

SCUOLA NAVALE MILITARE

“F. MOROSINI”

A.S. 2007/2008



**“Robotica e Intelligenza Artificiale, un futuro
carico di incertezze”**

Allievo III Corso Matr. 602

Giovanni Pecoraro

V Liceo Scientifico Sez. B

Scuola Navale Militare Francesco Morosini

Anno scolastico 2007/2008

Robotica e Intelligenza Artificiale, un futuro carico di incertezze




Giovanni Pecoraro Matr. 602

Classe V B



A tutti coloro
che hanno dedicato la propria vita alla scienza
nella speranza di creare un futuro migliore per l'umanità



“A questo modo noi possiamo definire e stabilire che cosa significa la parola *ragione* quando la consideriamo fra le facoltà dello spirito. Poiché *ragione* in questo senso significa nient'altro che calcolo, cioè addizione e sottrazione, delle conseguenze dei nomi generali usati per convenzione come *annotazioni* e come *espressioni* dei nostri pensieri; io li chiamo *annotazioni* quando il detto calcolo noi lo facciamo mentalmente, li chiamo *espressioni* quando noi dimostriamo e proviamo i nostri calcoli mentali agli altri uomini.”

(Thomas Hobbes, *Leviatano*, I, cap. V)

Premessa

Nel realizzare questo elaborato in vista del mio esame di maturità ho tentato di infondere nelle poche righe la mia innata passione per la scienza. Da tempo ho sviluppato l'idea che qualsiasi fenomeno legato alla natura sia riconducibile ad un insieme di leggi fisico-matematiche. Ricordando le parole del filosofo inglese Thomas Hobbes affermo con estrema convinzione che allo stesso modo ogni processo che coinvolge la ragione umana è razionalizzabile in una specifica equazione. Persino un sentimento astratto come l'amore, la vera essenza dell'irrazionalità, può essere ricondotto ad una formula fisica basata su un linguaggio matematico a tal punto sopraffino e complicato da essere incomprensibile all'uomo moderno. Noi siamo ancora troppo legati ai classici modelli matematici, che risalgono ai tempi delle civiltà mesopotamiche. Per quanto possiamo svilupparli, questi rappresentano pur sempre un limite.

Magari un giorno un novello Gauss, privato di ogni insegnamento scolastico, riuscirà autonomamente a sviluppare fin da bambino una nuova e più libera matematica, che ci permetterà di risolvere dilemmi che attualmente sembrano insolubili.

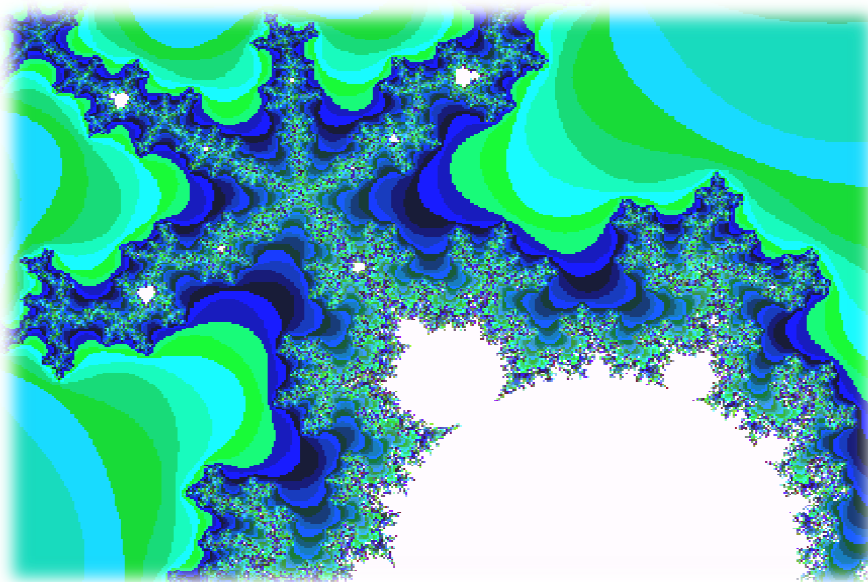
Attraverso una prima parte esclusivamente teorica ed una seconda applicativa ho provato a riassumere quelli che sono stati gli enormi, ma purtroppo ancora limitati, sviluppi della Robotica e dell'Intelligenza Artificiale nell'ultimo secolo. Sono queste le scienze del futuro, quelle su cui è opportuno investire tempo e denaro, tenendo sempre sott'occhio il lato perverso della mente umana, una realtà così meravigliosa, ma allo stesso tempo capace di trasformare la bellezza innata della scienza in uno strumento di morte e distruzione.

