



# Architetture per Agenti

**S. Costantini**

Ho ottenuto questo materiale (che uso per scopi accademici) assemblando e modificando materiale trovato sul Web. Ringrazio tutti i colleghi che hanno inserito il loro materiale online e hanno contribuito alla diffusione della conoscenza. Il materiale risultante è, in ogni caso, libero e gratuito per l'uso accademico.

# Sommario

- 
- 
- 
- 
- 
- 

## Introduzione

## Architetture per Agenti

- agenti con ragionamento deduttivo
- agenti con ragionamento pratico
- agenti reattivi
- agenti Ibridi

# Da dove vengono gli agenti?

- Lavoro Seminale sugli agenti: Modello attore di Carl Hewitt (1977)
  - *“un oggetto auto-contenuto, interattivo e ad esecuzione concorrente, con alcuni stati interni encapsulati e che può rispondere a messaggi da parte di altri oggetti simili”*

# Cos'è un Agente?

- Dizionario di Webster:  
*“uno che è autorizzato ad agire al posto di un altro”*
- Gli agenti software sono programmi che sono capaci di agire per conto di un utente o di un altro agente per eseguire compiti specifici e/o raggiungere gli obiettivi che gli sono stati dati.

# Agenti e MAS

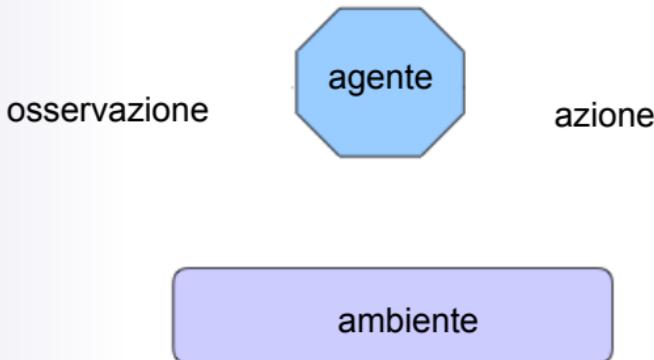
- Gli agenti e i sistemi multi-agente (MAS) sono emersi come una tecnologia molto potente per affrontare la complessità di una varietà di scenari dove è richiesta una certa autonomia.
- Esistono ora alcune applicazioni industriali che dimostrano il vantaggio di usare gli agenti.
- Ad ogni modo, I sistemi di agenti devono ancora acquisire una distribuzione ampia negli ambienti operativi, in quanto la tecnologia deve ancora muoversi dalla pura ricerca allo sviluppo.

# Agenti

- Sistemi che possono (devono) decidere da soli di cosa hanno bisogno in modo da soddisfare i loro obiettivi di costruzione.
- Si trovano in alcuni ambienti
  - Hanno controllo parziale dell'ambiente
  - Capaci di azioni autonome
- Primi esempi:
  - Sistemi di Controllo
  - Demoni Software

# Agenti: una definizione famosa

- M. Wooldridge
  - Un agente è
    - Un sistema computer ...
    - ... che si trova in un certo ambiente, ...
    - ... capace di azioni autonome in questo ambiente ...
    - ... in modo da raggiungere i suoi obiettivi di costruzione.



# Paradigma Agent-Oriented

- Le piattaforme per costruire software autonomo richiedono concetti e linguaggi di base dedicati.
- A questo livello di agenti individuali, abbiamo bisogno di rappresentare come oggetti di prima classe:
  - osservazioni
  - eventi
  - azioni
  - credenze
  - obiettivi
  - ...

# Paradigma Agent-Oriented

- Le funzionalità che ogni framework Agent-Oriented deve fornire sono almeno:
  - reattività: è l'abilità di un agente di percepire il suo ambiente esterno e prendere misure appropriate in risposta alle percezioni.
  - proattività: è l'abilità di un agente di prendere iniziative basate sulla sua stessa valutazione di condizioni rilevanti.
  - abilità sociale: include l'abilità di un agente di comunicare con gli altri agenti con modalità appropriate.

# Agenti Intelligenti

- Lungo il cammino dello sviluppo di software autonomi, le nuove applicazioni hanno bisogno di “intelligenza”, nel senso dell’abilità:
  - di esibire, comporre e adattare comportamenti
  - di essere in grado di imparare la strada appropriata (migliore) per eseguire un compito invece di chiedere istruzioni su come eseguirlo.

# Perchè abbiamo bisogno della tecnologia degli agenti intelligenti? – Alcuni esempi:

- esplosione di informazioni e sovraccarico dovuto alla popolarità di internet
  - fornitura di informazioni (fornitori)
  - domanda di informazioni (consumatori)
- assistenza personale di accesso e utilizzo delle informazioni
- miglioramento dell'efficienza della comunicazione e dell'interazione tra le persone
- costruzione di comunità e società virtuali basate sugli agenti

# Cosa possono fare gli agenti intelligenti? – Alcuni esempi

- raccolta e ricerca di dati in internet
- notizie elettroniche e filtraggio mail
- amministrazione del calendario e organizzazione degli incontri
- assistente nei work-flow
- organizzazione dei viaggi
- monitorizzazione degli eventi
  - allertando gli utenti su opportunità di investimento
  - allertando dottori su eventi di emergenza su pazienti

# Da Agenti a Sistemi Multi-Agente

- Un sistema multi-agente (MAS) è una collezione di agenti software che funziona in congiunzione con ogni altro.
  - Cooperazione, collaborazione e competimento
  - Obiettivi/intenzioni comuni
  - Piani comuni
  - Comportamenti di Squadra
  - Formazione di Coalizioni
- A giudicare dalle potenziali applicazioni, sistemi distribuiti di controllo/monitoraggio (DCMS) sembrano essere un reame naturale per agenti, per la virtù dei controlli di essere principalmente entità autonome.

# Intelligenza d'Ambiente

- Uno scenario di “Intelligenza d’Ambiente” può essere visto come un controllo distribuito e un sistema di monitoraggio.
- L’ “oggetto” delle attività di monitoraggio degli agenti sono gli utenti e l’ambiente dove si trovano.
- Il grado di controllo qui è vago, in quanto un utente può essere avvisato di fare qualcosa, ma in ogni caso mantiene la sua autonomia.
- Il grado di controllo deve aumentare se, per esempio, un sistema è creato per l’assistenza di persone disabili.

# Proprietà dell'agente: sommario

- proprietà dell'agente

- |            |  |
|------------|--|
| essenziale | <ul style="list-style-type: none"><li>• reattività<ul style="list-style-type: none"><li>reagisce agli stimoli (cambi nell'ambiente, reazione a comunicazioni, ...)</li></ul></li><li>• autonomia<ul style="list-style-type: none"><li>non richiede controllo utente</li></ul></li><li>• pro-attività<ul style="list-style-type: none"><li>cerca di raggiungere il suo scopo eseguendo le azioni appropriate</li></ul></li><li>• abilità sociale<ul style="list-style-type: none"><li>coopera / coordina / comunica / ...</li></ul></li><li>• incorporato<ul style="list-style-type: none"><li>si trova nell'ambiente</li></ul></li></ul> |
| extra      | <ul style="list-style-type: none"><li>• mobile<ul style="list-style-type: none"><li>si muove intorno alla rete</li></ul></li><li>• impara<ul style="list-style-type: none"><li>impara dalle esperienze passate</li></ul></li><li>• ...</li></ul>   |

# Agenti contro Oggetti

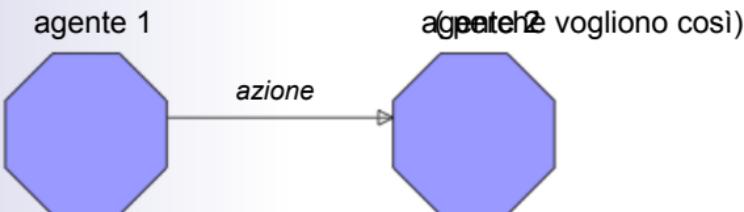
- Il paradigma ben conosciuto della Programmazione Orientata agli oggetti è ampiamente usato per una varietà di applicazioni.
- Il nuovo paradigma della programmazione orientata agli Agenti sta emergendo.
- E' utile conoscere similitudini e differenze, anche per capire se questo paradigma è davvero utile.

# Agenti contro Oggetti

- Oggetti (Java / C++ / C# / Smalltalk / Eiffel / ...)
  - encapsulano
    - stato “attributi” / “membri dei dati” / ...
    - comportamento “operazioni” / “metodi” / ...
  - rappresentano entità del mondo reale

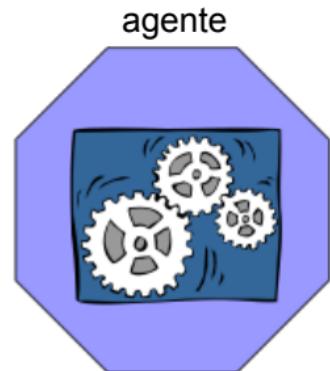
# Agenti contro Oggetti: differenze

- autonomia: chi decide l'esecuzione di una "azione" particolare
  - gli oggetti hanno controllo sullo stato (attraverso le operazioni)  
gli oggetti non hanno controllo sui loro comportamenti
    - un oggetto può chiamare qualsiasi operazione pubblica sull'oggetto
    - un oggetto non può decidere quando eseguire il suo comportamento
  - gli agenti richiedono un azione da un altro agente [...]
    - il controllo passa totalmente all'agente ricevente
- "*gli oggetti lo fanno gratis, gli agenti per denaro*"



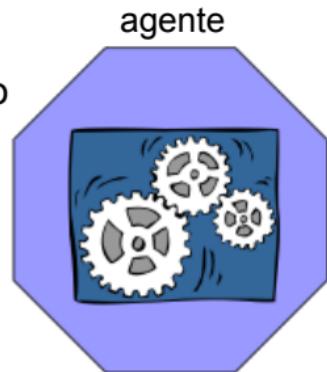
# Agenti contro Oggetti: differenze (cont.)

- considera l'architettura del comportamento, obiettivo:  
*integrazione di comportamenti flessibili autonomi*
  - oggetti
    - operazioni per offrire comportamenti
  - agenti
    - integrano
      - comportamenti reattivi
      - comportamenti sociali
      - comportamenti proattivi
    - ...



# Agenti contro Oggetti: differenze (cont.)

- Singolo- o Multi-Processo
  - oggetti
    - non separano processi di controllo
    - eccetto per “oggetti attivi”
  - agenti
    - concettualmente processi diversi di controllo



# Agenti e Oggetti: sommario

- Gli oggetti sono entità computazionali che encapsulano degli stati e sono in grado di eseguire alcune azioni (metodi) su questo stato e comunicare passando messaggi.
- Differenze tra agenti e oggetti:
  - Gli agenti non invocano metodi su un altro, ma invece richiedono azioni per essere eseguiti.
  - Gli oggetti non hanno comportamenti autonomi flessibili.
  - Ogni agente ha il suo processo di controllo.
- La Programmazione Orientata agli Oggetti può essere usata per implementare agenti, con alcune modifiche.

# Agenti e Oggetti: sommario

	Oggetto	Agente
Stati di definizione parametri	Nessuna Restrizione	Credenze, impegno nelle scelte, capacità,
Processo di computatione	Passaggio di Messaggi e Metodi di Responso	Passaggio di Messaggi e Metodi di Responso
Tipi di messaggio	Nessuna Restrizione	Informa, richiede, offre, ecc.
Restrizioni sui metodi	Nessuna	Onestà, consistenza, ecc.

# Agenti contro Sistemi Esperti



- Sistemi Esperti, es. MYCIN
  - agiscono come consultatori computerizzati
  - per dottori e altri esperti umani
  - Esempio: MYCIN ha conoscenze di malattie del sangue in umani
    - il sistema ha molte conoscenze riguardo le malattie del sangue, nella forma di regole
    - un dottore può ottenere consigli esperti riguardo le malattie del sangue dando a MYCIN fatti e creando delle query

# Agenti contro Sistemi Esperti

- Differenze: Sistemi Esperti
  - sono **scorporati**
    - non operano in un ambiente
  - non hanno **comportamento attivo/proattivo**
    - controllato dall'utente
  - non hanno **comportamento sociale**
    - come cooperazione / coordinazione / negoziazione / ...



