

# Intelligenza Artificiale II

Introduzione al corso

Dal calcolo simbolico  
al *natural computing*

Marco Piastra

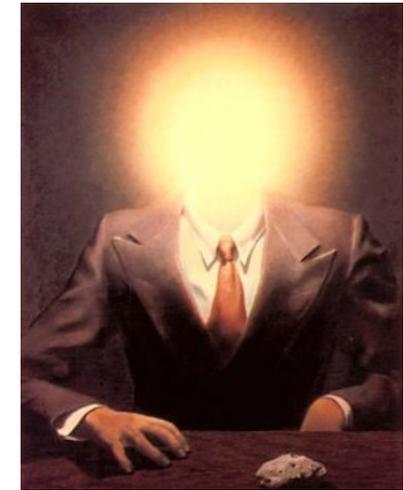
# “Artificial Intelligence”, prima apparizione

- John McCarthy et al., 1955

“We propose that a two-month, ten man study of **artificial intelligence** be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire.

The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of **intelligence** can in principle be *so precisely described* that a machine can be made to *simulate* it.

[...] It may be speculated that a large part of human thought consists of manipulating **words** according to **rules of reasoning** and **rules of conjecture**.



- Searle, J. R., *Minds, Brain and Science*, 1986

“Because we do not understand the brain very well we are constantly tempted to use the latest technology as a model for trying to understand it.

In my childhood we were always assured that the brain was a telephone switchboard (*'What else could it be?'*).

I was amused to see that Sherrington, the great British neuroscientist, thought that the brain worked like a telegraph system. Freud often compared the brain to hydraulic and electro-magnetic systems. Leibniz compared it to a mill, and I am told some of the ancient Greeks thought the brain functions like a catapult.

At present, obviously, the metaphor is the digital computer.”

- Crick, F., *The Astonishing Hypothesis*, 1994

“You, your joys and your sorrows, your memories and your ambitions, your sense of personal identity and free will, are in fact no more than the behavior of a vast assembly of nerve cell and their associated molecules.”

# Macchina di Turing

- Modello astratto di computer

Un nastro con celle elementari, un simbolo in ogni cella

Una testina di lettura e scrittura delle celle, può muoversi

Uno stato della macchina (un simbolo)

Una CPU, vale a dire una **tavola di transizione**

{ *<stato attuale, lettura cella>* → *<azione, nuovo stato>* }

(azione può essere *scrittura cella* o *movimento*)

- Generalità

Descrive in astratto la potenza di calcolo di un computer qualsiasi

Qualunque funzione sia calcolabile da un computer,  
è calcolabile da una macchina di Turing

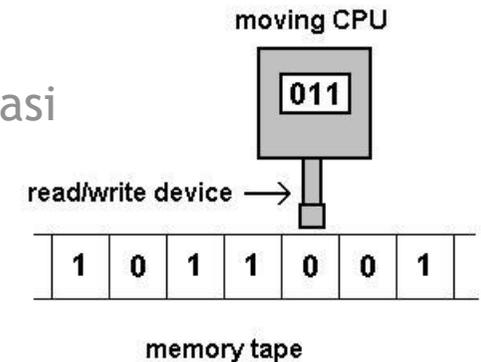
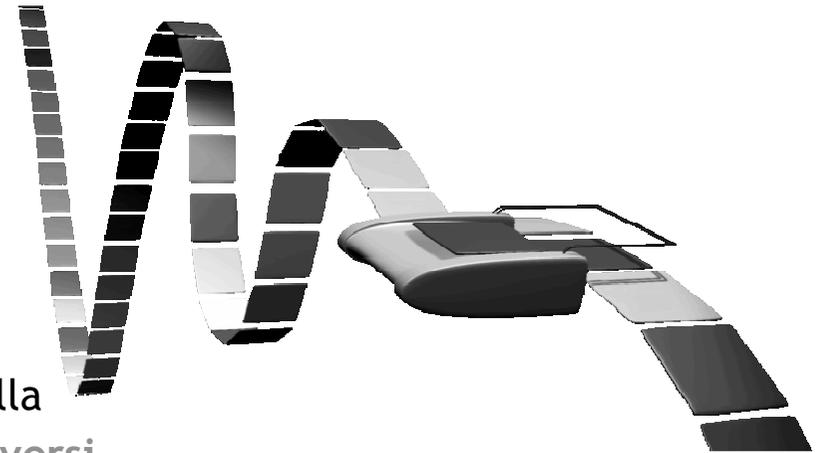
L'unica differenza rispetto alle realizzazioni è il nastro illimitato

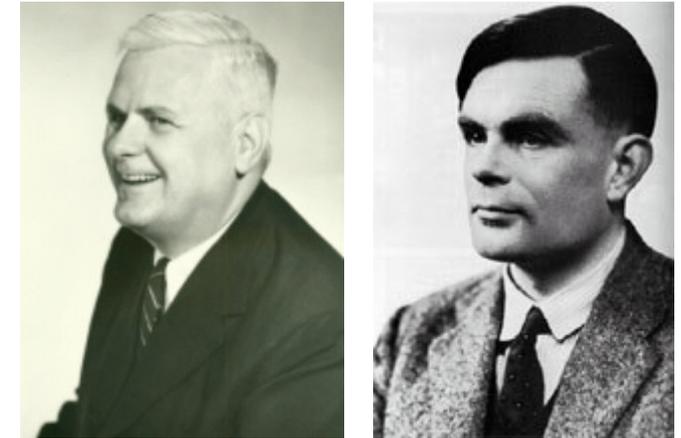
- Macchina di Turing Universale (*Universal Turing Machine*)

