende dal suo *ambiente operativo*:

- Misura di **Prestazioni** (***P***erformance)
- Tipo di **Ambiente** (***E***nviroment) fisico o artificiale/ virtuale in cui agisce e con cui interagisce
- **Attuatori** (***A***ctuators) per eseguire azioni nell'ambiente
- **Sensori** (***S***ensors) per percepire informazioni sull'ambiente

Esempio: Taxi Driver Automatizzato

- Per progettarlo bisogna considerare:
  - **Misura prestazioni**: Sicurezza, tempo, legalità, confort viaggio, profitto,...
  - **Ambiente (fisico)**: Strade, altri veicoli, pedoni, cliente, ...
  - **Attuatori**: Volante, accelleratore, freni, luci, segnale sonoro,
  - **Sensori**: Telecamera, GPS, sensore luce, sensori motore ecc..

Esempio: Sistema di Diagnosi Medica

- **Performance**: salute paziente, costi economici, costi sociali, costi legali,tempi,...
- **Ambiente**: paziente, ospedale, medici, staff,...
- **Attuatori**: video (domande, test, diagnosi, cure, referti, ecc.)
- **Sensori**: tastiera, acquisizione immagini (input sintomi, risultati tests, risposte paziente, ...)

Esempio: Robot Raccoglitore/ Classificatore

- **Performance**: Percentuali di parti nei cesti
- **Ambiente**: Nastro trasportatore con pezzi da raccogliere e porre in cesti differenti
- **Attuatori**: Mani e bracci meccanici
- **Sensori**: telecamere, sensori per rilevare forma-colore-dimensione-ecc..

Proprietà di un Ambiente

- Accessibile/non accessibile: i sensori rivelano tutta l'informazione ***rilevante*** necessaria per decidere l'azione "oggettivamente" migliore?
  => *Osservabilità completa/parziale/nulla*.
- Deterministico/non deterministico: gli effetti delle azioni sono noti *all'agente* con *certezza*? Se si, lo sono anche per le azioni degli *altri* agenti?
- Episodico/non episodico: la qualità di un'azione scelta dipende dalle precedenti?

Proprietà di un Ambiente

- Statico/dinamico/semidinamico: il mondo cambia mentre l'agente decide? Il tempo di decisione è rilevante per la performance?
- Discreto/continuo: il numero di percezioni e azioni possibili è praticamente illimitato?
- Singolo Agente/Multi Agente: se ci sono più agenti, essi agiscono in modo *cooperativo (squadra di robot calciatori)* o *competitivo (scacchi)*? Sono controllati *centralmente* o in modo *distribuito*? Come/cosa possono *comunicare e con chi*?

Tipi di Ambiente

ESEMPI	**Chess with a clock**	**Chess without a clock**	**Taxi driving**
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- Il tipo di ambiente determina fortemente la progettazione dell'agente
- Il mondo reale in generale è parzialmente osservabile, stocastico, non episodico, continuo, multi-agente
- Ma esistono anche applicazioni per il mondo reale con ambienti più semplici