# Agenti: Un paradigma di costruzione del sistema



$$\frac{\text{agenti}}{1998} = \frac{\text{oggetti}}{1982} = \frac{\text{Programmazione Strutturata}}{1974}$$

# Architetture per Agente

- Come può qualcuno creare un nuovo approccio/formalismo/linguaggio Agent-Oriented?
- Dato un certo linguaggio, come può qualcuno creare e implementare un agente o un MAS?
- E' richiesto un modello sottostante dei "nostri stessi" agenti, che determina:
  - Metodologie di creazione che indicano come la creazione di un agente deve essere divisa in componenti, e come questi componenti interagiscono
  - Metodologie di programmazione
  - Linee guida implementative per l'infrastruttura sottostante

# Architetture per Agente

- Abbiamo bisogno di *Architetture Agente* per fornire una visione astratta degli approcci Agent-Oriented.
- Differenti architetture riflettono differenti punti di vista su cosa è un agente.
- Inizieremo da architetture molto astratte che descrivono agenti ad un livello molto alto, dando nessun dettaglio.

# Architetture per Agente

- Poi, illustreremo architetture *concrete* che specificano in dettaglio, per il loro approccio particolare ad agenti:
  - quale tipo di compiti dovrebbero fronteggiare
  - quali sono i componenti di un agente
  - come opera ogni componente
  - come i componenti interagiscono
  - come gli agenti differenti interagiscono (se gli agenti devo comporre MAS)



# Architetture per Agente

- Differenti architetture riflettono anche differenti punto di vista su
  - come un agente deve comportarsi (in pratica)
  - quali sono le semantiche formali e operazionali
- Per ogni architettura, ci sono diverse implementazioni (linguaggi e/o sistemi) che incorporano i principi sottostanti quell'architettura

# Architetture per Agente: come fare la cosa giusta ?

- Pattie Maes [1991]
  - '[Una] particolare metodologia per costruire [agenti]. Specifica come . . . l'agente può essere **decomposto** nella costruzione di un insieme di **moduli** componenti e come questi moduli dovrebbero essere fatti per **interagire**. L'insieme totale di moduli e la loro interazione deve fornire una risposta alla domanda di **come** i dati dei sensori e lo stato interno corrente dell'agente **determina l'azione** . . . e il futuro stato interno dell'agente.'
- Abbiamo bisogno di modelli che un progettista può adottare per definire linguaggi agenti e a cui un programmatore può riferirsi per implementare agenti

# Architetture Astratte per Agenti

- Architetture Agente: vista generale di un agente che identifica i suoi componenti e i suoi comportamenti operazionali.
- L'architettura più astratta:
  - elementi
    - $E$  insieme di stati ambiente
    - $Ac$  insieme di possibili azioni agente
  - $Ag: E^* \rightarrow Ac$  (mappatura)
  - In risposta a (la sequenza di) stimoli ambientali, l'agente esegue azioni.

# Architetture Astratte per Agenti

- agente puramente reattivo

**Agente:** E  Ac

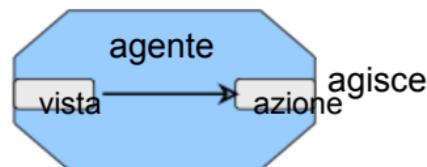
- mappatura diretta semplice
- nessuna storia

- percezione: uno stato d'ambiente E diventa una percezione Per per conto di “sensori”

**vista:** E  Per

**azione:** Per\*  Ac

osserva



# Architetture Astratte per Agenti

- agente puramente reattivo

**Agente:**  $E \sqsubseteq Ac$

- mappatura diretta semplice
- nessuna storia

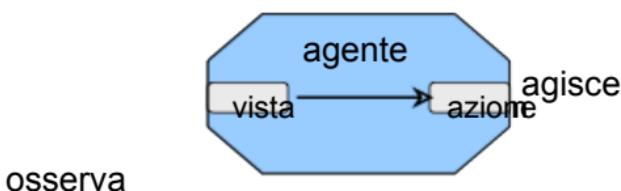
- manca un punto: come fa l'agente a conoscere lo stato dell'ambiente? Deve avere un modo per osservare/percepire l'ambiente

# Architetture Astratte per Agenti

- Agente reattivo in più dettagli: percezioni
- Percezione: uno stato ambientale E diventa una percezione Per grazie a dei “sensori”. L’ insieme Per\* degli stati dell’ambiente percepito determinano azioni.

**vista:** E  Per

**azione:** Per\*  Ac



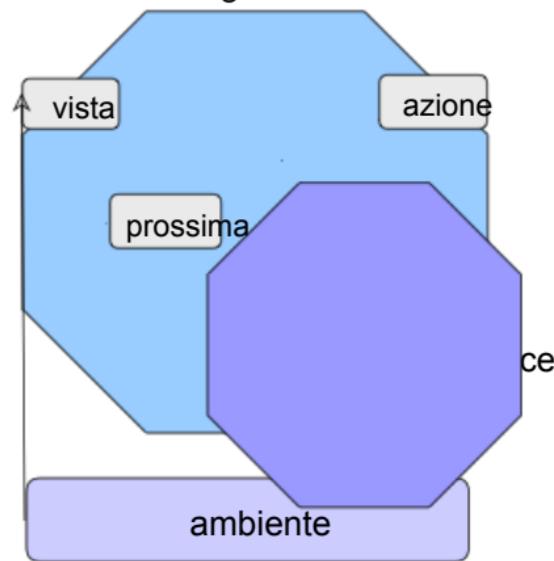
ambiente

# Architetture Astratte per Agenti: Agenti con stato

- Da uno stato iniziale interno, le nuove percezioni determinano nuovi stati interni. Ad ogni momento, nuove percezioni determinano nuovi stati interni. Ad ogni momento, lo stato interno corrente determina l'azione eseguita

I stato interno agente

vista | E  Per  
azione |  Ac  
prossima | x Per  osserva



# Architetture concrete per agenti

- Agenti con ragionamento deduttivo
  - 1956 – presente
  - “Gli agenti compiono decisioni riguardo cosa fare attraverso la manipolazione di simboli. La sua espressione più pura propone che l’agente usi esplicito ragionamento logico in modo da decidere cosa fare”
- Agenti con ragionamenti pratico
  - 1990 – presente
  - “Gli Agenti usano ragionamento pratico (verso le azioni, non verso i comportamenti) – credenze / desideri / intenzioni.”

# Architetture concrete per agenti

## Agenti reattivi

1985 – presente

“Problemi con ragionamento simbolico portano ad una reazione contro questa — spinta verso il movimento degli **agenti reattivi**”

## Agenti ibridi

1989 – presente

“*Architetture ibride provano a **combinare** il meglio delle architetture reattive e di ragionamento*”.

# 1. Agenti con ragionamento deduttivo

- architetture basate sulle idee dell' "IA simbolica"
  - rappresentazione simbolica
    - ambiente
    - comportamenti
    - obiettivi
    - ...
  - rappresentazione: formule logiche
  - manipolazione sintattica: deduzione logica / dimostrazione di teoremi

Risultato: "agente deliberativo"

# Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi

- agenti deliberativi
  - database di credenze specificate usando formule di predicato logico di prim'ordine
    - es. is\_open (door1)  
is\_closed (door2)  
visible (wall32)  
...
    - un insieme di azioni, es. open(Door)
  - insieme di regole di deduzione

## Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi (cont.)

- Un agente con ragionamento deduttivo punta ad eseguire azioni:
  - l'agente ha un insieme di azioni che deve eseguire
  - ogni azione può essere eseguita *solo* se alcune **precondizioni** possono essere inferite come vere
  - le precondizioni sono formule logiche
  - esempio di precondizione: in modo da aprire la porta (azione) lo devo avere la chiave (precondizione)

## Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi (cont.)

- Le regole di deduzione operano sul database delle credenze per inferire:
  - quale azione può essere eseguita, quali sono le azioni con le precondizioni che sono verificate (la loro verità può essere derivata dalle credenze correnti)
  - quale azione può essere *possibilmente* eseguita, in quanto non è dimostrabile come falsa (la sua falsità non può essere derivata dalle credenze correnti)

# Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi (cont.)

- Ciclo base dell'agente

**per ogni** azione a  
**se** la-precondizione-per-azione a può essere inferita dalle credenze correnti  
**restituisci** a

**per ogni** azione a  
**se** la-precondizione-per-azione a non è esclusa dalle credenze correnti  
**restituisci** a

**restituisci null**

- Alcune azioni possono essere fattibili ad ogni ciclo:  
l'agente ha bisogno di una *funzione di selezione* che seleziona quella da eseguire

# Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi (cont.)

- **“razionalità calcolativa”**
  - “Inferito” significa “provato dalle credenze ricevute attraverso le regole di inferenza ricevute”
  - “Non escluso” significa “negazione non provata dalle credenze ricevute attraverso le regole di inferenza ricevute”
- **non accettabile** in ambienti che **cambiano** più velocemente rispetto al tempo di decisione dell’agente

## **Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi (cont.)**

- Vantaggi
  - semantica logica pulita
  - espressivo
  - dominio della logica ben ricercato

# Agenti con ragionamento deduttivo: Agenti come dimostratori di teoremi (cont.)

- Problemi
  - come costruire rappresentazioni interne dalle percezioni
    - es. immagine  formule logiche
  - complessità computazionale inerente alla dimostrazione di teoremi
    - funzioni temporali !
  - molti (la maggiorparte) algoritmi di manipolazione simboli basati sulla ricerca sono *altamente intrattabili*

# Architetture concrete per agenti

## Agenti a Ragionamento Deduttivo

1956 – presente

"Gli agenti prendono decisioni su cosa devono fare attraverso la manipolazione dei simboli. La sua espressione più pura propone che l'agente usa il ragionamento logico esplicito in modo da decidere cosa fare."

## Agenti a Ragionamento Pratico

1990 – presente

"Gli agenti usano ragionamenti pratico (attraverso le azioni, non attraverso le credenze) – credenze / desideri / intenzioni."

## Agenti Reattivi

1985 – presente

"Problemi con il ragionamento simbolico hanno portato alla creazione degli agenti reattivi"

## Agenti Ibridi

1989 – presente

"Le Architetture Ibride provano a combinare il meglio delle architetture a ragionamento e di quelle a reattività"

## 2. Agenti a Ragionamento Pratico

- Cos'è il ragionamento pratico?
  - “**ragionamento diretto verso le azioni**”
  - distingue il ragionamento pratico dal *ragionamento teorico*:
    - il ragionamento teorico è diretto verso le credenze
    - il ragionamento pratico è diretto verso le azioni

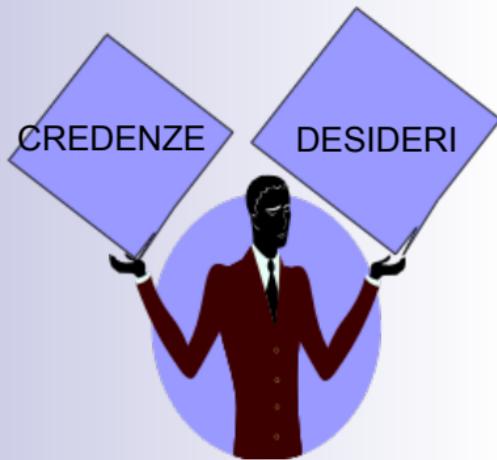
# Architetture BDI

- *BDI = Belief, Desires, Intentions (Credenze, Desideri, Intenzioni)*
- *BDI – teoria del ragionamento pratico* (Bratman, 1988) per “agenti legati alle risorse”
- include
  - analisi del significato
  - pesatura delle alternative competenti
  - interazioni tra queste due forme di ragionamento
- Concetti Fondamentali
  - Credenze = informazioni che l'agente ha del mondo
  - Desideri = stato degli affari che l'agente vorrebbe sbrigare
  - Intenzioni = desideri (o azioni) che l'agente deve compiere

# Agenti BDI

- **Credenze:**
  - rappresenta le caratteristiche dell'ambiente
  - sono aggiornate appropriatamente dopo ogni azione di sensazione
  - possono essere viste come il componente ***informativo*** del sistema
- **Desideri**
  - contengono l'informazione sugli obiettivi da raggiungere, le priorità e i premi associati con i vari obiettivi
  - possono essere pensati come rappresentanti dello stato ***motivazionale*** del sistema.