Versione estesa della macchina di Turing

La **tavola di transizione** viene caricata dal nastro, all'inizio

Può emulare qualsiasi macchina di Turing, data la **tavola di transizione**





# Tesi di Church-Turing

Non esiste un'unica, concisa formulazione originale:  
si tratta di un concetto espresso in più passaggi, da integrare con risultati teorici

- Una possibile formulazione (Wikipedia)

*“Every 'function which would naturally be regarded as computable' can be computed by a Turing machine.”*

La vaghezza della formulazione ha dato luogo a diverse interpretazioni,  
una molto comune (e non del tutto esatta) può essere espressa come (Wikipedia):

*“Every 'function that could be physically computed' can be computed by a Turing machine.”*

Searle: “... At present, obviously, the metaphor is the digital computer.”

# IA1: Logica classica

## ▪ Linguaggio logico + Struttura semantica

Linguaggio formale ( $L_{PO}$ )

Simboli di base: costanti, variabili, funzioni, predicati

Connettivi (+ parentesi e virgole)

Quantificatori (universale ed esistenziale)

Regole sintattiche (o di buona formazione), formule ben formate (*fbf*)

Struttura semantica  $\langle U, v \rangle$

Interpretazione composizionale (connettivi come operatori di un algebra di Boole)

Predicati come relazioni tra oggetti in  $U$

Funzioni in  $U$

Quantificatori come astrazioni su sottoinsiemi di  $U$  o di relazioni in  $U$

## ▪ Relazioni semantica tra formule

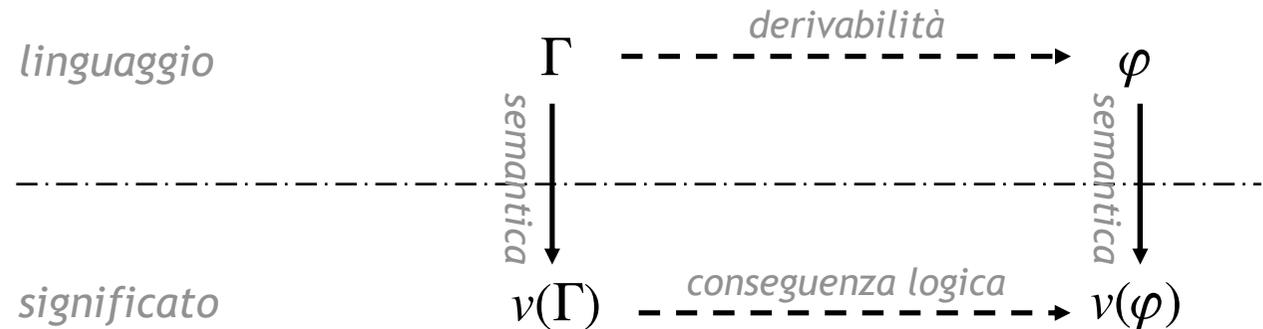
Conseguenza logica  $\Gamma \models \varphi$

## ▪ Calcolo simbolico (per trasformazione sintattica di formule)

Derivabilità  $\Gamma \vdash \varphi$

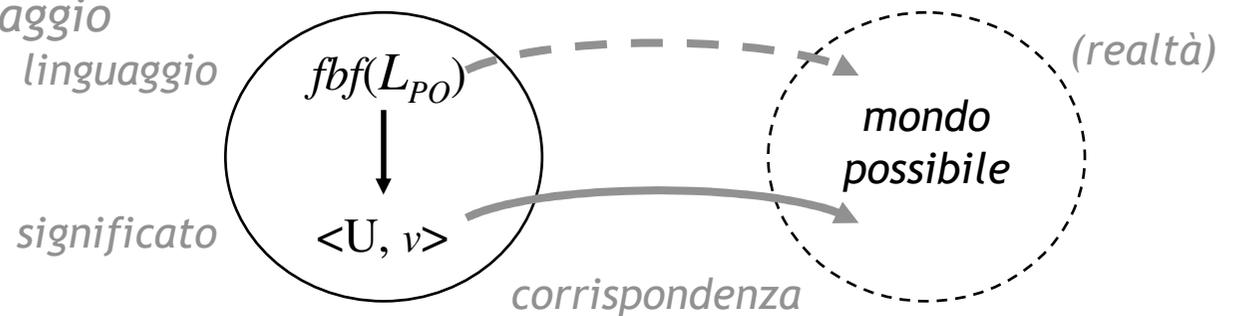
# IA1: Rappresentazione logico-formale

- Semantica come relazione tra linguaggio e significato



- La corrispondenza con la realtà è su entrambi i piani

*La corrispondenza del linguaggio (formale) passa attraverso il significato (formale)*



Molteplicità delle interpretazioni (mondi possibili):  
una fbf vera (in una teoria) corrisponde ad un *insieme* di mondi possibili

# IA1: Calcolo automatico

La derivabilità ( $\Gamma \vdash \varphi$ ) non presuppone l'esistenza di una procedura automatica ...