Agent functions e programmi

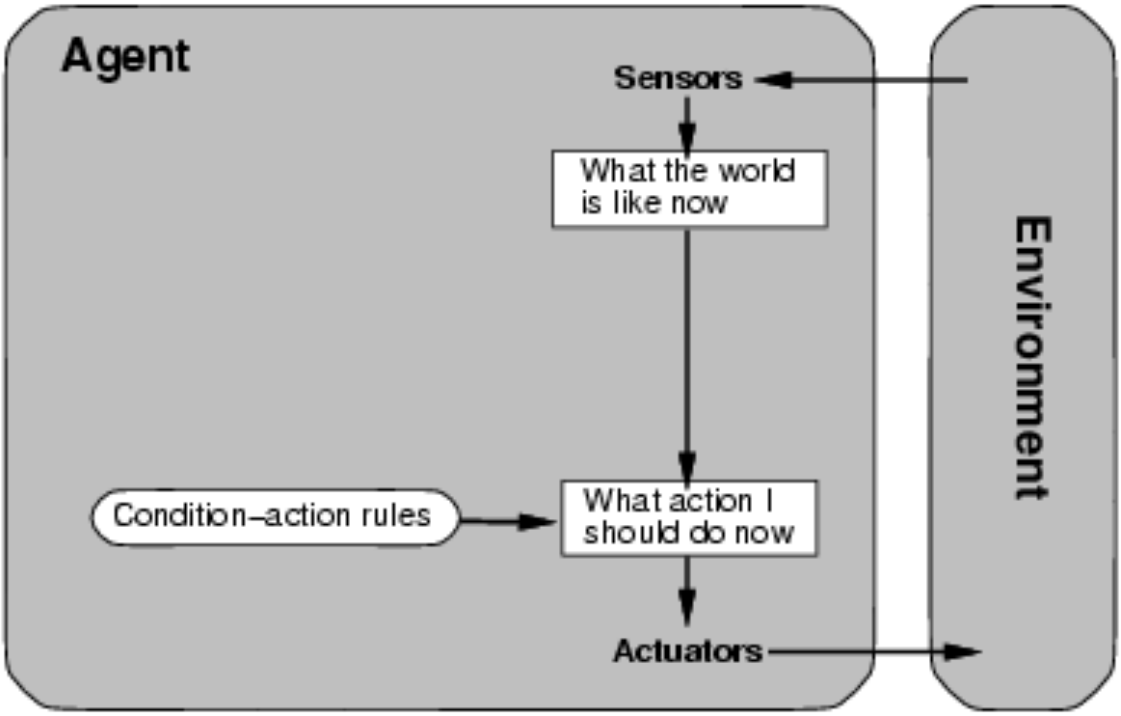
- Un agente è completamente specificato da una **agent function** (*f*: $\mathcal{P}^* \rightarrow \mathcal{A}$)
- *Problema*: come definire e implementare in *modo conciso* la funzione razionale dell'agente?
- L'uso di una semplice tabella molto spesso è impossibile (troppo grande!)

Tipologie di Agenti (razionali)

Cinque tipi base (in ordine di generalità):

- *Simple reflex agents*
- *Model-based reflex agents*
- *Goal-based agents*
- *Utility-based agents*
- *Learning agents*

Simple reflex agents



Simple reflex agents

Esempio Robot 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
```

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Simple reflex agents: algoritmo

```
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
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Simple reflex agents

E' adeguato per il robot aspirapolvere se può percepire **solo** la presenza/assenza di sporco? (non sa locazione)

Es regola: *Se "sensore-rileva-sporco=true" allora ASPIRA*

Cosa fare se ***non*** si osserva sporco??

=> Se robot è in A e va a sinistra o se B e va destra fallirà e non si fermerà mai!!

In generale: è adeguato solo se si può identificare la "decisione corretta" in base alla sola percezione corrente (cioè l'ambiente è completamente osservabile e le *percezioni precedenti sono irrilevanti*)

Simple reflex agents

Una possibile idea

Due regole applicabili quando non si rileva sporco::