# Agenti BDI



FUNZIONE DI  
SELEZIONE



INTENZIONE



# Agenti BDI

La **funzione di selezione** deve abilitare il sistema a raggiungere i suoi obiettivi, date:

- le risorse computazionali disponibili nel sistema
- le caratteristiche dell'ambiente in cui il sistema è situato
- le preferenze dell'agente, le funzioni utilità, ecc

## • Intenzioni

- rappresentano il corso correntemente scelto di azioni (l'output della più recente chiamata alla funzione selezionata)
- catturano il componente *deliberativo* del sistema

# Credenze, desideri e intenzioni: un trucco sulla semantica

- Credenze, Desideri e Intenzioni sono considerati essere “stati mentali” di un agente
- La logica modale è usata per modellare questi stati mentali
  - **Credenze: operatore modale Bel (o B in breve)**
  - **Desideri: operatore modale Des (o D in breve)**
  - **Intenzioni: operatore modale Int (o I in breve)**

# Credenze, desideri e intenzioni: un trucco sulla semantica

- Credenze, Desideri e Intenzioni sono considerati essere “stati mentali” di un agente
- Semantica: possibili mondi (Kripka, Intikke). Intuitivamente (ma non troppo precisamente):
  - i possibili mondi in questo contesto sono state alternative “plausibili” della conoscenza dell’agente.
  - un possibile mondo è raggiungibile o accessibile dallo stato presente se l’agente è capace di trasformare il mondo presente in uno nuovo applicando le sue regole di inferenza

# Credenze

- **X Bel b** quando X è un agente e b è una proposizione se b si trova in ogni possibile mondo che l'agente può raggiungere da quello presente
  - Osservazione: Razionalità Limitata / Risorse computazionali limitate vogliono dire che l'agente non può derivare tutto quello che in principio dovrebbe “credere”

# Desideri

- **X Des d se d si trova in qualche mondo possibile che l'agente vorrebbe raggiungere**
- L'agente potrebbe non sapere come raggiungere lo stato in cui vorrebbe essere
- Un agente potrebbe desiderare di stare in stato conflittuale
- **Gli obiettivi sono nel sottoinsieme dei desideri dell'agente che sono raggiungibili e consistenti**

**NON HA ONNISCENZA LOGICA:** Non desidera solo stati che può raggiungere

# Intenzioni

- X Int p se p si trova lungo tutti i cammini raggiungibili dall'agente quando sta consistentemente provando ad inseguire l'intenzione stessa (cammini "intesi")
- Un agente può anche avere un'intenzione **insoddisfacibile** (se l'insieme di cammini "intesi" è vuoto)
- Un agente può intendere qualcosa, e ancora fallire a farlo diventare vero (se procede in un cammino che non è nell'insieme di cammini "intesi")
- In molte descrizioni informali ci sono ambiguità: alcune volte le intenzioni sono interpretate come *obiettivi* e alcune volte come *piani*. Formalmente, un'intenzione è un **desiderio fattibile** (es. un desidero soddisfacibile in pratica)

# Responsabilità (Commitment)

- Gli agenti che persistono nelle loro intenzioni (finchè sono soddisfacibili) si dicono essere **responsabili** di quelle intenzioni
- La Responsabilità è un'importante proprietà che, se implementata, crea uno scenario più sicuro per gli altri agenti, incluso l'utente: infatti, questi diventano capaci di prevedere che un agente non cambi arbitrariamente il suo modo di comportarsi.

## **Svantaggi della semantica modale**

- Prove della procedura sono indecidibili
- Anche in casi decidibili le procedure sono molto pesanti a livello di computazione
- I sistemi reali “approssimano” la semantica modale invece di implementarla realmente

# Perché studiamo BDI

BDI è particolarmente interessante per:

- componenti filosofiche - basate su una teoria di azioni razionali negli umani
- architettura software – è stato implementato e successivamente usato in un numero di applicazioni complesse:
  - IRMA - Intelligent Resource-bounded Machine Architecture
  - PRS - Procedural Reasoning System
- componenti logici – il modello è stato rigorosamente formalizzato in una famiglia di logiche BDI (Rao & Georgeff, Wooldridge)

# Agenti con Ragionamento Pratico (cont.)

- ragionamento umano pratico

ragionamento pratico = deliberazione + ragionamento means-ends

- deliberazione

- decide **quale** stato degli affari vuole raggiungere

l'output della deliberazione sono le **intenzioni**

- ragionamento means-ends

- decide **come** raggiungere *quello stato degli affari*

l'output del ragionamento means-ends sono i **piani**

# Agenti con Ragionamento Pratico (cont.)

- deliberazione
  - Identificazione e selezione di Obiettivi e Intenzioni
- means-ends reasoning
  - Pianificazione, valutazione del piano, selezione del piano

# Agenti con Ragionamento Pratico:

## 1. Deliberazione: Intenzioni e Desideri

- le intenzioni sono più forti dei desideri
  - “Il mio **desiderio** di giocare a basket questo pomeriggio è solo un potenziale influente sulla mia condotta di questo pomeriggio. Deve paragonarsi con i miei altri rilevanti desideri [ . . . ] prima che si decida cosa farò. Nel contesto, se **io intendo** giocare a basket questo pomeriggio, ho già deciso: normalmente non continuerò a vedere i pro e i contro. Quando arriverà pomeriggio, procederò normalmente ad eseguire la mia intenzione” [Bratman, 1990]

# Agenti con Ragionamento Pratico: Intenzioni

- gli agenti dovrebbero essere capaci di determinare modi di raggiungere intenzioni
  - *Se ho un'intenzione su  $\Phi$ , ti aspetterai che io impieghi risorse per decidere come ottenere  $\Phi$*

# Agenti con Ragionamento Pratico: Intenzioni

- gli agenti non possono adottare intenzioni che sono in conflitto
  - *Se ho un'intenzione su  $\Phi$ , non ti aspetterai che adotterò una intenzione  $\Psi$  che è incompatibile con  $\Phi$*
- gli agenti sono inclini nel riprovare se il loro tentativo di raggiungere l'intenzione fallisce
  - *Se il primo tentativo dell'agente di raggiungere  $\Phi$  fallisce, proverà un piano alternativo per raggiungere  $\Phi$*

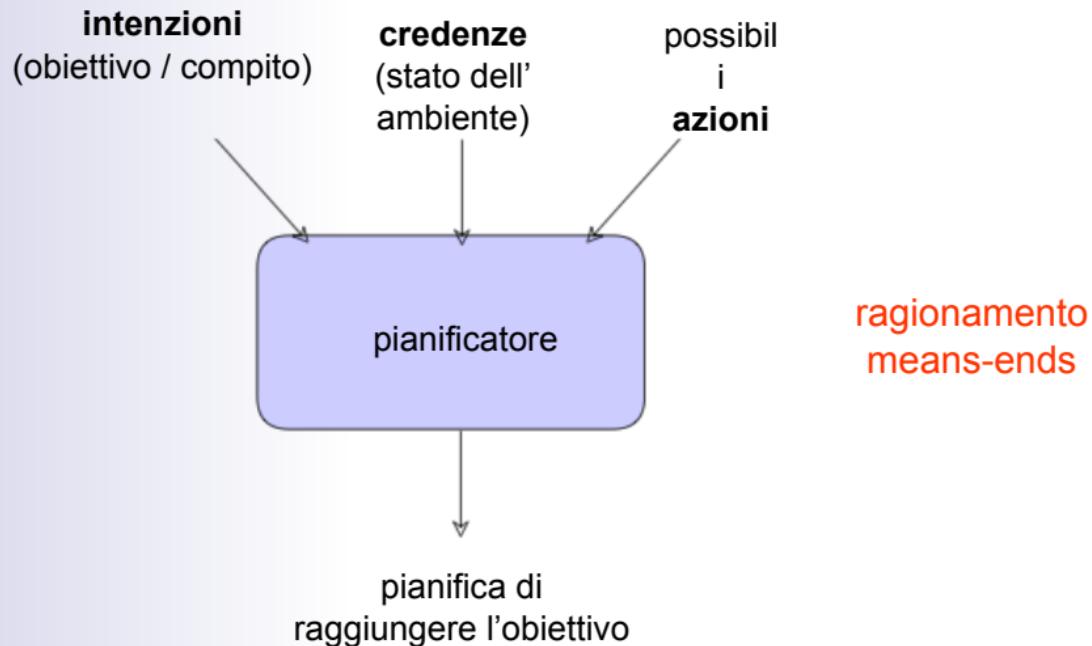
# Agenti con Ragionamento Pratico: Intenzioni

- gli agenti credono che le loro intenzioni sono possibili
  - *Credono che c'è almeno un modo in cui la loro intenzione può essere raggiunta.*
- gli agenti non credono che potranno non perseguire le loro intenzioni
  - *Non sarebbe razionale adottare un'intenzione  $\Phi$  se sono certo che fallirei nel perseguiirla*

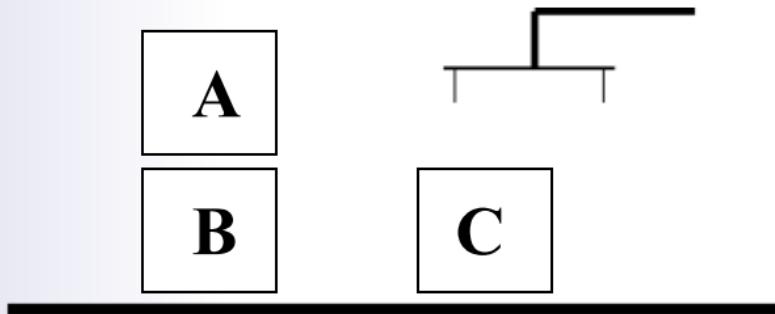
# Agenti con Ragionamento Pratico: Intenzioni

- sotto certe circostanze, gli agenti credono di poter perseguire le loro intenzioni
- *Se io intendo  $\Phi$ , allora credo che sotto “normali circostanze” io avrò successo con  $\Phi$*
- gli agenti non hanno bisogno di intendere tutti gli effetti collaterali
- *Io posso credere che andare dal dentista sia doloroso, e potrei intendere di dover andare dal dentista. Questo non vuol dire che io intenda soffrire!*

## Agenti con Ragionamento Pratico: 2. Ragionamento Means-ends



# Il Mondo dei Blocchi



- illustra il ragionamento means-ends con riferimento al *mondo dei blocchi*
- Contiene un braccio robotico, 3 blocchi (A, B, e C) di grandezza uguale, e un tavolo

# Ontologia del Mondo dei Blocchi

- Per rappresentare questo ambiente, abbiamo bisogno di un'*ontologia*

$On(x, y)$       obj  $x$  *in cima a* obj  $y$

$OnTable(x)$       obj  $x$  *è sul tavolo*

$Clear(x)$       niente è sopra obj  $x$

$Holding(x)$       il braccio sta tenendo  $x$

# Il Mondo dei Blocchi

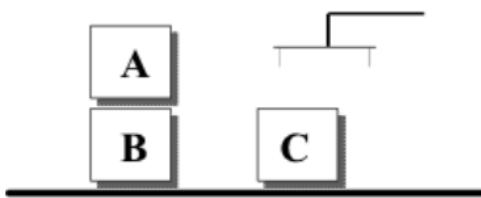
- Qui c'è una rappresentazione del mondo dei blocchi descritto sopra:

$Clear(A)$

$On(A, B)$

$OnTable(B)$

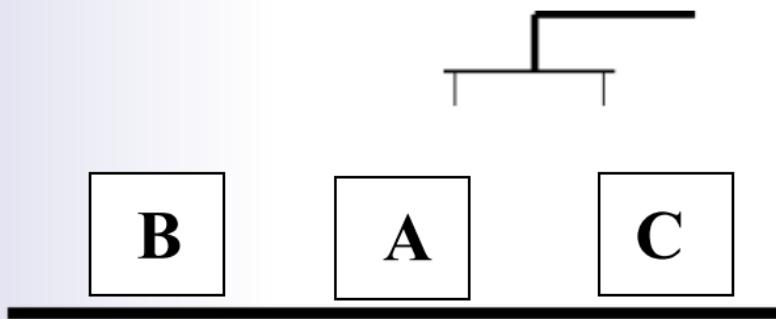
$OnTable(C)$



- Usa l'*assunzione del mondo chiuso*: qualsiasi cosa non listata è assunta essere *falsa*

