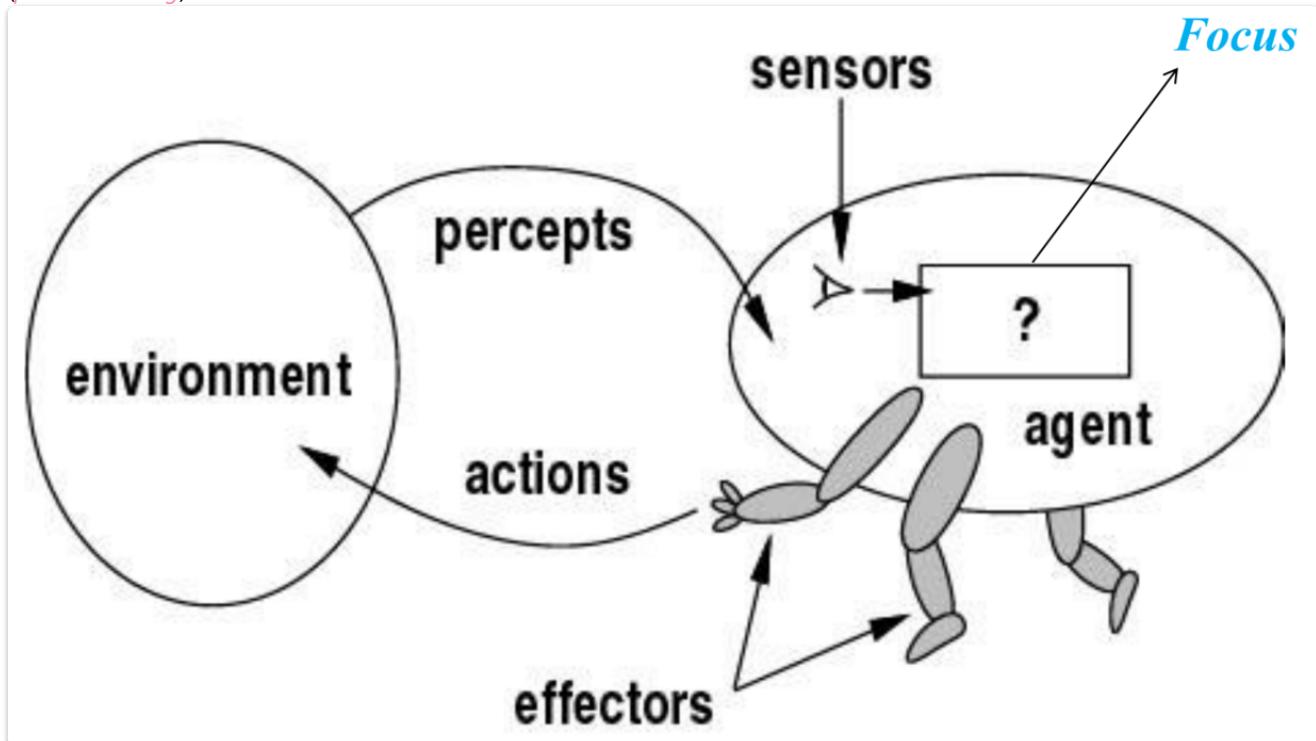


L'intelligenza artificiale non è una collezione di tecniche per risolvere problemi specifici. Ma vertice e fronte del progresso dei metodi / sistemi informatici per fornire metodologie sistematiche per dotare le macchine di comportamenti *intelligenti / razionali*.

Agenti intelligenti

L'approccio moderno all'IA consiste nella costruzione di **agenti intelligenti**, questo approccio ci offre un quadro di riferimento ed una prospettiva diversa dall'analisi dei sistemi software. È comoda per trattare sistemi razionali (*uniformità*) e vedremo schemi di agenti contenitori di funzionalità.

Il primo obiettivo è quello di trovare agenti per la risoluzione di problemi vista come una *ricerca* in uno *spazio di stati* (*problem solving*).