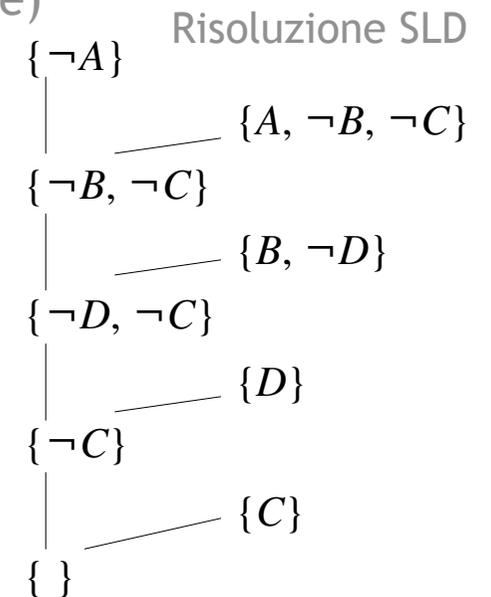
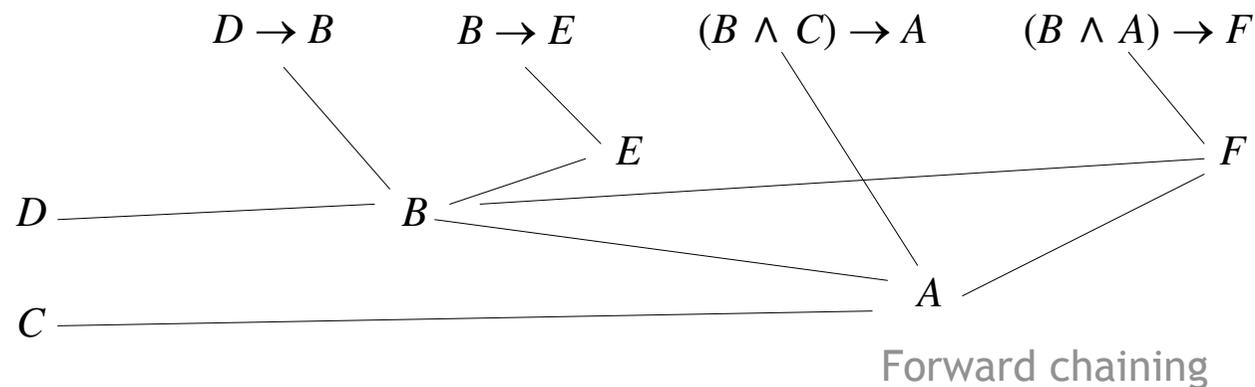
## ■ Decidibilità

Esiste una procedura effettiva per stabilire se  $\Gamma \vdash \varphi$ ?  
(oppure se  $\Gamma \models \varphi$ ?)

## ■ Metodi di calcolo

(fbf qualsiasi) Risoluzione per refutazione, Tableau

(clausole di Horn) Risoluzione SLD, Forward Chaining (+ alg. Rete)



# IA1: Agenti

- Agente razionale

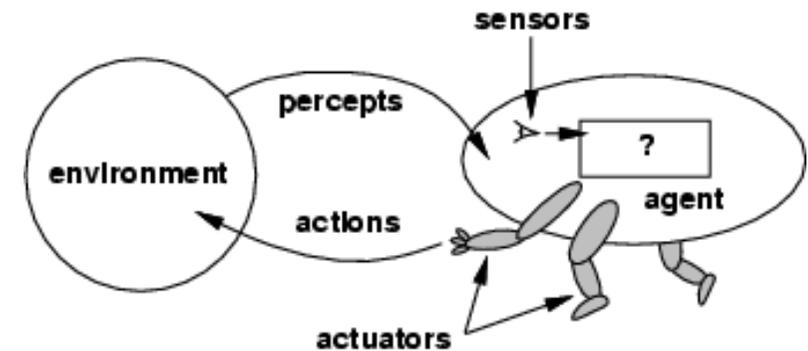
Un sistema come che interagisce con l'ambiente

Elaborazione ciclica (sequenziale o parallela)  $f: P^* \rightarrow A$

Percezione (attraverso i sensori,  $P^*$ )

Valutazione (processi interni  $f: P^* \rightarrow A$ )

Azione (attraverso gli attuatori,  $A$ )



- Tank in JessRobocode

Un programma Java/Jess che interagisce con l'ambiente Robocode

Elaborazione ciclica (sequenziale)  $f: P^* \rightarrow A$

Percezione (eventi prodotti in Robocode,  $P^*$ )

Valutazione (regole logiche del programma Jess  $f: P^* \rightarrow A$ )

Azione (comandi di Robocode,  $A$ )

Parallelismo: molteplici agenti (almeno 2 tank) in ciascun round

# Programmazione logica

Il Jess è un sistema relativamente semplice e facile da usare  
Dal punto di vista del calcolo (logico) automatico, si può fare di più ...

- Dalla risoluzione SLD in  $L_P$  alla programmazione logica in tre passaggi
  - 1) Clausole di Horn (in  $L_{PO}$ ) con funzioni e quantificazione esistenziale (implicita)
  - 2) Risoluzione per refutazione ed unificazione
  - 3) Dalla dimostrazione ( $\Gamma \vdash \varphi$ ) all'identificazione delle possibili istanziazioni (p.es. *Sorella(amelia, x) ? = "di chi è sorella Amelia?"*)

Si ottiene

La possibilità di derivare nuove clausole qualsiasi (anche regole, non solo fatti)

Un linguaggio di programmazione completo (Prolog)

(con qualche ulteriore "stranezza", p.es. *negation as failure*)

## Oltre la logica classica

L'uso di *assert* e *retract* in Jess esula dal paradigma della logica classica, in quanto viene meno la proprietà di *monotonia sintattica*

In programmazione logica si hanno ulteriori eccezioni

- Ipotesi del mondo aperto (logica classica)