# Il Mondo dei Blocchi

- Un *obiettivo* è rappresentato come insieme di formule
- Qui c'è un obiettivo:  
 $OnTable(A) \sqcup OnTable(B) \sqcup OnTable(C)$

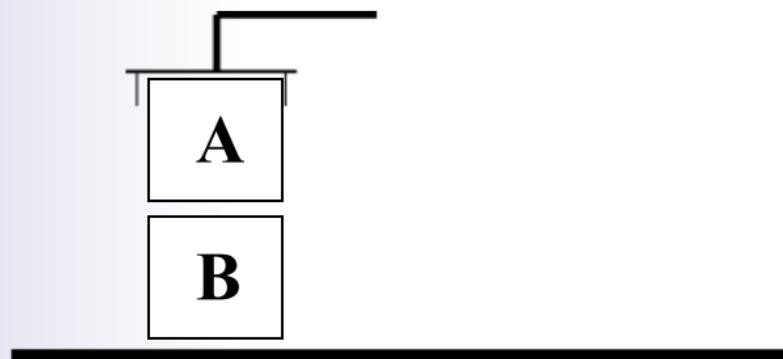


# Il Mondo dei Blocchi

- Le Azioni sono rappresentate usando una tecnica sviluppata nel pianificatore STRIPS (**NOTAZIONE**)
- Ogni azione ha:
  - un *nome*
  - che può avere argomenti
  - una *lista di precondizioni*  
lista di fatti che devono essere veri per le azioni che devono essere eseguite
  - una *lista di cancellazione*  
lista di fatti che non sono più veri dopo che le azioni sono eseguite
  - una *lista di aggiunta*  
lista di fatti diventati veri eseguendo l'azione

Ognuna di queste può contenere *variabili*

# Operatori nel Mondo dei Blocchi



- Esempio 1:  
Lo *stack* azione capita quando il braccio del robot mette l'oggetto  $x$  che sta mantenendo in cima all'oggetto  $y$ .

*Stack( $x, y$ )*

pre *Clear(y)  $\sqcap$  Holding(x)*

del *Clear(y)  $\sqcap$  Holding(x)*

add *ArmEmpty  $\sqcap$  On(x, y)*

# Operatori nel Mondo dei Blocchi

- Esempio 2:

L'operazione *unstack* capita quando il braccio robotico prende un oggetto *x* dalla cima di un altro oggetto *y*.

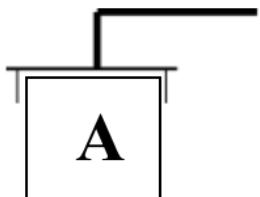
*UnStack(x, y)*

pre *On(x, y) ⊓ Clear(x) ⊓ ArmEmpty*

del *On(x, y) ⊓ ArmEmpty*

add *Holding(x) ⊓ Clear(y)*

Stack e UnStack sono *inversi* l'uno dell'altro.



B

# Operatori nel Mondo dei Blocchi

- Esempio 3:

L'azione *pickup* avviene quando il braccio prende un oggetto  $x$  dal tavolo.

$Pickup(x)$

pre  $\text{Clear}(x) \sqcap \text{OnTable}(x) \sqcap \text{ArmEmpty}$

del  $\text{OnTable}(x) \sqcap \text{ArmEmpty}$

add  $\text{Holding}(x)$

- Esempio 4:

L'azione *putdown* avviene quando il braccio poggia l'oggetto  $x$  sul tavolo.

$Putdown(x)$

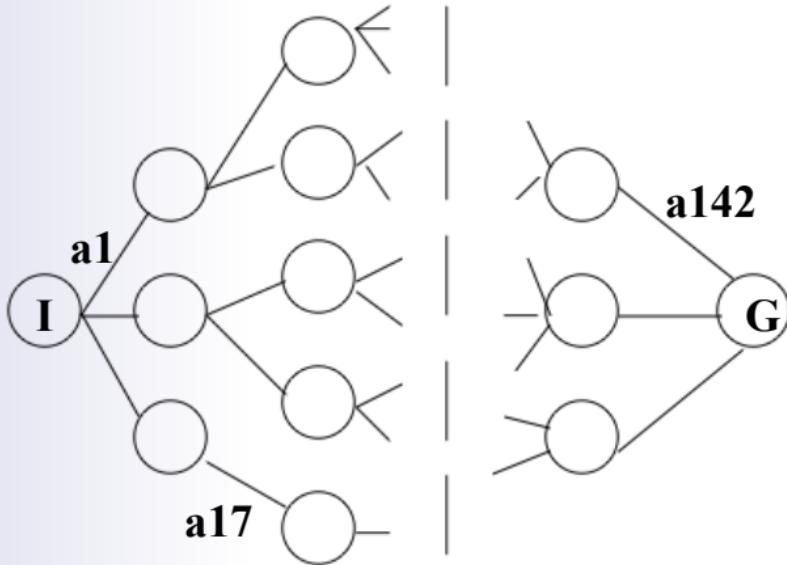
pre  $\text{Holding}(x)$

del  $\text{Holding}(x)$

add  $\text{Clear}(x) \sqcap \text{OnTable}(x) \sqcap \text{ArmEmpty}$

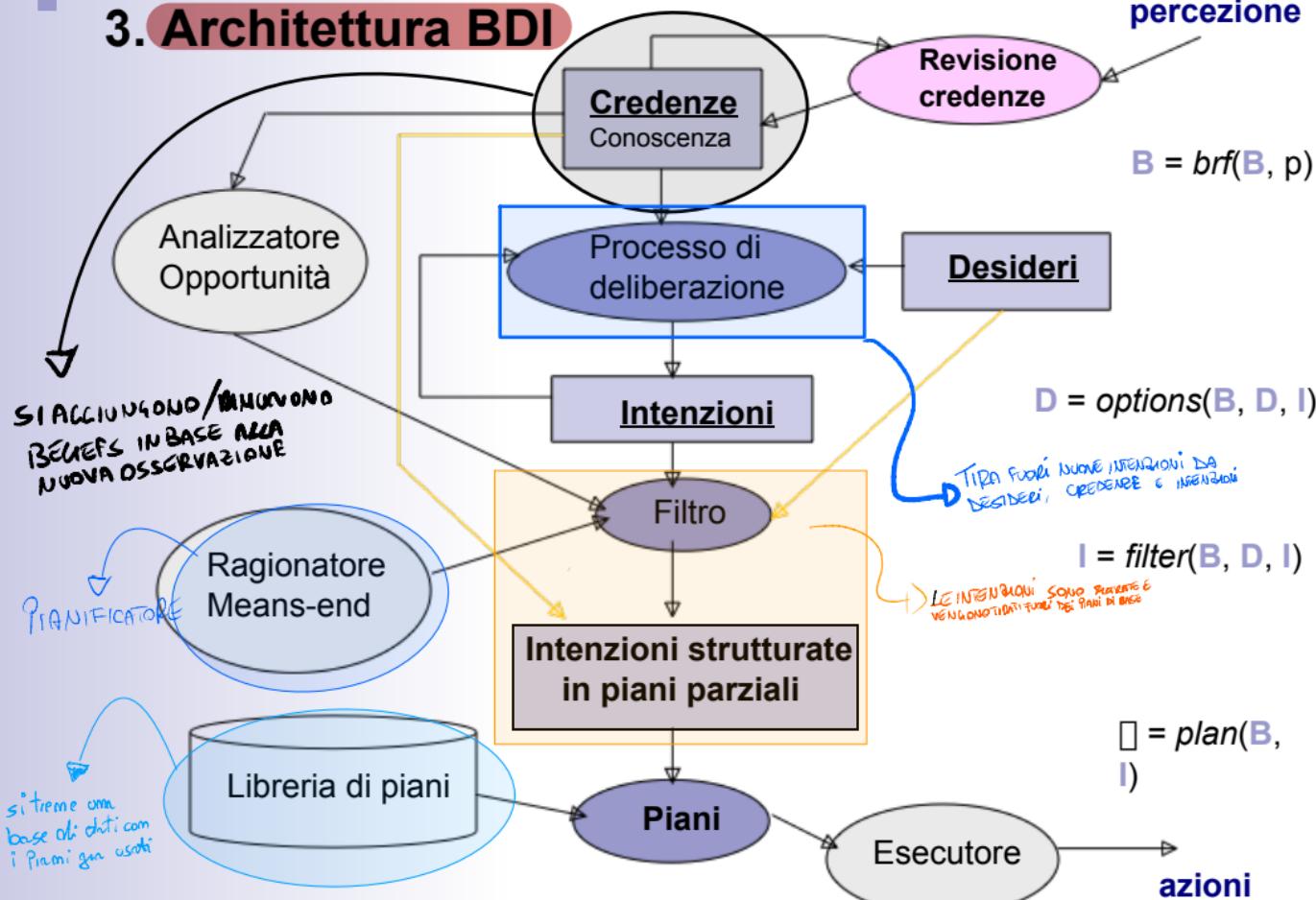
# Un Piano

i PLANNER non esplorano  
a caso



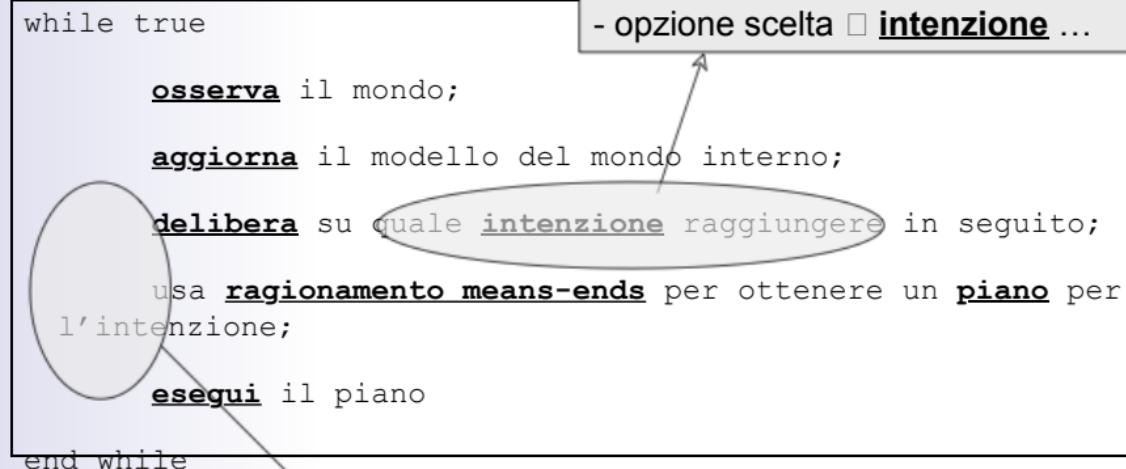
- Cos'è un piano?  
Una sequenza (lista) di azioni, con le variabili sostituite da costanti.

### 3. Architettura BDI



# Agenti con Ragionamento Pratico (cont.)

- controllo del ciclo dell'agente



# Implementare Agenti con Ragionamento Pratico

- Rendiamo l'algoritmo più formale:

```
Agent Control Loop Version 2
1.  $B := B_0; /* initial beliefs */$ 
2. while true do
3.   get next percept  $\rho$ ;
4.    $B := brf(B, \rho);$ 
5.    $I := deliberate(B);$ 
6.    $\pi := plan(B, I);$ 
7.    $execute(\pi)$ 
8. end while
```

# Implementare Agenti con Ragionamento Pratico

- questa versione: comportamento ottimo se
  - deliberazione e ragionamento means-ends impiegano un impercettibile piccolo ammontare di tempo;  
OR
  - il mondo è garantito che rimanga statico mentre l'agente sta deliberando ed eseguendo il ragionamento means-ends;  
OR
  - una intenzione che è ottima quando ottenuta al tempo t0 (il tempo in cui il mondo è osservato) è garantita che rimanga ottimale fino al tempo t2 (il tempo in cui l'agente ha trovato un percorso di azioni per raggiungere l'intenzione).