Giovanni Pecoraro

Sommario

Robotica e Intelligenza Artificiale, un futuro carico di incertezze	8
1. Che cos'è la Robotica?	8
1.1 La Robotica come scienza del futuro	8
1.2 Cosa sono i Robot?	10
1.3 Breve storia dei Robot	11
2. Intelligenza artificiale	14
2.1 Definizione	14
2.2 Brevi cenni storici	15
2.3 Fondamenti teorici dell'IA	16
2.3.1 Il concetto di simbolo	17
2.3.2 Il calcolo come manipolazione di simboli	19
2.4 La macchina e il test di Turing	21
2.5 I primi programmi	25
2.6 L'analisi del linguaggio naturale	29
2.6.1 Le grammatiche di Chomsky e l'analisi automatica del linguaggio naturale	29
2.6.2 Il problema del significato	31
2.6.3 I micromondi e SHRDLU	33
2.6.4 Le reti semantiche	35
2.6.5 I frame di Minsky	38
2.6.6 Gli script di Schank ed Abelson	40
2.7 Problem Solving	42
2.8 I sistemi esperti	45
2.8.1 Creazione di un sistema esperto	47
2.9 Le reti neurali	50
2.9.1 Limiti nello sviluppo di algoritmi	50
2.9.2 Il cervello umano	50
2.9.3 Le reti neurali artificiali	51
2.9.4 Caratteristiche di una rete neurale	52
2.9.5 Neuroni artificiali	53
2.9.5.1 Reti di Hopfield	58
2.9.5.2 Reti feed-forward multistrato	59
2.9.5.3 Reti ricorrenti	61
2.9.5.4 Reti competitive	62
2.9.5.5 Apprendimento	63
2.9.5.5.1 Apprendimento supervisionato	63
2.9.5.5.2 Apprendimento non supervisionato	66
2.10 Algoritmi genetici	67
2.10.1 Introduzione	67
2.10.2 Principi basilari	68
3. Studio del PICAXE Micro-Robot Line Follower	71
3.1 Struttura di base	72
3.2 Elementi di cinematica	74
3.3 Diagramma del circuito	77
3.4 Funzionamento dei motori elettrici	78

3.5 Sensori	81
3.6 Microcontrollore PICAXE	82
3.7 Programmazione del Micro-Robot	84
4. Il futuro della Robotica	88
4.1 Robotica biomedicale	88
4.2 Robotica marina	88
4.3 Robotica militare	89
4.4 Robotica spaziale	90
4.5 Robotica Industriale	91
5.APPENDICE: Genetic Algorithms	94
5.1 Basic principles	94
5.2 Representation of a problem	96
5.3 Complete process of a genetic algorithm	96
5.4 Fitness function	96
5.5 Proportional selection	97
5.6 Crossover	98
6. Contenuto del CD	99
BIBLIOGRAFIA	99
SITOGRAFIA	101



Robotica e Intelligenza Artificiale, un futuro carico di incertezze

1. Che cos'è la Robotica?

La Robotica è la scienza che si occupa dello sviluppo di macchine, definite robot, in grado di eseguire compiti specifici e di interagire con l'ambiente esterno. Essa nasce dai sogni più antichi e profondi dell'uomo, di creare la macchina più complessa esistente, ovvero di realizzare un organismo artificiale e autonomo dotato di un'intelligenza artificiale.

La Robotica si basa su studi interdisciplinari articolati tra:

- Meccanica
- Elettronica
- Controlli automatici
- Informatica
- Sensoristica
- Motoristica
- Scienze dei materiali
- Intelligenza Artificiale

Unita da forti legami con:

- Fisica/Matematica
- Logica/Linguistica
- Neuroscienze/Psicologia
- Biologia/Fisiologia
- Antropologia/Filosofia
- Arte/Design Industriale

1.1 La Robotica come scienza del futuro

La volontà dell'uomo di incrementare la propria potenza ha cambiato continuamente il suo modo di porsi nei confronti della scienza, orientandolo verso attività legate allo sviluppo economico e al superamento dei suoi limiti nell'affrontare situazioni in ambienti particolarmente ostili o addirittura inaccessibili. Questo fattore, unito ai continui progressi compiuti nei campi dell'informatica e delle telecomunicazioni ha contribuito a determinare una nuova visione della

realtà, sempre più legata alla realizzazione di macchine intelligenti. La stessa esplorazione dell'Universo sarebbe impossibile senza una proficua collaborazione tra una componente biologica ed una robotica, ma è solo l'inizio. L'infinita varietà di campi in cui sarebbe possibile attuarla con successo è ben più estesa e contribuisce a rendere la scienza dei robot il vero settore su cui puntare per il futuro. Analizzando gli ingenti investimenti compiuti dai Paesi industrializzati nelle attività di ricerca, si può comprendere che il mercato, in particolare quello asiatico, grazie all'opera del Giappone, si sta preparando ad accogliere questi esemplari della più alta tecnologia moderna.