Dati  $\Gamma$  e  $\varphi$ , in logica classica si hanno tre possibili casi

$$\Gamma \models \varphi$$

$$\Gamma \models \neg\varphi$$

$$\Gamma \not\models \varphi, \Gamma \not\models \neg\varphi \text{ (il valore di } \varphi \text{ non è determinato da } \Gamma \text{)}$$

- Ipotesi del mondo chiuso (*closed world assumption, CWA*)

$$\Gamma \not\models \varphi \Rightarrow \neg\varphi \text{ (assumo false le fbf } \varphi \text{ il cui valore non è determinato da } \Gamma \text{)}$$

- Negazione per fallimento (*negation as failure, NAF*)

$$\Gamma \not\vdash_P \varphi \Rightarrow \neg\varphi \text{ (assumo false le fbf } \varphi \text{ che } \underline{\text{non riesco a dimostrare}}, \text{ con una procedura } P \text{)}$$

- *CWA* e *NAF* non sono inferenze corrette, ma solo *plausibili*

# Inferenza plausibile (*defeasible reasoning*)

Le inferenze *plausibili* non sono *corrette* ma solo *compatibili* con le premesse

## ■ Nuovi schemi di ragionamento

### *Logica deduttiva* (inferenza corretta)

*Si derivano le conseguenze logiche delle premesse*

“Tutti i fagioli del sacco sono bianchi,  
questa manciata di fagioli viene dal sacco”  
“I fagioli sono bianchi”

$$\begin{array}{l} \text{modus} \\ \text{ponens} \end{array} \quad \begin{array}{l} \varphi \rightarrow \psi \\ \varphi \\ \hline \psi \end{array}$$

### *Logica induttiva* (inferenza plausibile)

*Date associazioni di fatti, si derivano regole*

“Questa manciata di fagioli viene dal sacco,  
i fagioli sono bianchi”  
“Tutti i fagioli del sacco sono bianchi”

$$\begin{array}{l} \varphi \\ \psi \\ \hline \varphi \rightarrow \psi \end{array}$$

### *Logica abduttiva* (inferenza plausibile)

*Dati regole ed effetti si derivano le cause*

“Tutti i fagioli del sacco sono bianchi,  
i fagioli sono bianchi”  
“Questa manciata di fagioli viene dal sacco”

$$\begin{array}{l} \varphi \rightarrow \psi \\ \psi \\ \hline \varphi \end{array}$$

*Attenzione!*  
*Non è il*  
*modus ponens*

# Logiche modali e temporali, *model checking*

## ▪ Logica modale

In  $L_p$  i simboli proposizionali rappresentano affermazioni

Una notazione per distinguere le affermazioni

$A$  = affermazione vera (p.es. "Oggi è venerdì")

$\Box A$  = affermazione vera secondo  $\Box$  (p.es. " $\Box$  ritiene che oggi sia venerdì")

Formule e regole di inferenza 'ad hoc'

$A \vdash \Box A$  Se  $A$  è vera, allora  $\Box$  ritiene vera  $A$

$\Box (A \rightarrow B), A \vdash \Box B$  Se  $\Box$  ritiene vera  $(A \rightarrow B)$  e  $A$  è vera, allora  $\Box$  ritiene vera  $B$

$\Box (A \rightarrow B), A \not\vdash B$  Dalle stesse premesse non è derivabile che  $B$  è vera