# Deliberazione

- La funzione *deliberate* può essere decomposta in due componenti funzionali distinte:
  - *generazione di opzioni*  
in cui l'agente genera un insieme di possibili alternative;  
Rappresenta la generazione di opzioni attraverso una funzione, *options*, che prende le credenze correnti dell'agente e le intenzioni correnti e da loro determina un insieme di opzioni (= *desideri*)
  - *filtraggio*  
in cui l'agente sceglie tra le alternative in concorrenza e si impegna a trovare il modo di raggiungerle.  
In modo da scegliere tra le opzioni in concorrenza, un agente usa una funzione *filter*.

# Deliberazione

Agent Control Loop Version 3

- 1.
2.  $B := B_0;$
3.  $I := I_0;$
4. while true do
5.     get next percept  $\rho$ ;
6.      $B := brf(B, \rho);$
7.      $D := options(B, I);$
8.      $I := filter(B, D, I);$
9.      $\pi := plan(B, I);$
10.      $execute(\pi)$
11. end while

## Agenti con Ragionamento Pratico (cont.)

- Se un'opzione passa con successo attraverso la funzione filtro ed è scelta dall'agente come intenzione, diciamo che  
**l'agente ha creato una dedizione (commitment) su quella opzione**
- La dedizione implica una persistenza temporale sull'intenzione. Una volta che un'intenzione è adottata, non deve essere immediatamente lasciata.

# Agenti con Ragionamento Pratico(cont.)

**Domanda:** Quanto dedicato deve essere un'agente sulla sua intenzione?

- grado di dedizione
  - **dedizione cieca**
    - ≈ dedizione fanatica: continua finchè la raggiunge
  - **dedizione a mente singola**
    - continua finchè la raggiunge o non è più possibile
  - **dedizione a mente aperta**
    - continua finchè non è più creduta possibile

# Strategie di Dedizione

- Un agente ha dedizione
  - verso *ends* (cioè il desiderio da portare a termine)
  - e *means* (cioè il meccanismo attraverso il quale l'agente desidera raggiungere lo stato degli affari)
  - la versione corrente del controllo di ciclo dell'agente è troppo dedicata, sia a means che a ends
- modifica: *riplanifica* se un piano va male

```

1.
2.    $B := B_0;$ 
3.    $I := I_0;$ 
4.   while true do
5.       get next percept  $\rho$ ;
6.        $B := brf(B, \rho);$ 
7.        $D := options(B, I);$ 
8.        $I := filter(B, D, I);$ 
9.        $\pi := plan(B, I);$ 
10.      while not empty( $\pi$ ) do
11.           $\alpha := hd(\pi);$ 
12.          execute( $\alpha$ );
13.           $\pi := tail(\pi);$  da azioni restanti:
14.          get next percept  $\rho$ ;
15.           $B := brf(B, \rho);$ 
16.          if not sound( $\pi, I, B$ ) then
17.               $\pi := plan(B, I)$ 
18.          end-if
19.      end-while
20.  end-while

```

]

*Stesso di  
Piano*

*PIANO: sequenza di azioni:  
 $\alpha = \{a, b, \dots\}$*

*Se un percepione è riconosciuta se il piano è ancora fattibile*

*Reattività,  
riplanificazione*

# Strategie di Dedizione

- questa versione è ancora troppo dedicata alle intenzioni:
  - non si ferma a considerare se l'intenzione è appropriata
- 
- modifica: si ferma per determinare se l'intenzione è stata raggiunta o se è impossibile:  
“*Dedizione a mente singola*”

# Dedizione a Mente Singola

Agent Control Loop Version 5

```
2.    $B := B_0;$ 
3.    $I := I_0;$ 
4.   while true do
5.       get next percept  $\rho$ ;
6.        $B := brf(B, \rho);$ 
7.        $D := options(B, I);$ 
8.        $I := filter(B, D, I);$ 
9.        $\pi := plan(B, I);$ 
10.      while not empty( $\pi$ )
11.          or succeeded( $I, B$ )
12.          or impossible( $I, B$ ) do
13.               $\alpha := hd(\pi);$ 
14.              execute( $\alpha$ );
15.               $\pi := tail(\pi);$ 
16.              get next percept  $\rho$ ;
17.               $B := brf(B, \rho);$ 
18.              if not sound( $\pi, I, B$ ) then
19.                   $\pi := plan(B, I)$ 
20.              end-if
21.      end-while
22.  end-while
```

*Lascia intenzioni  
che sono impossibili  
o sono state raggiunte*

Si fermano anche se raggiunge l'obiettivo  
(prima di completare il piano) o se si trova  
impossibile

*Reattività,  
riplanificazione*

# Riconsiderazione di Intenzioni

- Il nostro agente deve riconsiderare le sue intenzioni quando:
  - ha completamente eseguito un piano per raggiungere le sue intenzioni correnti; o
  - crede di avere raggiunto le sue intenzioni correnti; o
  - crede che le sue intenzioni correnti non siano più possibili.
- Questo è limitato nel modo che permette ad un agente di **riconsiderare** le sue intenzioni
  - modifica:  
Riconsidera l'intenzione dopo avere eseguito ogni azione  
*“Dedizione a mente aperta”*

```

1.
2.    $B := B_0;$ 
3.    $I := I_0;$ 
4.   while true do
5.     get next percept  $\rho$ ;
6.      $B := brf(B, \rho);$ 
7.      $D := options(B, I);$ 
8.      $I := filter(B, D, I);$ 
9.      $\pi := plan(B, I);$ 
10.    while not (empty( $\pi$ )
11.          or succeeded( $I, B$ )
12.          or impossible( $I, B$ )) do
13.       $\alpha := hd(\pi);$ 
14.      execute( $\alpha$ );
15.       $\pi := tail(\pi);$ 
16.      get next percept  $\rho$ ;
17.       $B := brf(B, \rho);$ 
18.       $D := options(B, I);$ 
19.       $I := filter(B, D, I);$ 
20.      if not sound( $\pi, I, B$ ) then
21.         $\pi := plan(B, I)$ 
22.      end-if
23.    end-while
24.  end-while

```

## Dedizione a mente aperta

SINGLE-MINDED

A differenza di Primo raccomanda

le intenzioni

get next percept  $\rho$ ;  
 $B := brf(B, \rho);$   
 if not sound( $\pi, I, B$ ) then  
 $\pi := plan(B, I)$

\*

# Riconsiderazione di Intenzioni

- Ma la riconsiderazione di intenzioni è *costosa!*  
Un dilemma:
  - un agente che non si ferma a riconsiderare le sue intenzioni abbastanza spesso continuerà a raggiungere le sue intenzioni anche se sarà chiaro che non potrà riuscire a farlo, o non c'è nessuna ragione di farlo
  - un agente che riconsidera *costantemente* le sue intenzioni potrebbe spendere tempo eccessivo lavorando per raggiungerle, con il rischio di non riuscire a raggiungerne nessuna.
- Soluzione: incorporare un esplicito componente di **controllo a metalivello** che decide quando riconsiderare

*L'METHODO INVENTATO DALLA COSTANTINI*

```

1.
2.    $B := B_0;$ 
3.    $I := I_0;$ 
4.   while true do
5.       get next percept  $\rho$ ;
6.        $B := brf(B, \rho);$ 
7.        $D := options(B, I);$ 
8.        $I := filter(B, D, I);$ 
9.        $\pi := plan(B, I);$ 
10.      while not ( $empty(\pi)$ 
           or succeeded( $I, B$ )
           or impossible( $I, B$ )) do
11.           $\alpha := hd(\pi);$ 
12.          execute( $\alpha$ );
13.           $\pi := tail(\pi);$ 
14.          get next percept  $\rho$ ;
15.           $B := brf(B, \rho);$ 
16.          * if reconsider( $I, B$ ) then
17.               $D := options(B, I);$ 
18.               $I := filter(B, D, I);$ 
19.          end-if
20.          if not sound( $\pi, I, B$ ) then
21.               $\pi := plan(B, I)$ 
22.          end-if
23.      end-while
24.  end-while

```

*controllo metalivello*


**MI CHIEDO SE  
RICONSIDERARE  
LE INTENZIONI**

# Riconsiderazione di Intenzioni Ottimale

- Le sperimentazioni di Kinny e Georgeff hanno investigato sull'efficacia delle strategie di riconsiderazione di intenzioni
- Due differenti tipi di strategie di riconsiderazione furono usate:
  - agente *audace*  
non si ferma a riconsiderare le intenzioni
  - agente *cauto*  
si ferma a riconsiderare ogni azione
- Il *Dinamismo* nell'ambiente è rappresentato dal *grado di cambiamento del mondo*,  $g$

LA VELOCITÀ CON CUI CAMBIA

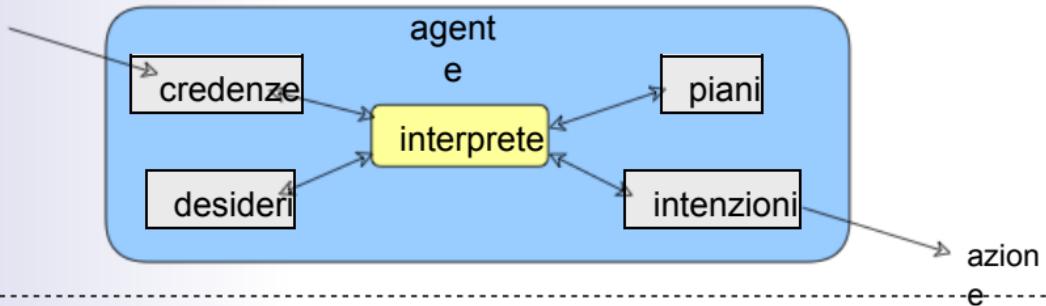
# Riconsiderazione di Intenzioni Ottimale

- Risultati (non sorprendenti):
- Se  $g$  è basso (cioè l'ambiente non cambia velocemente),  
gli agenti audaci sono come quelli cauti.
- quelli cauti perdono tempo a riconsiderare la loro dedizione mentre quelli audaci lavorano e raggiungono le loro intenzioni.
- Se  $g$  è alto (cioè l'ambiente cambia di frequente),  
gli agenti cauti tendono a superare quelli audaci.  
sono abili nel capire quando le intenzioni sono fallite, e sfruttano i vantaggi di situazioni emergenti e nuove opportunità quando si presentano.

# Agenti a Ragionamento Pratico: Sistema di Ragionamento Procedurale (PRS)

- “Architettura BDI” (*credenze / desideri / intenzioni*)
  - strutture dati esplicite per b/d/i
- **pianificazione**
  - nessuna pianificazione “al volo”  librerie di piani
    - un piano: obiettivo (post-condizione)  
contesto (pre-condizione)  
corpo (sequenza di azioni / sotto-obiettivi)
- **pila di intenzioni**

sensore in input

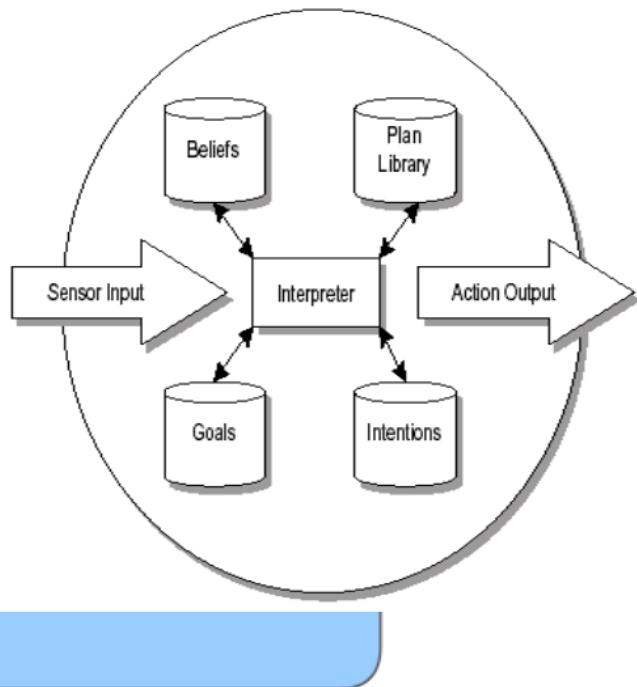


# Agenti BDI Implementati: PRS

PRS – Procedural Reasoning System (Georgeff, Lansky)

ogni agente è equipaggiato con una *libreria di piani*, che rappresenta la *conoscenza procedurale* dell'agente: conoscenza sui meccanismi che possono essere usati dall'agente in modo da realizzare le sue intenzioni

le opzioni disponibili per un'agente sono direttamente determinate dai piani che ha un agente: un agente con nessun piano non ha opzioni in aggiunta, gli agenti hanno una rappresentazione esplicita di credenze, desideri e intenzioni, come sopra



# PRS

- Il PRS consiste di:
  - un database contenente le **credenze** correnti o fatti sul mondo;
  - un insieme di **obiettivi** correnti da essere realizzati;
  - una libreria di **piani** che descrivono come certe sequenze di azioni e test possono essere eseguite per raggiungere obiettivi dati o per reagire a particolari situazioni;
  - una **struttura di intenzioni** contenente quei piani che sono stati scelti per l'esecuzione.
  - un'area di conoscenza

# PRS

- L'Area di Conoscenza (KA) è una base di conoscenza contenente conoscenze procedurali su come raggiungere gli obiettivi dati o reagire a certe situazioni. Ogni KA consiste di un corpo, che descrive i passi della procedura, ed una condizione di invocazione che specifica sotto quale situazione la KA è utile.
- La condizione di invocazione contiene una parte di attivazione che descrive gli eventi che devono accadere in modo da eseguire una KA (l'acquisizione di nuovi obiettivi, cambi nelle credenze del sistema...)