Caratteristiche degli agenti

- Sono **situati**: ricevono percezioni da un ambiente ed agiscono su questo mediante azioni (*attuatori*)
- Hanno **abilità sociale**: sono capaci di comunicare, collaborare e difendersi da altri agenti
- Hanno **credenze, obiettivi, intenzioni**...
- Sono **embodied**: hanno un corpo, fino a considerare i meccanismi delle emozioni

Percezioni ed azioni

☰ Percezione ▾

Input da **sensori**

☰ Sequenza percettiva

Storia completa *delle percezioni*

La scelta dell'azione è funzione unicamente della sequenza percettiva, ma non da qualcosa che non abbia percepito.

99 Funzione agente

Viene implementata da un *programma agente* e definisce l'azione da compiere per ogni sequenza percettiva.
Describe completamente l'agente

$$\text{Sequenza Percettiva} \xrightarrow{f} \text{Azione}$$

Può essere descritta in modo matematico (associa gli elementi di un dominio a un elemento di un'immagine) ma anche in maniera algoritmica, logica o come una funzione appresa dal machine learning

Esempi di architetture astratte di agenti

Agenti razionali

Interagisce con il suo ambiente in maniera *efficace* (fa la cosa "*giusta*"). Serve un criterio di *valutazione* oggettivo dell'effetto delle azioni dell'agente.

:≡ Example

Costo minimo di un cammino soluzione ([Esempio del viaggio in romania](#))

Valutazione della prestazione

Misura di prestazione

- esterna (come vogliamo che il mondo si evolva?)
- scelta del progettista a seconda del problema considerando l'effetto desiderato sull'ambiente
- valutazione su ambienti diversi

Definizione

La razionalità è relativa a (dipende da):

- le misure di prestazione
- le conoscenze progresse dell'ambiente
- le percezioni presenti e passate (*sequenza percettiva*)
- le capacità dell'agente (*azioni possibili*)

99 Agente razionale

L'agente che per ogni sequenza di percezioni compie l'azione che **massimizza il valore atteso della misura delle prestazioni**, considerando le sue percezioni passate e la sua conoscenza progettuale.

Cosa non è

RAZIONALITÀ NON ONNISCIENZA

Non si pretendono perfezione e conoscenza del futuro, ma massimizzare il risultato atteso.

La scelta dell'azione deve essere fatta solo in funzione della sequenza percettiva, quindi vogliamo un agente che massimizzi quello che fa in funzione di quello che ha effettivamente percepito, non qualcosa che non ha mai visto né conosciuto.

Ma potrebbero essere necessarie azioni di acquisizione di informazione o esplorative.

Ciò non toglie che l'agente possa esplorare l'ambiente e cercare di apprendere più informazioni, ma non è escluso che in alcuni casi l'agente possa non riuscire a trovare la soluzione, vogliamo infatti massimizzare il risultato **atteso** (in probabilità).

RAZIONALITÀ NON ONNIPOTENZA

Le capacità dell'agente possono essere limitate.

Non è detto che possa agire sull'ambiente così da ottenere quello che desidera (ad esempio: potrebbe non avere la potenza di calcolo per trovare la soluzione in tempo ottimale).

Razionalità e apprendimento

Raramente tutta la conoscenza sull'ambiente può essere fornita a priori (dal programmatore). L'agente razionale deve essere in grado di modificare il proprio comportamento apprendendo informazioni con l'*esperienza* (le percezioni passate) in ambienti diversi e/o mutevoli. Può migliorare esplorando, apprendendo, aumentando autonomia per operare in ambienti differenti o mutevoli.

Agenti autonomi

Agente autonomo

Un agente è *autonomo* nella misura in cui il suo comportamento dipende dalla sua capacità di ottenere esperienza, e non dall'aiuto del progettista.

Un agente il cui comportamento fosse determinato solo dalla sua conoscenza *built-in* (pregressa) sarebbe non autonomo e poco flessibile.

Struttura di un Agente

Un agente opera in un *Ambiente*

$$\text{Agente} = \text{Architettura} + \text{Programma}$$

$$Ag : P \rightarrow Az$$

percezioni *azioni*

Il *programma dell'agente* implementa la funzione *Ag*.