## ▪ Logica temporale

Notazione e regole per il ragionamento sulle sequenze temporali

$\Box (A \rightarrow B)$  D'ora in poi, se  $A$  allora  $B$

$A \rightarrow \Box B$  Se  $A$  allora d'ora in poi  $B$

## ▪ *Model checking* per le logiche temporali

Verifica diretta (per generazione di modelli/sequenze) di insoddisfacibilità

Serve a verificare le proprietà dei modelli di processo

# Multivalenza, logiche sfumate (*fuzzy logics*)

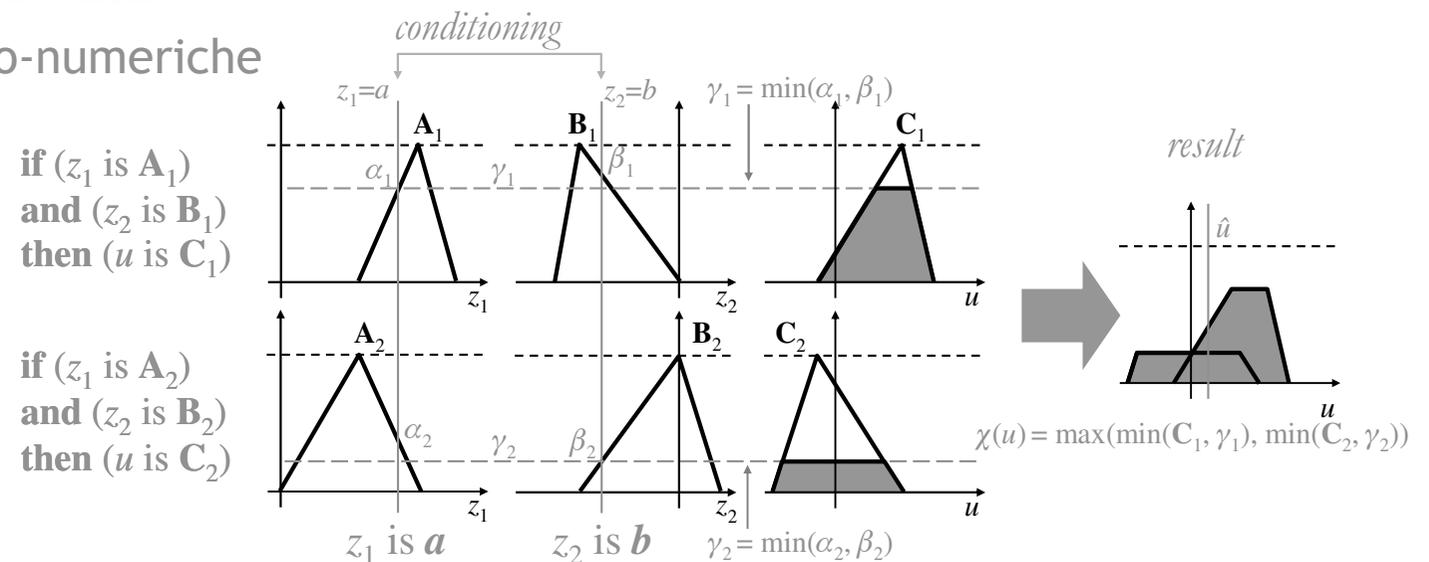
- Forme di logica su base numerica

Si ammettono valori compresi tra 0 e 1 (intermedi, tra falsità e verità) mantenendo l'idea di associare operatori algebrici ai connettivi

Ad esempio:  $v(A \wedge B) = \min(v(A), v(B))$

- Sistemi di inferenza sfumata

Insiemi di regole logico-numeriche



*Universal Approximators* (L.X. Wang, 1992)

Possono approssimare qualsiasi funzione continua

Costruzione modulare, combinazione per interpolazione

# Incertezza, probabilità, modelli grafici

## ▪ Incertezza e probabilità

La probabilità è una rappresentazione numerica dell'incertezza

Per sua natura, è sistemica: si riferisce ad un intero sistema di eventi

Non è vero-funzionale: non esiste una funzione  $f$  tale per cui  $P(A \wedge B) = f(P(A), P(B))$

## ▪ Distribuzioni (discrete) e densità (continue) di probabilità

La misura di probabilità si riferisce a tutte le possibili combinazioni di eventi

Dalla misura congiunta è possibile calcolare la misura di eventi particolari (marginalizzazione) o sotto determinate ipotesi (condizionalizzazione)

Una misura di probabilità si può stimare partendo dai dati (inferenza statistica)

Nel discreto, la complessità numerica è notevole

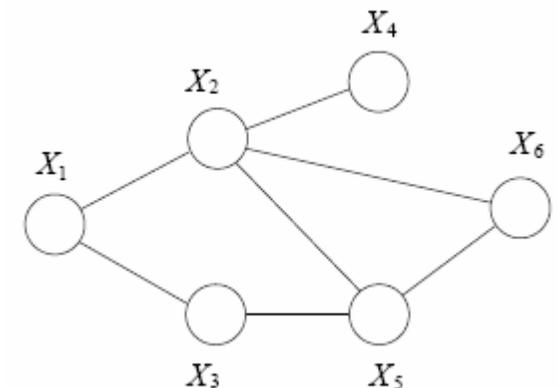
Nel continuo si aggiunge anche la complessità analitica  
(spesso il risultato non è esprimibile in forma analitica)

## ▪ Modelli grafici

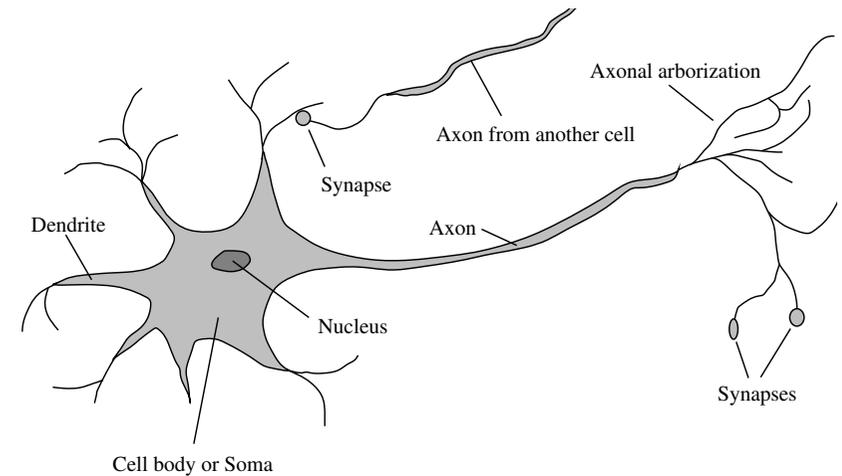
Rappresentazione strutturata di misure di probabilità

La struttura (grafo) descrive dipendenze tra eventi, i valori numerici completano la definizione della misura

I modelli grafici, in generale, sono più efficacemente trattabili



# IA1: Reti neurali artificiali



- Elaborazione numerica

Si basa sulla propagazione dei segnali attraverso la rete, a partire dalle unità di input