## **PRS e KA**

- L'interprete amministra questi componenti, selezionando piani appropriati basati su conoscenze e obiettivi del sistema, inserendo quelli che sono stati scelti nella struttura intenzionale ed eseguendoli.
- L'agente reagisce ad eventi, che sono generati da modifiche alle loro credenze o da stimoli provenienti dall'ambiente aggiungendo nuovi obiettivi.
- Un evento può attivare uno o più piani. Un piano che un agente decide di attuare sulla base di un insieme di politiche predefinite diventa un'intenzione.
- A qualsiasi dato momento, la struttura dell'intenzione potrebbe contenere un certo numero di intenzioni ma non tutte verrebbero soddisfatte.

## **PRS e KA**

- Alcune intenzioni saranno sospese o annullate, alcune aspetteranno bloccare l'avvenire di certe condizioni di attivazione, alcune saranno intenzioni di meta-livello per decidere quali azioni scegliere attraverso certe alternative.
- I PRS contengono un efficiente meccanismo per rispondere per tempo a cambi di ambiente.
- Da quando l'interprete prova continuamente a confrontare le KA con qualsiasi nuova conoscenza o obiettivo acquisiti, il sistema è in grado di notificare le nuove KA applicabili dopo che ogni azione primitiva è stata intrapresa.

# Agenti con Ragionamento Pratico (cont.)

- Altre Implementazioni
  - IRMA
  - DMARS
  - Jason, per AgentSpeak(L) (java – sourceforge.net)
  - JAM (java)
  - JACK (java)
  - ...

# Un linguaggio logico basato sul BDI: AgentSpeak(L)

- AgentSpeak(L) ha molte similitudini con la tradizionale logica di programmazione.
- Si mostra quasi intuitivo per quelli familiari con la logica di programmazione.
- Ha una chiara notazione, oltre a fornire specifiche eleganti degli agenti BDI.
- Ha una chiara semantica.
- Ha un interprete efficiente, chiamato Jason.

# Un linguaggio logico basato sul BDI: AgentSpeak(L)

- E' stato progettato per riempire il vuoto tra la teoria BDI (elegante ma troppo pesante) e la programmazione pratica.
- Limitazioni con BDI:
  - Ogni regola definisce (localmente) una parte di un piano;
  - Interi piani risultano usando (in un contesto come Prolog) differenti regole che, legate insieme, determinano passo dopo passo un piano intero;
  - Nessuna libreria di piano, nessuna selezione di intenzioni, nessun concetto chiaro di dedizione (costruito nel programma)

# AgentSpeak(L) – Nozioni Base

- ***Atomo di credenza***
  - È un predicato di primo ordine nella notazione usuale
  - Atomi di credenza o loro negazioni sono definiti ***letterali di credenza***.
- ***Obiettivo***
  - È uno stato del sistema che l'agente vuole raggiungere.

# AgentSpeak(L) – Nozioni Base

- Due tipi di obiettivi:
  - ***Obiettivi da raggiungere***
    - Predicati prefissi con l'operatore “!”
    - Mostra che l'agente vuole raggiungere uno stato del mondo dove il predicato associato è vero.
    - in pratica questi iniziano l'esecuzione di *sottopiani*.
  - ***Obiettivi di test***
    - Predicati prefissi con l'operatore ‘?’
    - Restituiscono un'unificazione per i predicati associati con una delle credenze dell'agente.  
Falliscono se non è trovata nessuna unificazione.

# AgentSpeak(L) – Nozioni Base

- ***Obiettivi attivabili***

- Definisce quali eventi possono iniziare l'esecuzione di un piano.
- Un evento può essere
  - interno, quando un sotto-obiettivo deve essere raggiunto
  - Esterno, quando generato dall'aggiornamento delle credenze come risultato di percezione dell'ambiente.
- Due tipi di eventi attivabili:
  - Relazionati all'*addizione* ('+') e *cancellazione* ('-') di attitudini (credenze o obiettivi).

# AgentSpeak(L) – Nozioni Base

- **Piani**

- Si riferisce alle *azioni di base* che un agente è capace di eseguire nel suo ambiente.

$$p ::= te : ct \leftarrow h$$

Dove:

- *te* – evento attivato (*denota lo scopo per quel piano*)
- *ct* – una congiunzione di letterali di credenza rappresentanti un contesto.
- Il contesto deve essere una conseguenza logica delle credenze correnti di quell'agente per il piano per essere applicabile.
- *h* – sequenza di azioni base o (sotto)obiettivi che l'agente ha da raggiungere (o testare) quando il piano, se applicabile, è scelto per l'esecuzione.

Evento attivabile

Contesto

```
+concert (A,V) : likes(A) <-
    !book_tickets(A,V) .
```

```
+ !book_tickets(A, V) :
    ~busy(phone)
    <- call(V) ;
    ...
    !choose_seats(A,V) .
```

Obiettivo da raggiungere inserito

Azione base

# Architetture concrete per agenti

## Agenti a Ragionamento Deduttivo

1956 – presente

"Gli agenti prendono decisioni su cosa devono fare attraverso la manipolazione dei simboli. La sua espressione più pura propone che l'agente usa il ragionamento logico esplicito in modo da decidere cosa fare."

## Agenti a Ragionamento Pratico

1990 – presente

"Gli agenti usano ragionamenti pratico (attraverso le azioni, non attraverso le credenze) – credenze / desideri / intenzioni."

## Agenti Reattivi

1985 – presente

"Problemi con il ragionamento simbolico hanno portato alla creazione degli agenti reattivi"

## Agenti Ibridi

1989 – presente

"Le Architetture Ibride provano a combinare il meglio dell'architettura a ragionamento e reattività"

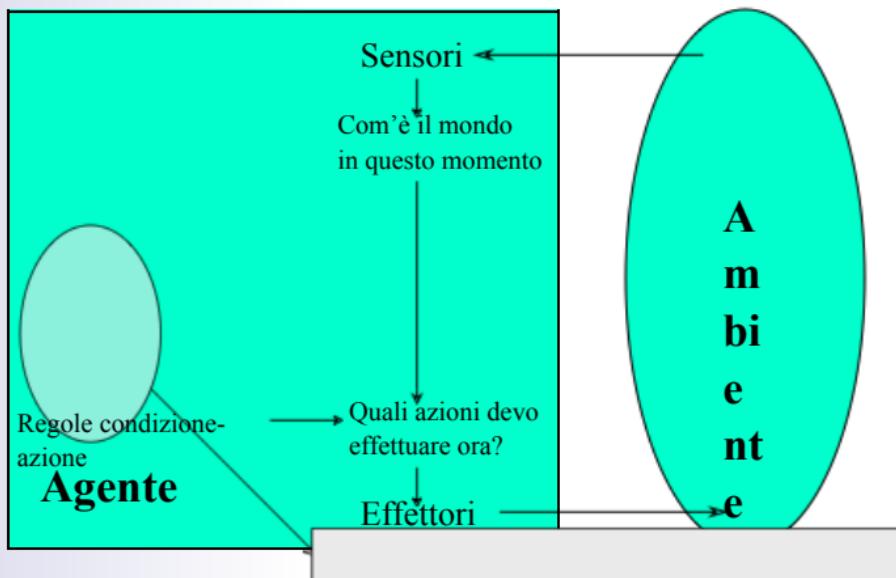
### 3. Agenti Reattivi

- “simbolismo”
- “connessionismo”



- “simbolismo”
  - “l’intelligenza richiede una simbolica rappresentazione del mondo”
- “connessionismo”
  - Il comportamento intelligente è prodotto di interazioni con l’ambiente ed emerge da semplici comportamenti
  - Esempio: comportamento sociale (es. Insetti sociali)

# Un semplice agente reattivo



# Rodney Brooks

- Architettura di classificazione di Brooks (subsumption)
- PENGI
- Automatismo Situato RULER / GAPPSS
- Pattie Maes: “Architettura di rete di comportamenti”
- Architettura Free-flow

# Rodney Brooks (subsumption)

- Due idee base
  - situazionismo e incorporazione
    - L'intelligenza “reale” è situata nel mondo, non in sistemi scorporati come i dimostratori di teoremi o sistemi esperti
  - Intelligenza ed emergenza
    - Comportamenti “intelligenti” si mostrano come risultati dell’interazione di un agente con il suo ambiente
    - L’intelligenza è ‘nell’occhio dello spettatore’, non è una proprietà innata o isolata

# Rodney Brooks (subsumption)

- Due tesi chiave
  - Intelligenza senza rappresentazione
    - I comportamenti intelligenti possono essere ottenuti senza rappresentazione esplicita del tipo che l'IA simbolica propone
  - Intelligenza senza ragionamento
    - I comportamenti intelligenti possono essere ottenuti senza ragionamento astratto esplicito del tipo che l'IA simbolica propone

# Rodney Brooks: Architettura di Classificazione (subsumption)

- Architettura di Classificazione
  - “una gerarchia di comportamenti atti a completare lavori”
  - ogni regola / comportamento:  
se situazione allora azione
    - mappa la percezione direttamente sulle azioni
  - Ogni comportamento ‘compete’ con altri per esercitare il controllo sull’agente

# Rodney Brooks: Architettura di subsumption

- Architettura di Classificazione
  - selezione azioni: organizza le regole in strati
    - strati bassi inibiscono ("assumono") gli strati alti
    - Gli strati bassi rappresentano un tipo più primitivo di comportamento (come evitare gli ostacoli) ed hanno precedenza sugli strati più in alto gerarchicamente

# Rodney Brooks: Architettura di Classificazione (subsumption)

- Computazionalmente: molto semplice
- “alcuni dei robot fanno compiti che sarebbero impressionanti se fossero eseguiti da sistemi di Al simbolica”

# Rodney Brooks: Architettura di Classificazione (subsumption)

- Un comportamento è una coppia  $(c, a)$   
dove  $c \sqsubset P$  è un insieme di percezioni chiamate le  
condizioni  
e  $a \sqsubset A$  è un'azione
  - Un comportamento  $(c, a)$  può partire quando l'ambiente è  
nello stato  $s \sqsubset S$  se  $\text{see}(s) \sqsubset c$

# Rodney Brooks: Architettura di Classificazione (subsumption)

- $Beh = \{(c, a) \mid c \in P \text{ ed } a \in A\}$  è l'insieme di tutte queste regole
- Associato con l'insieme di regole di comportamento di un agente,  $R \sqsubset Beh$  è una relazione di inibizione binaria dell'insieme di comportamenti  
 $< \sqsubset R \sqsubset R$   
Questa relazione è un ordinamento totale su  $R$

# Rodney Brooks: Architettura di Classificazione (subsumption)

- $b_1 < b_2$  se  $(b_1, b_2) \sqsubset <$   
*“ $b_1$  inhibisce  $b_2$ ” o “ $b_1$  assume  $b_2$ ”*  
( $b_1$  è più basso in gerarchia di  $b_2$ ,  
e otterrà priorità su  $b_2$ )
- La “funzione di selezione azione” è quindi definita come segue...