Programma agente

```
function Skeleton-Agent (percept) returns action
    static: memory      //the agent's memory of the world
    memory ← UpdateMemory(memory, percept)
    action ← Choose-Best-Action(memory)
    memory ← UpdateMemory(memory, action)
    return action
```

Nel meta algoritmo ho una memoria, ogni volta che ho una *percezione* aggiorno la memoria, quando devo scegliere un'azione scelgo la migliore della memoria.

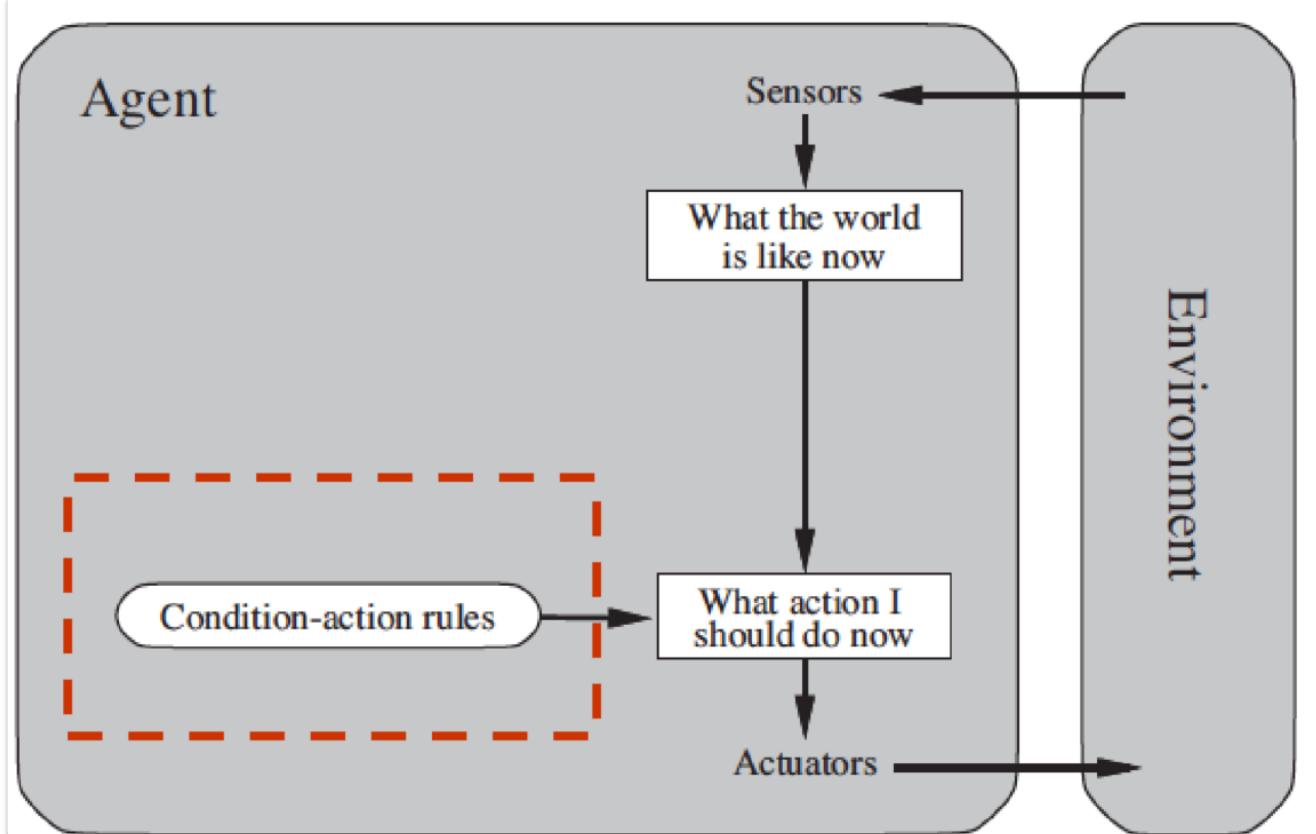
Agente basato su tabella

La scelta dell'azione è un accesso a una *tabella* che associa un'azione ad ogni possibile sequenza di percezioni.