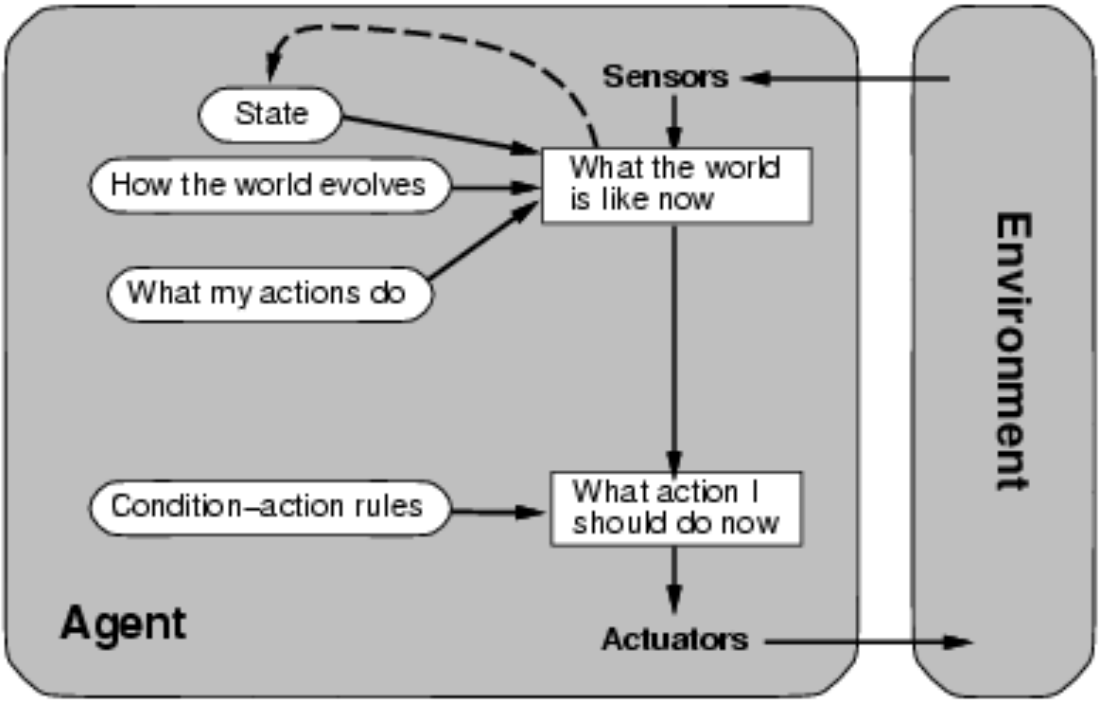
"se sensore-rileva-sporco=false allora MOVE-RIGHT"

"se sensore-rileva-sporco=false allora MOVE-LEFT"

Ne scelgo una *casualmente* w prima o poi tutto sarà pulito.

Ma come me ne accorgo e quando mi fermo?

Model-based reflex agents



Model-based reflex agents

Esempi:

- Robot Aspirapolvere che conosce posizione iniziale e mantiene posizione corrente

  Nel caso di due stanze: se pos-corrente diversa da pos iniziale e non rilevo sporco, allora ho finito (eseguo azione STOP)

- Taxi driver che percepisce suono ambulanza (si avvicina?)

- Luci rosse accese auto di fronte (luci posizione o freni?)