# Rodney Brooks: Architettura di Classificazione (subsumption)

```
function action ( $p : P$ ) :  $A$ 
var  $fired : \wp(R)$ 
var  $selected : A$ 
begin
     $fired := \{(c,a) \mid (c,a) \in R \text{ and } p \sqsubseteq c\}$ 
    for each  $(c,a) \in fired$  do
        if  $\exists (c',a') \in fired$  such that  $(c',a') < (c,a)$  then
            return  $a$ 
        end-if
    end for
    return  $null$ 
end function action
```

Computa tutti i comportamenti che può assumere

Determina se alcuni comportamenti assumono altri

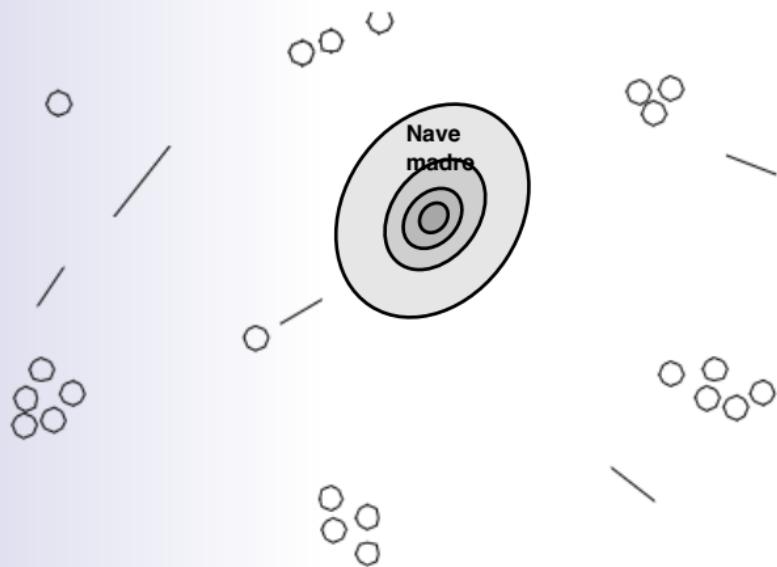
Restituisce l'azione appropriata o null

# Architettura di Classificazione (subsumption) : Mars Explorer

- Mars Explorer (L. Steels)
  - obiettivo
  - *Per esplorare un pianeta distante e in particolare per collezionare esemplari di una roccia particolare, la locazione delle rocce non si conosce in anticipo ma si sa che gli esemplari tendono ad essere raccolti in posti comuni*

# Architettura di Classificazione (subsumption) : Mars Explorer

- Mars Explorer (L. Steels)
- assunzione (*COSA DOBBIANO?*)
  - la nave madre diffonde un segnale radio
    - diminuisce con la distanza
  - nessuna mappa disponibile
  - collaborativo



veicolo autonomo

rocci

# Architettura di Classificazione (subsumption) : Mars Explorer

- soluzione singolo esploratore:
  - comportamenti / regole
    1. se ostacolo allora cambia direzione
    2. se trasporta esemplare e in base allora lascialo
    3. se trasporta esemplare e non in base allora viaggia attraverso il campo del segnale della nave madre
    4. se trova esemplari allora prendili
    5. se vero allora cammina casualmente
  - relazione d'ordine totale  
 $1 < 2 < 3 < 4 < 5$

# Architettura di Classificazione (subsumption) : Mars Explorer

- soluzione multi-esploratore ?
  - pensaci ...
  - se un agente trova una roccia - comunica ?
    - raggio ?
    - posizione ?
    - come risolvere con questi messaggi?

# Architettura di Classificazione (subsumption) : Mars Explorer

- soluzione multi esploratore ?
  - comunicazione indiretta:
    - ogni agente porta “pezzi radioattivi” che possono essere lasciati, presi e rilevati da robot passanti
    - la comunicazione attraverso l’ambiente è chiamata **stigmergia**

# Architettura di Classificazione (subsumption) : Mars Explorer

- soluzione ispirata dal comportamento delle formiche
  - l'agente crea un percorso di pezzi radioattivi fino alla nave madre quando trova una roccia
  - se un altro agente arriva vicino al percorso, può seguirlo fino al pezzo trovato
- raffinamenti:
  - gli agenti che seguono il percorso prendono alcuni pezzi per rendere il percorso sconnesso
  - il percorso che porta all'esemplare trovato sarà infine rimosso

# Architettura di Classificazione (subsumption) : Mars Explorer

- insieme di regole modificate
- se trova un ostacolo allora cambia direzione
- se porta un esemplare ed è in base allora poggia esemplare
- se porta un esemplare e non è in base allora *poggia 2 pezzi* e viaggia lungo il gradiente, se trova un esemplare allora prende un esemplare
- *se sente pezzi    allora prende 1 pezzo e viaggia lungo il gradiente*
- se vero allora muoviti casualmente (niente di meglio da fare)
- relazione d'ordine: 1 < 2 < 3 < 4 < 5 < 6

## Architettura di Classificazione (subsumption) : Mars Explorer

- raggiunge prestazioni all'incirca ottime in molte situazioni
- soluzioni economiche e robuste (la perdita di un singolo agente non è critica).
- L. Steels sostiene che gli agenti (deliberativi) sono “interamente irrealistici” per questo problema.

# Architettura di Classificazione (subsumption) : Mars Explorer

- vantaggi
  - semplice
  - economico
  - trattabile computazionalmente
  - robusto contro fallimenti
- svantaggi
  - gli agenti agiscono nel breve termine in quanto usano solo informazioni locali
  - nessun apprendimento
  - come ingegnerizzare questi agenti? Difficile se interagiscono più di 10 regole
  - nessun metodo formale di analisi e predizione

# Rete di comportamenti di Maes per agenti situati

- osservazioni
  - “l’approccio deliberativo non funziona in ambienti reali dinamici”
    - fragilità, inflessibilità, tempo basso di risposta
  - comportamento orientato all’obiettivo
    - nozione esplicita di obiettivo (non implica pianificazione!)
    - inerzia nel comportamento

# Rete di comportamenti di Maes per agenti situati

- osservazioni
  - agenti: comportamenti multipli in un preciso momento in una data situazione
    - parallelo
  - conflitto di comportamenti
    - usa lo stesso meccanismo (azione) o risorse condivise
  - gli agenti hanno comportamenti o azioni “in concorrenza”

## Rete di comportamenti di Maes per agenti situati (cont.)

- un agente è un insieme di “moduli di competenza” (comportamenti)
- ogni modulo di competenza ha:
  - pre-condizioni (*situazione*)
  - post-condizioni (*aggiunta / cancellazione*)
  - livello di attivazione (~ rilevanza nella situazione corrente)

## Rete di comportamenti di Maes per agenti situati (cont.)

- un agente è un insieme di “moduli di competenza” (comportamenti)
- i moduli di competenza sono collegati
  - in accordo con le pre-/post-condizioni

## Rete di comportamenti di Maes per agenti situati (cont.)

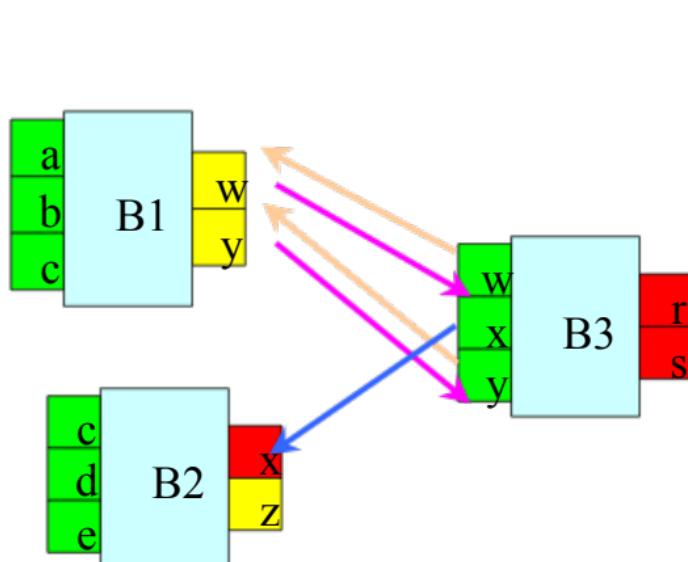
- I moduli di competenza (nodi) con le loro connessioni formano una Rete
- Lo stato iniziale dà “energia” ad alcuni dei nodi della rete.
- L’energia scorre attraverso i collegamenti.

## Rete di comportamenti di Maes per agenti situati (cont.)

- Ogni modulo di competenza è in grado di eseguire un compito (caso più semplice: una singola azione).
- Un modulo di competenza è *abilitato* e quindi esegue il suo compito se ha abbastanza “energia”
  - Livelli di energia necessari associati al modulo.
- L’energia è “trasmessa” ad altri nodi attraverso il collegamento e diffusa attraverso la rete.

# Rete di comportamenti di Maes per agenti situati (cont.)

- rete di comportamenti: un grafo
  - nodi: **comportamenti**
  - archi: tre tipi di collegamenti
    - successore
    - predecessore
    - conflittore



# Rete di comportamenti di Maes per agenti situati (cont.)

- quale azione (comportamento) eseguire?
- flusso di “attivazione energia”
  - obiettivi globali 
    - sorgente di motivazione precostruita
    - alcuna energia è associata all'obiettivo nello stato iniziale
  - ambiente  rilevanza situazionale
    - Una quantità di energia è associata ai moduli che interagiscono con l'ambiente nello stato iniziale
  - comportamenti  (avanti) attivazione dai successori (indietro) attivazione dai predecessori
  - inibizione dai conflittuatori

# Rete di comportamenti di Maes per agenti situati (cont.)

- algoritmo di selezione azione

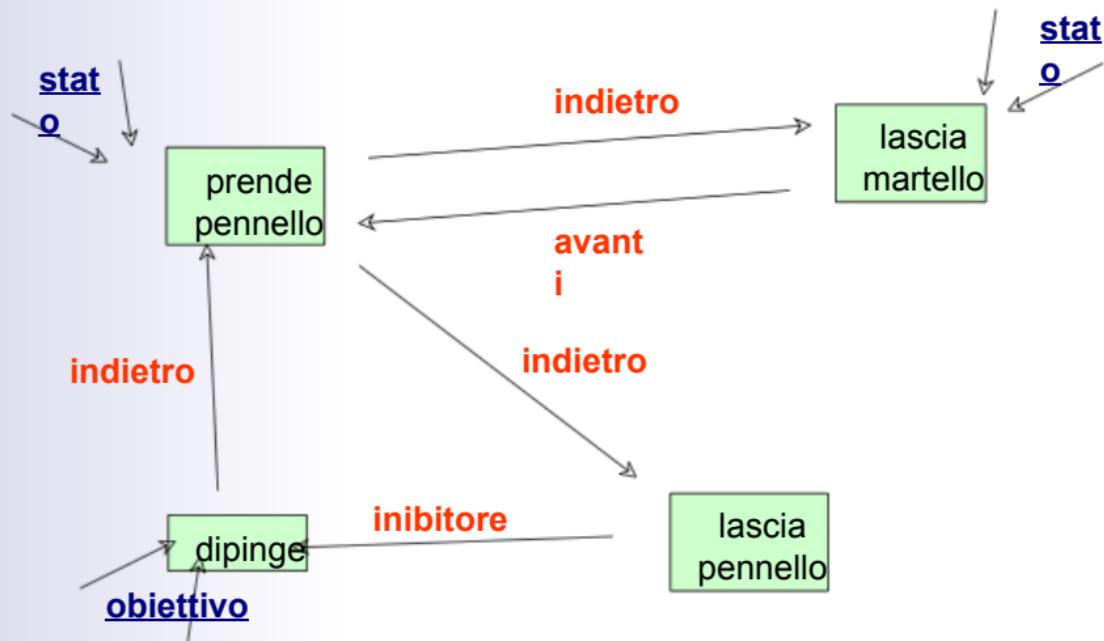
- **fai per sempre**

- **aggiungi** energia di attivazione esterna
  - da obiettivi & ambiente
- **diffondi attivazione/inibizione tra i comportamenti**
  - spedisci attivazione attraverso collegamenti successivi
  - ritorna attivazione attraverso collegamenti precedenti
  - ritorna inibizione attraverso collegamenti conflittuatori
- **decadenza:** la totale attivazione nel sistema è costante
- il comportamento è selezionato se
  - è eseguibile (tutte le precondizioni sono soddisfatte)
  - il suo livello di attivazione è oltre un livello (teta)
  - la sua attivazione è la più alta tra tutti i comportamenti eseguiti/attivati
- **se si esegue un comportamento**
  - la sua attivazione è settata a zero
  - il valore di soglia è riportato al default.
- **se non esiste comportamento, riduce il valore di soglia dell'x%**
  - gli agenti "pensano" per un ciclo e riprovano di nuovo nel ciclo successivo