Problemi:

- Dimensione: per giocare a scacchi serve una tabella con un numero di righe $>> 10^{80} \Rightarrow$ ingestibile
- Difficile da costruire
- Nessuna autonomia
- Di difficile aggiornamento, apprendimento complesso
Noi vogliamo realizzare agenti razionali con programma compatto.

Agenti reattivi semplici



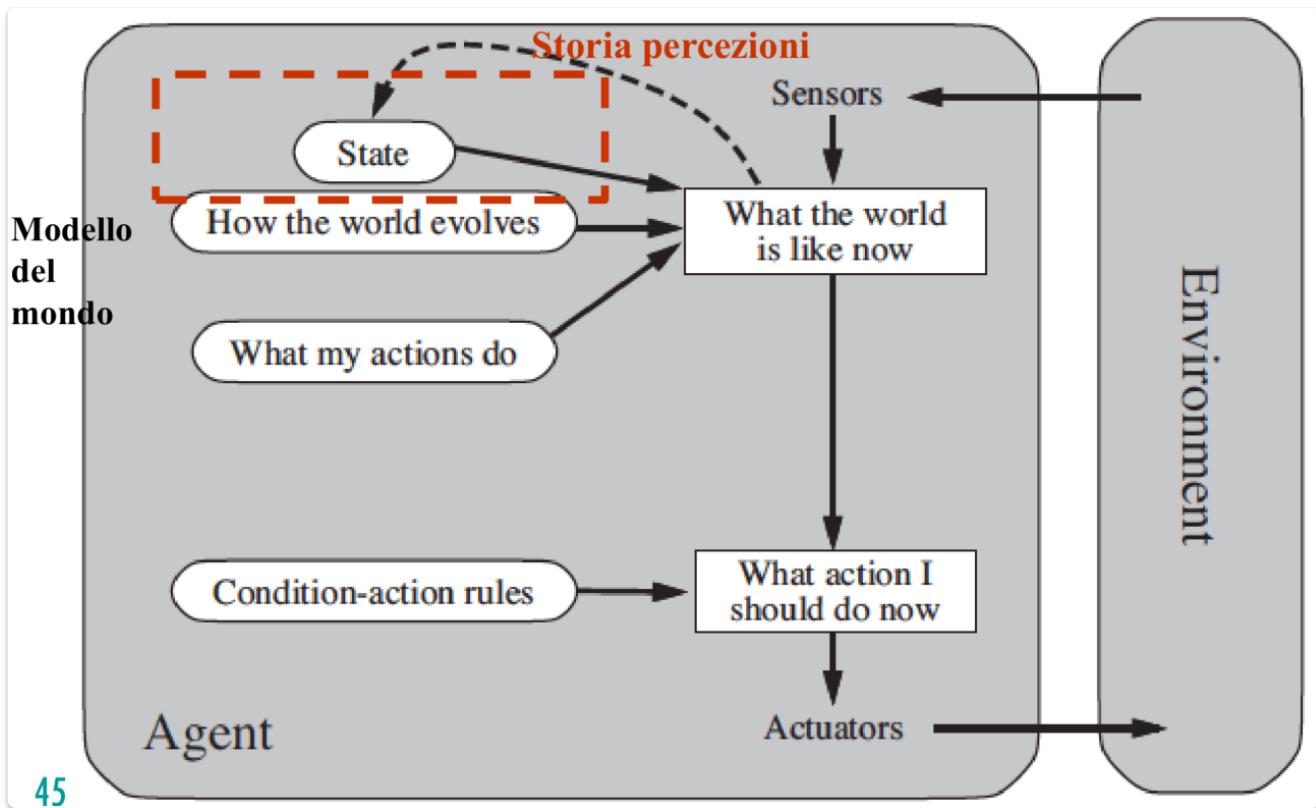
```

function Agente-Reattivo-Semplice (percezione) returns azione
    persistent: regole      // insieme di regole
    condizione azione      // if - then
        stato <- Interpreta-Input (percezione)
        regola <- Regola-Corrispondente (stato, regole)
        azione <- regola.Azione
    return azione

```

Non c'è storia da memorizzare, si reagisce solo alle situazioni attuali. Assume che l'ambiente sia completamente osservabile.