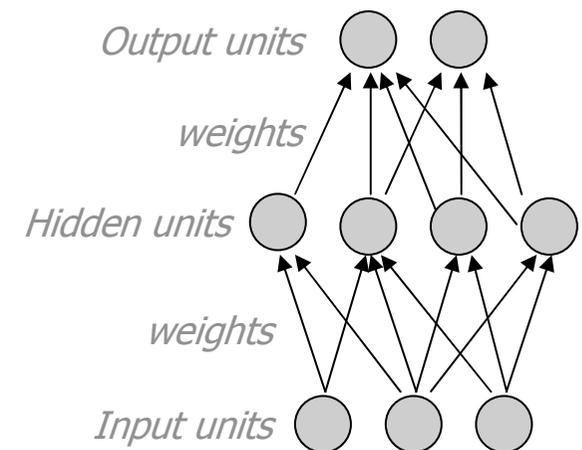
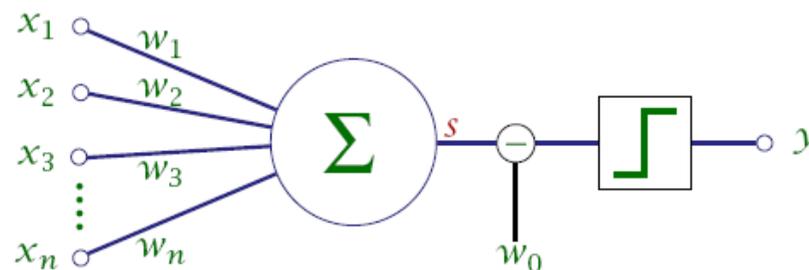
- Apprendimento supervisionato (*backpropagation*)

Le reti apprendono le configurazioni dei pesi a partire da associazioni input/output note a priori

*Universal Approximators* (Hornik et al., 1989)

Possono approssimare qualsiasi funzione continua

Basta un solo livello di *hidden units*



## Proprietà *emergenti* (*self-organization*)

- [*Emergence as a Construct: History and Issues*, Goldstein, 1999]

“The arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems”

“*Radical novelty*: emergents have features that are not previously observed in the complex system under observation. This novelty is the source of the claim that features of emergents are neither predictable nor deducible from lower or micro-level components.[...]”

“*Coherence or correlation*: emergents appear as integrated wholes that tend to maintain some sense of identity over time. This coherence spans and correlates the separate lower-level components into a higher-level unity.”

“*Global or macro level*: since coherence represents a correlation that spans separate components, the locus of emergent phenomena occurs at a global or macro level, in contrast to the micro-level locus of their components. Observation of emergents, therefore, is of their behavior on this macro level.”

“*Dynamical*: emergent phenomena are not pre-given wholes but arise as a complex system evolves over time. [...]”

“*Ostensive*: emergents are recognized by showing themselves, i.e., they are ostensively recognized. [...] Because of the nature of complex systems, each ostensive showing of emergent phenomena will be different to some degree from previous one

## *Self-organizing maps (SOM)* Kohonen, T., 1995

- Struttura (tipica) a due livelli

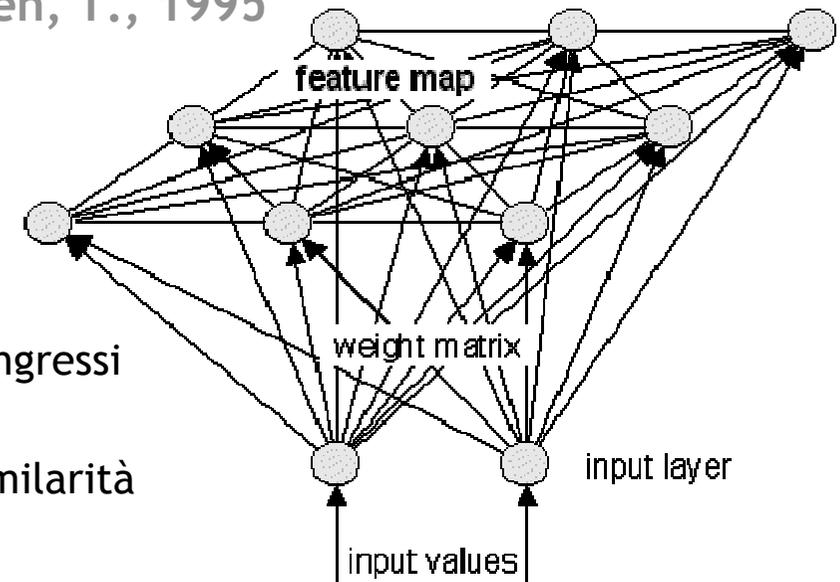
Livello di mappa, con unità organizzate secondo una topologia prestabilita