# Rete di comportamenti di Maes per agenti situati (cont.)

- situazione corrente
  - l'agente ha il martello in mano
  - il pennello si trova di fronte al tavolo

→ = flusso di energia



## Rete di comportamenti di Maes per agenti situati (cont.)

- Settare la dinamica
  - la selezione dell'azione **emerge** dalle dinamiche di spargimento dell'attivazione
  - **parametri** settabili:
    - ammontare di attivazione iniettata dall'ambiente
    - ammontare di energia di attivazione iniettata dagli obiettivi
    - livello di soglia

# Rete di comportamenti di Maes per agenti situati (cont.)

- valutazione / caratteristiche:
  - orientato all'obiettivo
  - reattivo e veloce
  - orientato alle situazioni e opportunistico
  - in qualche modo inerte: incline verso il piano/obiettivo correnti
  - gli obiettivi interagiscono ed evitano conflitti
  - robusto

# Architetture concrete per agenti

## Agenti a Ragionamento Deduttivo

1956 – presente

"Gli agenti prendono decisioni su cosa devono fare attraverso la manipolazione dei simboli. La sua espressione più pura propone che l'agente usa il ragionamento logico esplicito in modo da decidere cosa fare."

## Agenti a Ragionamento Pratico

1990 – presente

"Gli agenti usano ragionamenti pratico (attraverso le azioni, non attraverso le credenze) – credenze / desideri / intenzioni."

## Agenti Reattivi

1985 – presente

""Problemi con il ragionamento simbolico hanno portato alla creazione degli agenti reattivi""

## Agenti Ibridi

1989 – presente

"Le Architetture Ibride provano a combinare il meglio dell'architettura a ragionamento e dell'architettura a reattività"

# Agenti Ibridi

- Meglio di entrambi i mondi?
  - deliberativo
  - reattivo
- Approccio ovvio:
  - costruisce un agente composto da due (o più) sottosistemi:
    - uno **deliberativo** contenente un modello di mondo simbolico, che crea piani ed effettua decisioni nel modo proposto dall'IA simbolica
    - uno **reattivo** che è capace di reagire ad eventi senza ragionamenti complessi

# Agenti Ibridi

- migliore di entrambi i mondi ?
  - deliberativo
  - reattivo
- Architetture stratificate: ci sono molti componenti (reattivo, deliberativo, pianificazione, ecc.) che sono usati in un ordine dato predefinito
  - in parallelo: stratificazione orizzontale
  - in sequenza: stratificazione verticale

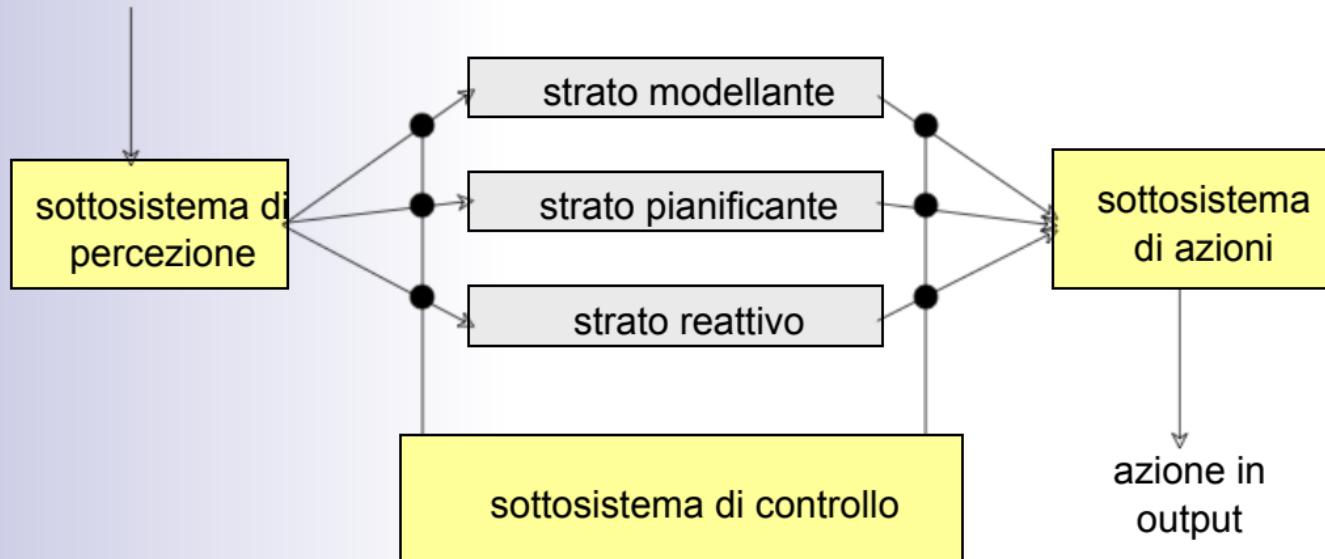
# Agenti Ibridi

- **stratificazione orizzontale**
  - Tutti gli strati sono connessi direttamente ai sensori di input e alle azioni di output; tutti gli strati processano l'input e contribuiscono all'output.
- **stratificazione verticale**
  - I sensori in input e le azioni in output sono entrambi forniti con al più uno strato ciascuno. In genere il sensore input arriva ad un certo strato e l'output di uno strato diventa l'input del successivo; l'ultimo strato produce un output.

# Agenti Ibridi: Stratificazione Orizzontale

input del sensore

Ferguson's TORINGMACHINES

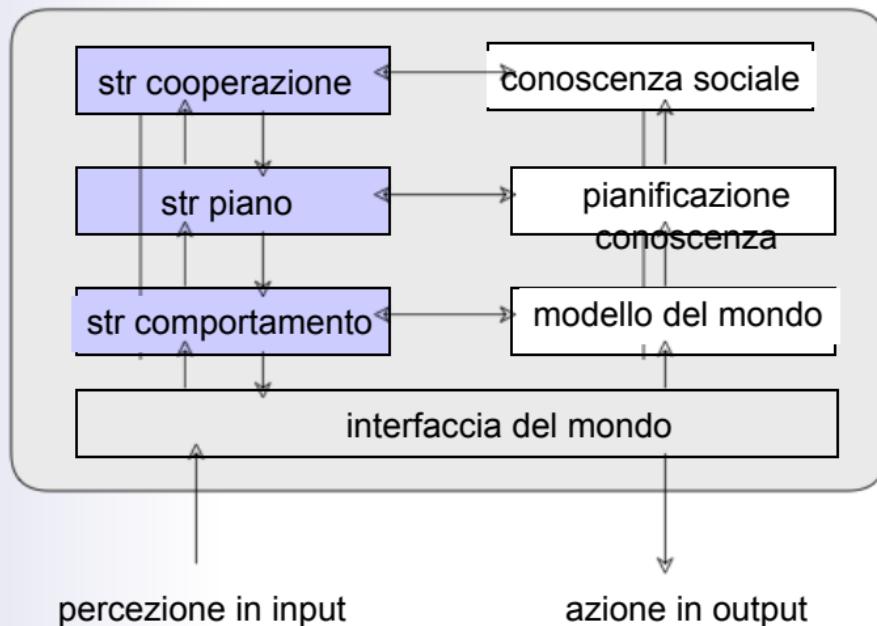


## Architettura a Stratificazione Orizzontale(cont.)

- vantaggi
  - concettualmente semplice
    - se ci serve che un agente esibisca  $n$  differenti tipi di comportamento, allora implementiamo  $n$  differenti strati!
- svantaggi
  - strati in competizione  nessuna garanzia su comportamenti coerenti ?
    - per evitare ciò: sottosistema di controllo
      - crea decisioni su quale strato ha il controllo ad un tempo dato
      - i creatori di agenti devono potenzialmente considerare tutte le interazioni tra strati!

# Architettura a Stratificazione Verticale: Müller –InteRRaP

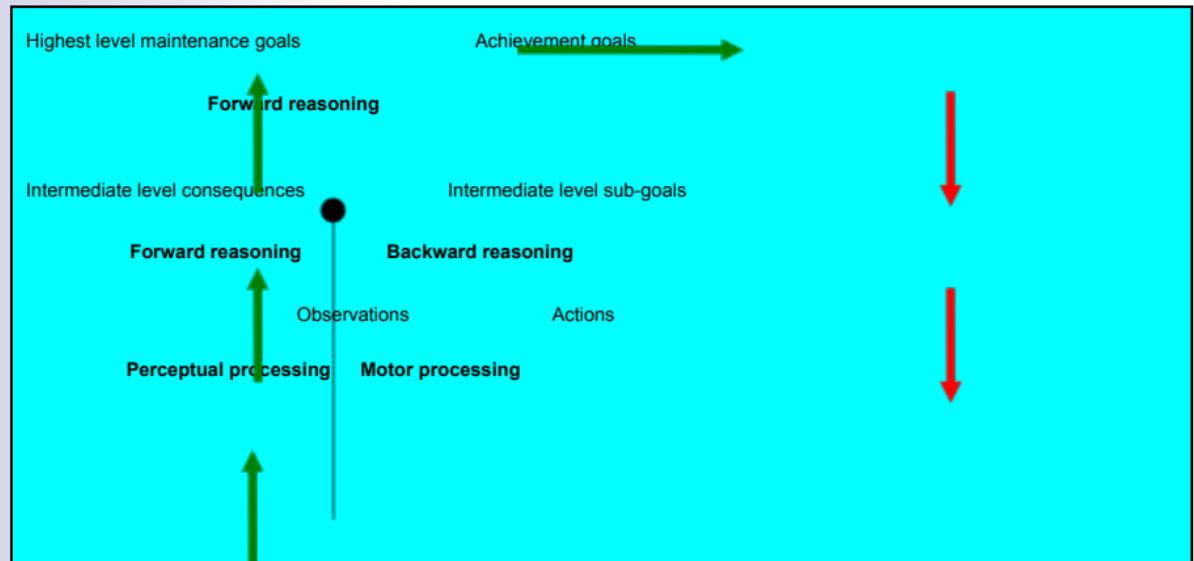
- stratificato in verticale, architettura a due sentieri



# Architettura a Stratificazione Verticale (cont.)

- vantaggi
  - complessità di interazioni tra gli strati ridotta
    - $n-1$  interfacce tra  $n$  strati
      - se ogni strato è in grado di suggerire  $m$  azioni al più  $m^2$  ( $n-1$ ) interazioni
      - molto più semplice del caso orizzontale ( $mn$  interazioni)
- svantaggi
  - questa semplicità ha un costo in flessibilità
    - il controllo deve passare tra ogni strato diverso non tollerante sugli errori:
      - errori in uno strato possono portare a serie conseguenze riguardo le prestazioni dell'agente

# Riconciliare razionalità e reattività: l'agente di Kowalski



Il mondo

# Riconciliare razionalità e reattività: l'agente di Kowalski

- Obiettivi di raggiungimento: obiettivi che l'agente intende ottenere per vari motivi.
- Obiettivi di mantenimento: obiettivi di cui l'agente necessita in modo da mantenere il paramentro denotante il suo stesso stato all'interno di un raggio accettabile.
  - La creazione di entrambi i tipi di obiettivi può essere determinata da (effetto indiretto di) stimoli esterni.
  - Ad ogni modo, gli obiettivi possono essere configurati in conseguenza dello stato presente della conoscenza dell'agente: un agente può iniziare un'attività “su sua iniziativa”, cioè, proattivamente.

# Architetture Logiche per Agenti

- Dal lavoro di Kowalski, ricerche sugli agenti logici hanno prodotto diversi approcci, tra cui
  - KGP
  - DALI