Model-based reflex agents: algoritmo

```
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
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

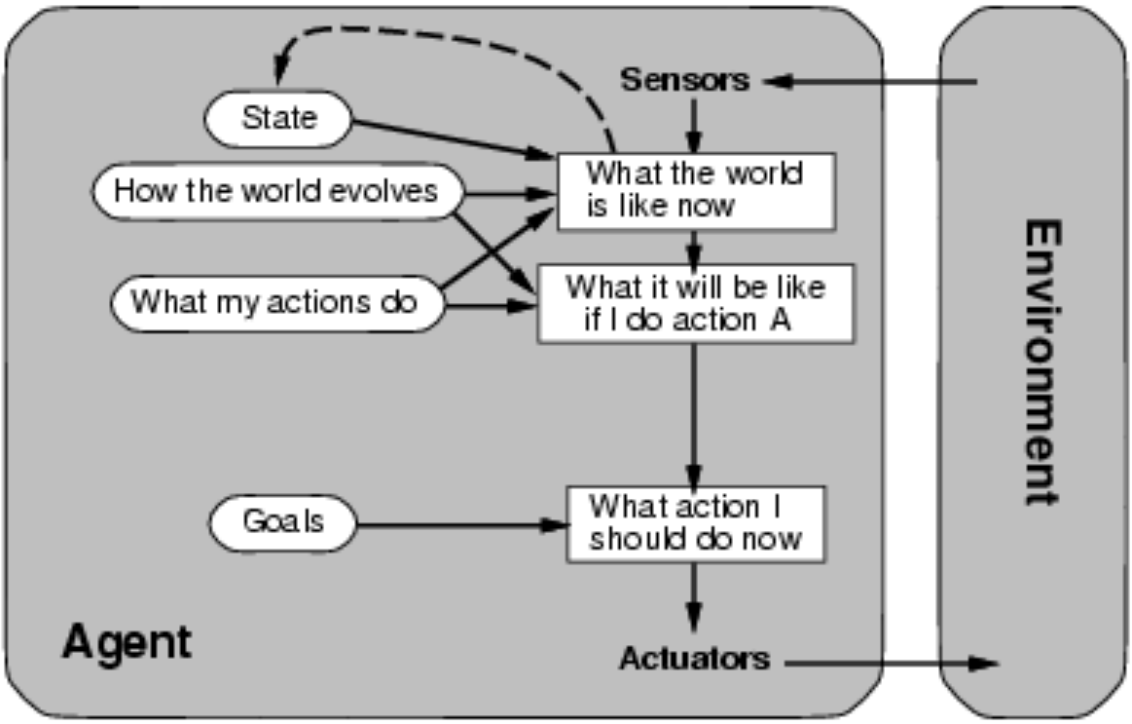
Goal-based agents

Spesso conoscere l'ambiente non è sufficente per decidere la corretta azione

Esempio: taxi driver a un incrocio: *dove vado??*

Per decidere la migliore azione bisogna considerare un **obiettivo esplicito** e ragionare su sequenze di azioni che lo possono conseguire a partire dallo stato dell'ambiente corrente osservato.

Goal-based agents



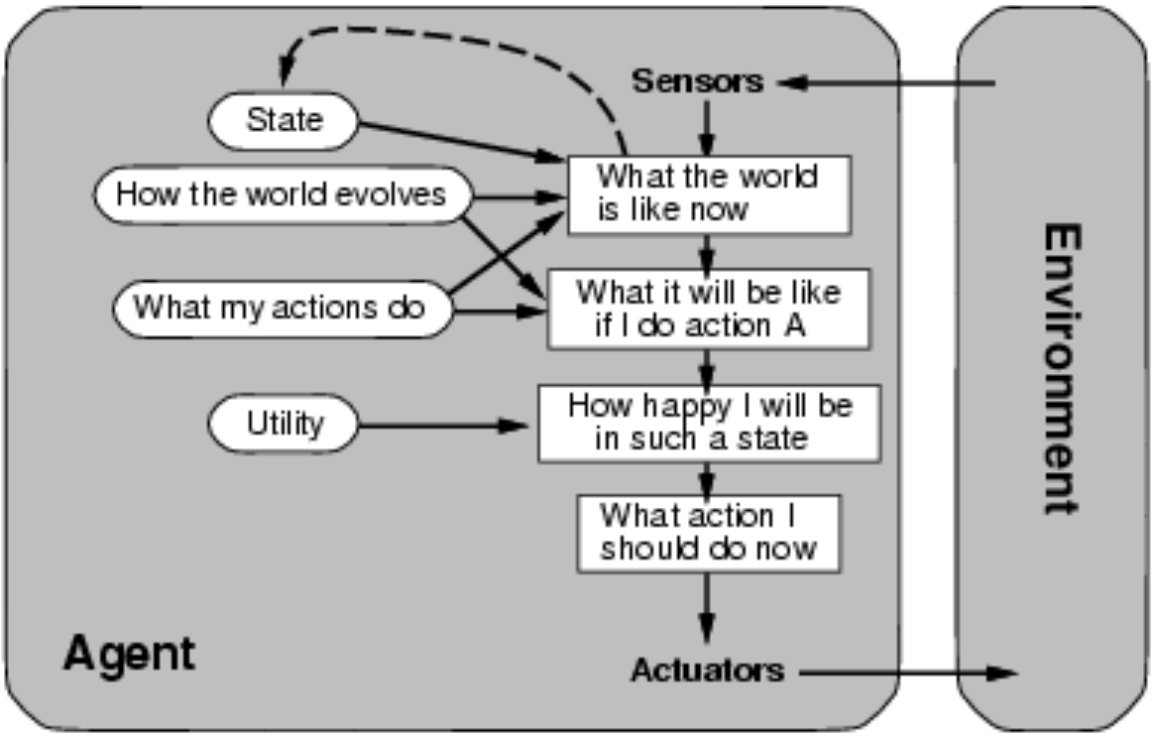
Utility-based agents

Spesso ci sono più possibilità per raggiungere un obiettivo

Esempio: taxi driver che ha più percorsi per raggiungere la destinazione e con differenti risorse (tempo, benzina, sicurezza, ecc..)