Agenti basati su modello



45

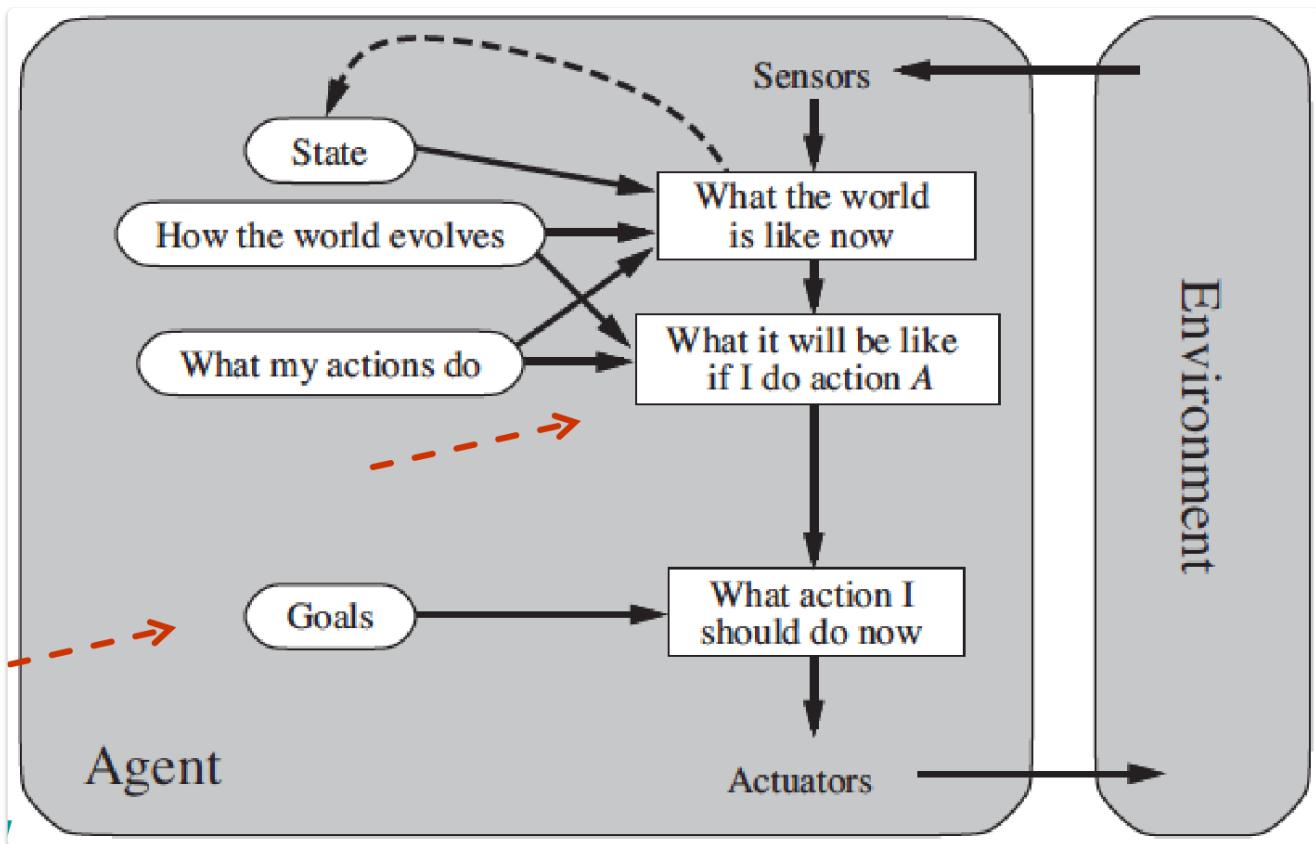
```

function Agente-Basato-Su-Modello (percezione) returns azione
    persistent: stato,      // descrizione stato corrente
                modello, // conoscenza del mondo
                regole,   // insieme di regole condizione-azione
                azione    // azione più recente
    stato ← Aggiorna-Estado (stato, azione, percezione, modello)
    regola ← Regola-Corrispondente (stato, regole)
    azione ← regola.azione
    return azione