Le principali motivazioni dell'utilizzo dei robot sono:

- Eliminazione dei lavori faticosi e rischiosi per l'uomo
- Aumento della produttività e riduzione dei costi
- Miglioramento della qualità dei prodotti
- Ottimizzazione dei processi produttivi

Country	Yearly installations				Operational stock at year-end			
	2002	2003	2004	2007	2002	2003	2004	2007
Japan	25,373	31,588	33,200	41,300	350,169	348,734	352,200	349,400
United States	9,955	12,693	12,800	15,900	103,515	112,390	121,300	145,100
European Union	26,096	27,114	28,800	34,400	233,769	249,200	266,100	325,900
Germany	11,882	13,381	14,100	16,300	105,212	112,693	121,500	151,400
Italy	5,170	5,198	5,500	6,100	46,881	50,043	53,100	63,100
France	3,012	3,117	3,300	3,900	24,277	26,137	28,100	35,900
United Kingdom	750	1,111	1,200	1,500	13,651	14,015	14,600	16,300
Austria a/	670	385			3,521	3,602		
Benelux a/	654	715			8,708	9,052		
Denmark	249	288			1,853	2,078		
Finland	376	387			3,151	3,407		
Portugal	138	135			1,282	1,367		
Spain	2,420	2,031			18,352	19,847		
Sweden	495	386			6,881	6,959		
Other Europe	582	922	1,000	1,300	11,009	11,409	11,900	14,200
Czech Rep. a/	87	498			1,022	1,445		
Hungary	61	35			211	216		
Norway	80	48			664	684		
Poland	128	60			622	584		
Russian Fed. a/	21	9			5,000	5,000		
Slovakia b/	24	1						
Slovenia b/	25	31						
Switzerland a/	156	240			3,490	3,480		
Asia/Australia	5,123	6,695	7,200	8,900	60,427	65,419	69,900	78,500
Australia	392	533			3,192	3,571		
Rep. of Korea (all types of industrial robots)	3,998	4,660			44,265	47,845		
Singapore a/	53	48			5,299	5,273		
Taiwan, Province of China	680	1,454			7,671	8,730		
Other countries a/	1,466	2,764	3,200	4,500	11,216	13,620	16,500	27,200
Subtotal, excl. Japan and Rep. of Korea	39,224	45,528	47,900	58,700	375,671	404,193	485,700	590,900
Total, including Japan and Rep. of Korea	68,595	81,776	86,200	106,300	770,105	800,772	886,200	997,700

Tabella 1. Vendite del mercato internazionale della Robotica

1.2 Cosa sono i Robot?

Il termine “ROBOT” deriva dalla parola ceca “robota”(lavoro pesante), il suo primo utilizzo lo si deve Karel Čapek che nel 1920 lo utilizzò per la prima volta nella sua opera teatrale “I robot universali di Rossum” . In verità il vero ideatore del termine era stato il fratello Josef, che lo aveva adoperato la parola *automat* (automa) in un suo racconto intitolato “Opilec”(“L’ubriacone”, 1917). Senza alcun dubbio il concetto espresso in quel periodo era molto differente da quello inteso nella società contemporanea.

L’Enciclopedia Britannica fornisce questa definizione:

"A robot device is an instrumented mechanism used in science or industry to take the place of a human being. It may or may not physically resemble a human or perform its tasks in a human way, and the line separating robot devices from merely automated machinery is not always easy to define. In general, the more sophisticated and individualized the machine, the more likely it is to be classed as a robot device."