Per decidere la migliore azione bisogna considerare anche una **misura di utilità degli stati** del mondo raggiungibili attraverso le azioni dell'agente

Utility-based agents

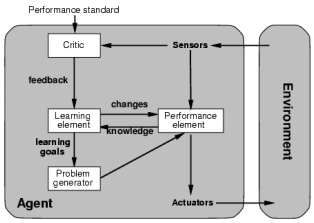


Learning agents

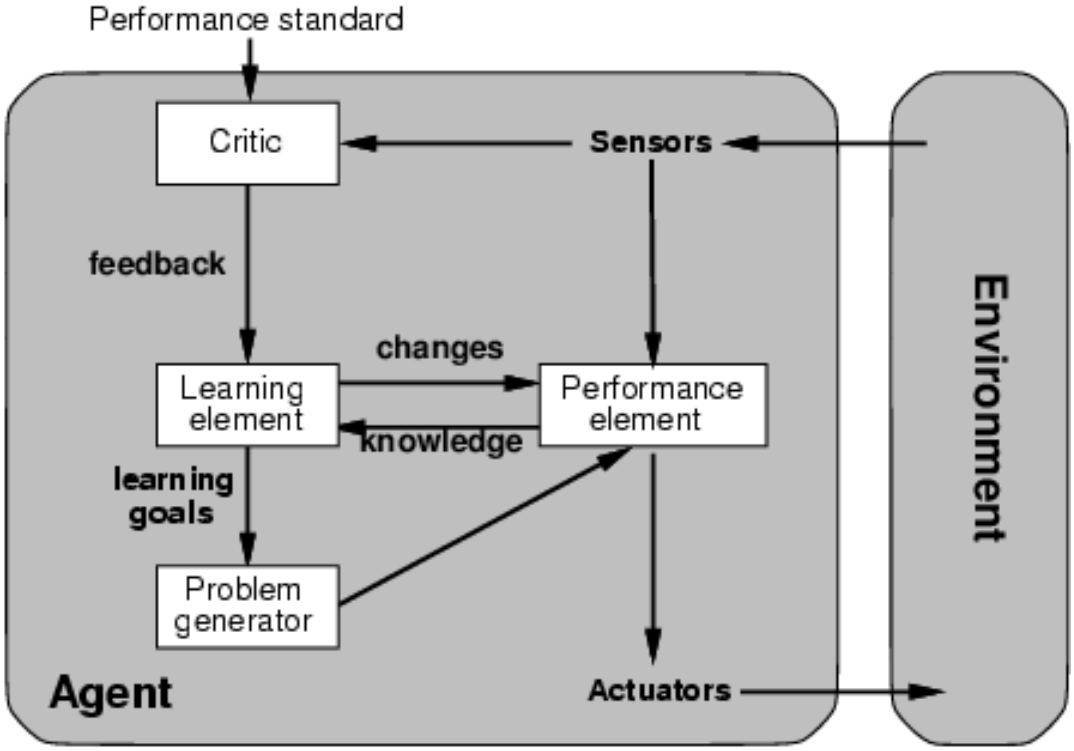
•Sono in grado di modificare il proprio comportamento (azione “migliore”) di fronte a situazioni (stato del mondo e percezioni) precedentemente incontrate

•Possono definire il comportamento da adottare in nuove situazioni a partire dal successo o insuccesso del proprio comportamente in situazioni “simili” già affrontate.

•Necessitano di un modulo di apprendimento basato su un’analisi critica dell’esperienza acquisita.



Learning agent



Esercizio per casa

Progettare un agente per il mondo del robot aspirapolvere (dettagli alla lavagna e/o su Russel & Norvig: es 2.11 ed.3, es 2.10 ed. 2)