```

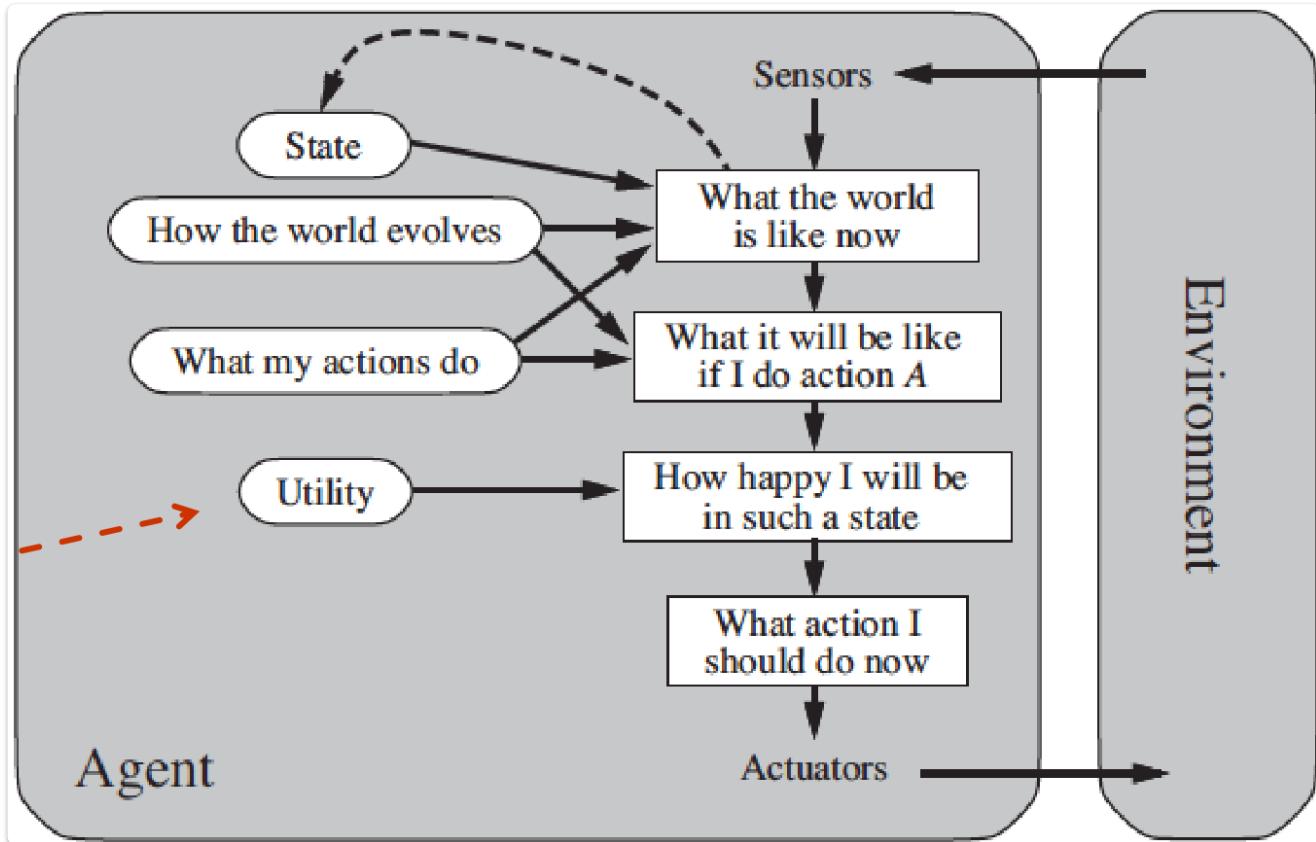
Rispetto all'agente reattivo semplice, abbiamo in più *conoscenza sulle azioni del mondo*. Lo stato è un'aggiornamento di stato, azione, percezione e modello.