Livello di input, o ingressi

Il livello di mappa è completamente connesso agli ingressi

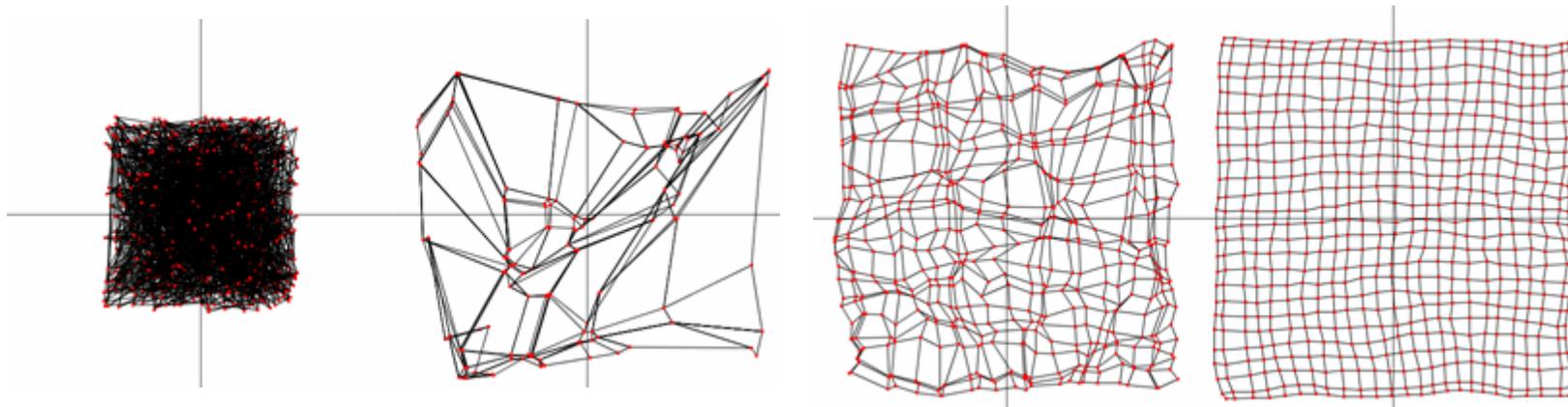
Le connessioni hanno un peso associato

Gli input modificano i pesi secondo un criterio di similarità



- Adattamento (apprendimento?) non supervisionato

Le SOM si adattano progressivamente alla topologia del segnale in input



# *Self-organizing maps* ed altre strutture adattative

- Demo

<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/SOM.html>

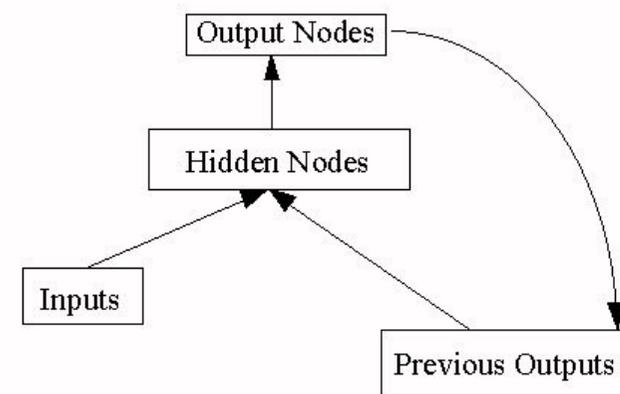
<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html>

## Reti neurali ricorsive (*feedback*)

- Le reti neurali ricorsive sono macchine da calcolo complete

Il feedback include unità di memoria nel tempo  
(il requisito minimo è avere delle *delay unit*)

Sono traducibili in macchine di Turing  
(Hyötyniemi, H., 1996)



Non si conosce un metodo generale di apprendimento  
per le reti neurali ricorsive

# Automati cellulari (*cellular automaton, CA*)

- Sistemi distribuiti e di calcolo parallelo

In ogni istante, ciascuna **cella** si trova in un determinato **stato**

Sono dati  $N$  stati possibili (minimo 2, ovviamente)

Ogni cella è connessa ad un certo numero di **vicini** (*neighbors*)

La **transizione di stato** dipende dallo stato attuale e lo stato dei vicini

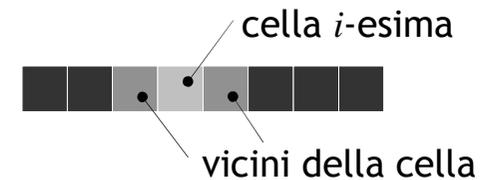
Il processo è definito da regole

Ciascuna cella opera in parallelo (in modalità sincrona)

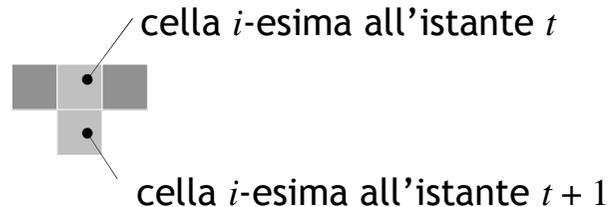
- Esempio

Un vettore di celle (automa unidimensionale)

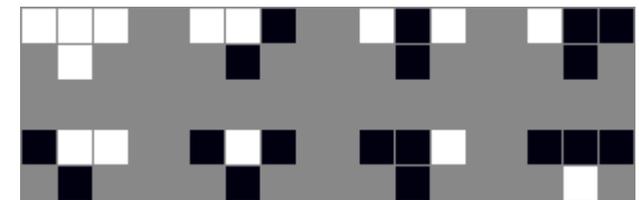
2 possibili stati, 8 possibili regole di transizione



Regola  
di transizione



8 regole possibili ( $N = 2$ )



- Demo

<http://math.hws.edu/xJava/CA/>

# Automi cellulari come macchine di Turing

- Gli automi cellulari sono modelli di calcolo

Data una configurazione iniziale ed un'opportuno set di regole, un CA unidimensionale può calcolare qualsiasi funzione

Esempio (Wolfram, 2002)

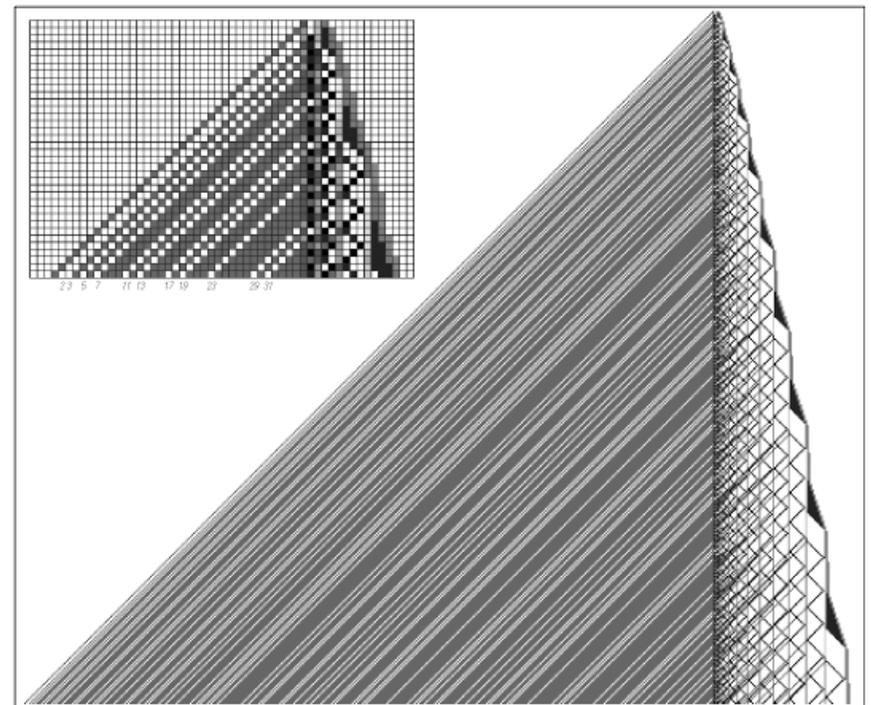
Il sistema in figura calcola i numeri primi