Agenti con obiettivo



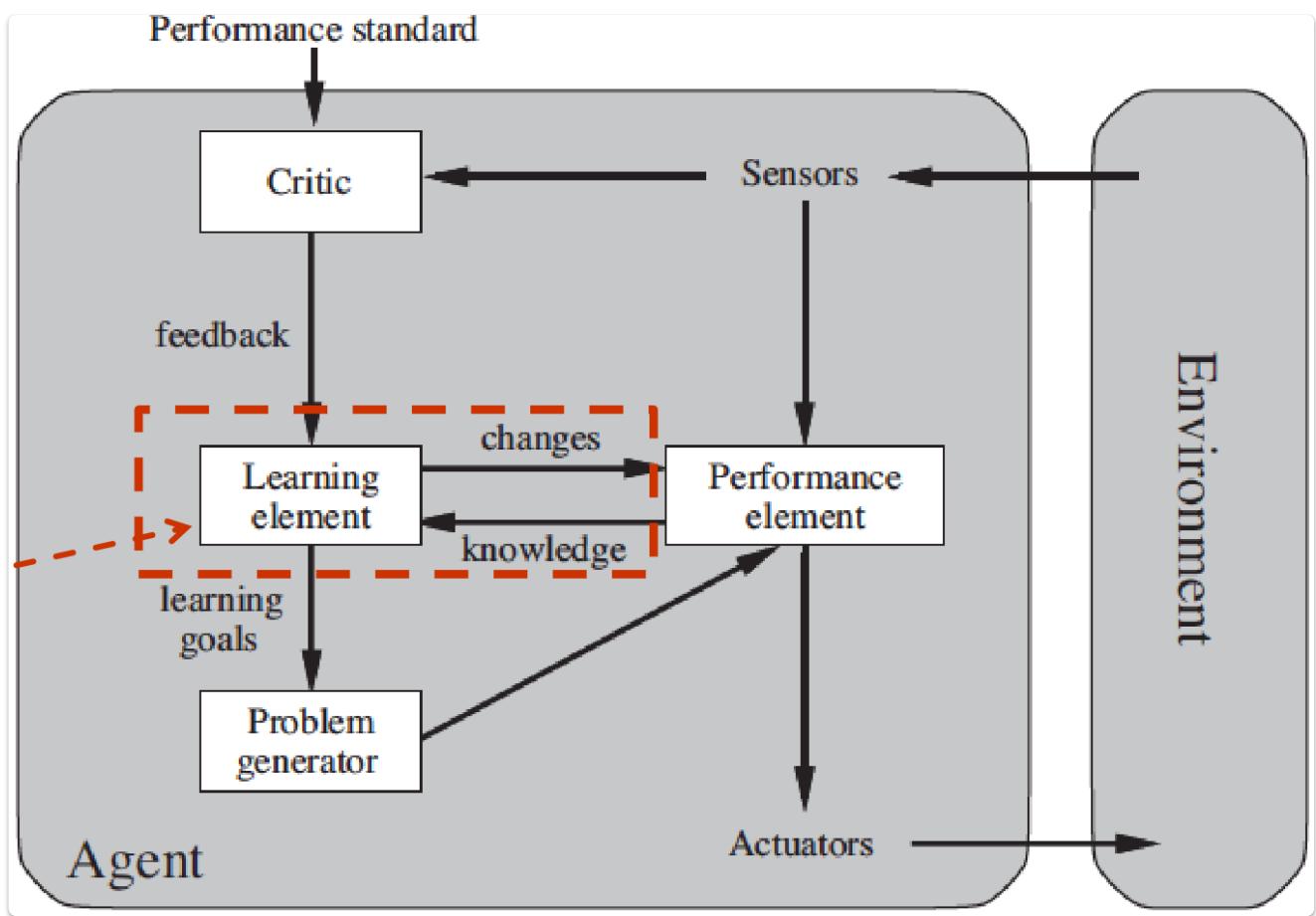
Sono guidati da un *obiettivo nella scelta dell'azione*, è fornito un goal esplicito (es. città da raggiungere). A volte l'azione migliore dipende da qual è l'obiettivo da raggiungere, devono pianificare una *sequenza di azioni* per raggiungerlo. Sono meno efficienti ma *più flessibili* di un agente reattivo, dato che l'obiettivo può cambiare non essendo codificato nelle regole.