Regione triangolare di sinistra

Le caselle grigie sono multipli di qualche  $n > 2$

I 'buchi' bianchi sono numeri primi

Il 'risultato' si sposta progressivamente a sinistra



- *Universal Cellular Automaton* (Wolfram, 2002)

Un automa cellulare che può simulare qualsiasi altro automa cellulare

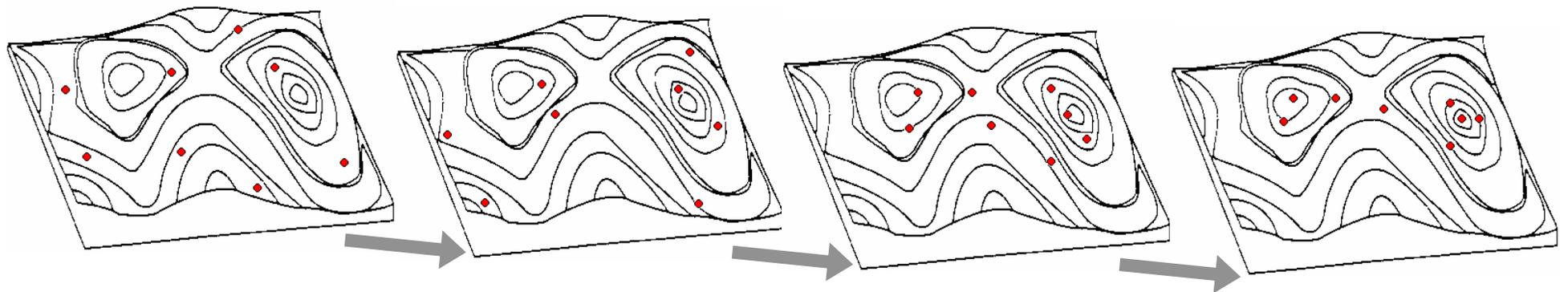
# Altre forme di sistemi adattativi distribuiti

- Demo

<http://www.red3d.com/cwr/boids/>

# Calcolo evolutivo

- Idea intuitiva: evoluzione di una popolazione
  - Una funzione obiettivo (*fitness*) da ottimizzare
  - Una popolazione di campioni (punti nel dominio della funzione)
  - Operatori di ricampionamento: nuovi campioni da uno (o più) campioni esistenti
  - Selezione casuale della popolazione guidata dalla *fitness*:
    - Si scelgono campioni a caso, *preferendo* i campioni a *fitness* più elevata
    - I campioni prescelti vengono ricampionati e costituiscono una nuova generazione (la prima generazione è generata in maniera completamente casuale)
- Gradualmente, la popolazione migra verso i massimi della *fitness*



# GA, EP e GP

Fondamentale è l'idea che il ricampionamento utilizzi **parti** della **rappresentazione** del campione selezionato (come fosse il DNA di un campione ..)

- **Genetic Algorithms (GA)**

Dominio: vettori di numeri

Ricampionamento: mutazione di una cella, ricombinazione di due vettori

Applicazione tipica: insiemi di parametri, associazioni, regole

- **Evolutionary Programming (GA)**

Dominio: grafi

Ricampionamento: modifica di una grafo, ricombinazione di due grafi

Applicazione: automi a stati finiti, reti neurali, circuiti

- **Genetic Programming (GP)**

Dominio: strutture ad albero

Ricampionamento: mutazione per ricostruzione, ricombinazione di due alberi

Applicazione: programmi, espressioni analitiche

# DNA come macchina di Turing

Gli sviluppi della biochimica moderna permettono di operare in modo molto efficace sulle sequenze di DNA  
La scala nanometrica dei processi consente di operare ad elevatissimo parallelismo e basso consumo di energia

## Operazioni sulle sequenze di DNA

### Sintesi

Produzione di *oligonucleotidi* (brevi sequenze) con sequenza prestabilita

### Denaturazione e rinaturazione (*annealing*)

*Separazione in due filamenti complementari e ricombinazione*

### Connessione e taglio

### Amplificazione

La *Polimerase Chain Reaction (PCR)* permette di copiare sequenze. Induce crescita *esponenziale*

### Identificazione

Tramite elettroforesi o ibridazione è possibile identificare la presenza di sequenze

## DNA Computing

*L. Adelman, 1994 ha per primo utilizzato il DNA computing per risolvere un problema di cammino minimo sui grafi. Un noto algoritmo si applica al problema SAT*

In generale, l'uso delle operazioni sul DNA per il calcolo automatico