Agenti con valutazione di utilità



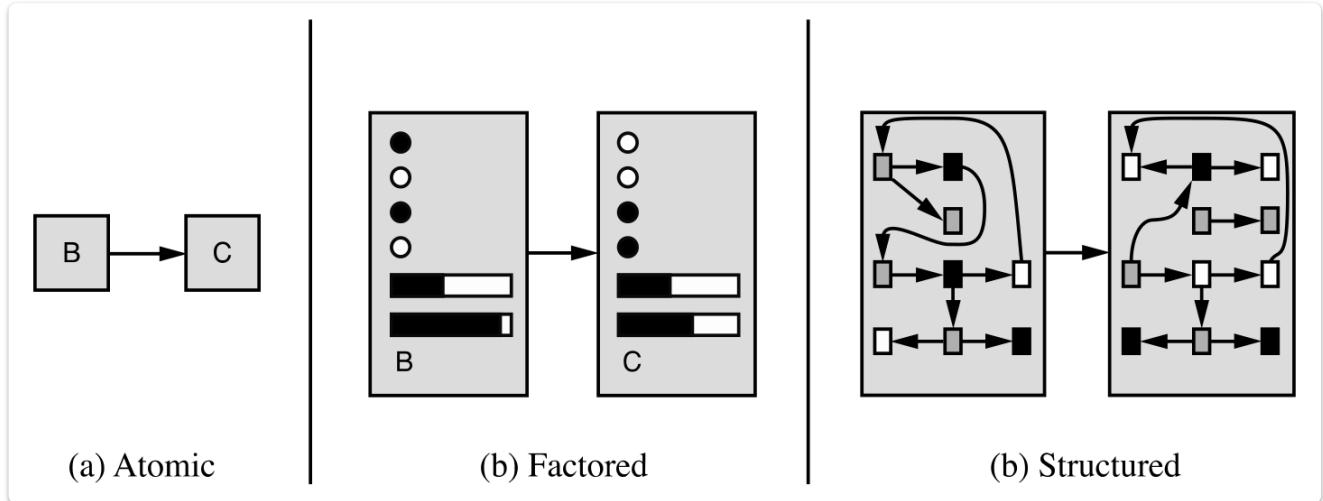
Possono esserci *obiettivi alternativi* (o più modi per raggiungerlo): l'agente deve decidere verso quali di questi muoversi, è necessaria una *funzione utilità* che associa ad uno stato obiettivo un numero reale. Ci sono alcuni obiettivi più facilmente raggiungibili di altri, la funzione utilità tiene conto anche della *probabilità di successo*.

Agenti che apprendono



1. **Componente di apprendimento:** produce cambiamenti al programma agente, migliora le prestazioni adattando i suoi componenti, apprendendo dall'ambiente
2. **Elemento esecutivo (performance element):** Il programma agente (visto sinora per decidere le azioni)
3. **Elemento critico:** osserva e da feedback sul comportamento
4. **Generatore di problemi:** suggerisce nuove situazioni da esplorare

Tipi di rappresentazione



- rappresentazione atomica: solo il passaggio da uno stato all'altro.
- rappresentazione fattorizzata: nello stato ci sono più variabili e attributi che assumono valore diverso
- strutturata: aggiunge relazione fra gli stati