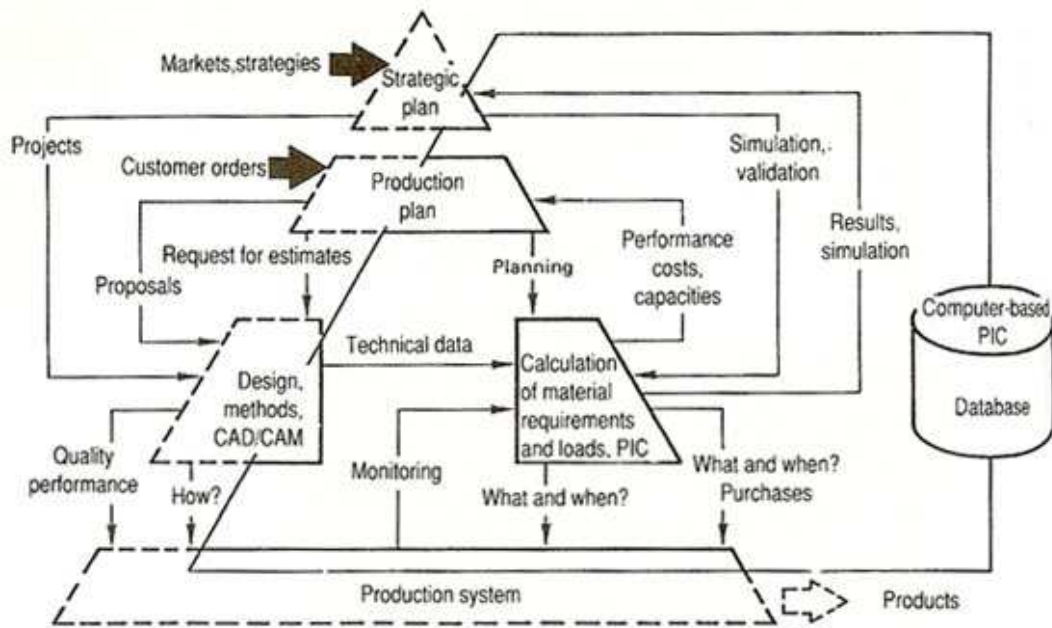
Ovvero:

"Un robot è uno strumento utilizzato nella scienza e nell’industria per prendere il posto di un essere umano. Esso potrebbe, oppure no, assomigliare ad un essere umano e svolgere i suoi compiti come un uomo; la linea che separa i robot da una macchina puramente automatizzata non è sempre facile da definire. In generale, la più sofisticata e specifica macchina più simile a ciò deve essere classificata come un robot."

1.3 Breve storia dei Robot

Dal punto di vista storico lo sviluppo di sistemi automatizzati risale ai tempi antichi, infatti già tra il 400-350 a.C. il greco Archita di Taranto costruì un piccione messo in moto da un getto di vapore. Intorno al 100 d.C. Erone di Alessandria progettò macchine azionate da acqua e pesi in caduta e circa quattrocento anni dopo i Bizantini eressero un'immensa statua di Ercole che racchiudeva un innovativo orologio idraulico. Occorse aspettare il '200 per vedere la nascita dei primi esemplari seppur rudimentali di androidi: Roger Bacon creò una testa parlante, quasi contemporaneamente al uomo di ferro di Albertus Magnus. Una svolta significativa, che diede all'uomo l'opportunità di realizzare i primi dispositivi automatizzati, si ebbe verso la fine del XVII secolo con l'invenzione della macchina a vapore da parte di James Watt. Il successo fu immediato, soprattutto in campo industriale: non era più necessaria la presenza di corsi d'acqua o di vento per produrre lavoro meccanico, infatti la semplice espansione di vapore acqueo surriscaldato azionava uno stantuffo che produceva la potenza necessaria. Nei primi anni del Novecento invece l'introduzione dell'uso dei derivati del petrolio contribuì allo sviluppo di nuove tipologie di macchine automatiche, dotate del cosiddetto motore a scoppio, in quanto l'energia derivava da reazioni esplosive.

La nascita della robotica in questo secolo a partire dagli anni '40; i robot inizialmente destinati solo ad un uso industriale, con lo sviluppo graduale di teorie per il controllo del movimento e l'incremento dei gradi di libertà acquisiscono una sempre maggiore complessità fino alla costruzione dei cosiddetti bracci manipolatori che hanno invaso il settore produttivo mondiale. A partire dagli anni '70 il Giappone, pur essendo partito in ritardo, crea un enorme mercato interno basato sui robot, influenzando in maniera decisiva sullo sviluppo del settore a livello mondiale. È proprio in questi anni che inizia a svilupparsi in parallelo il campo dell'informatica, in particolare con i sistemi operativi e le reti per la gestione dell'automazione: l'immediata conseguenza è lo sviluppo di nuove modalità di produzione come il CIM (Computer Integrated Manufacturing). Il CIM rappresenta il modello della fabbrica automatizzata, basata su una fitta rete di comunicazione sia in verticale che in orizzontale, creando una struttura piramidale con al vertice la centrale informatica e alla base gli attuatori.



Source : "CIM: Principles of Computer Integrated Manufacturing", Jean-Baptiste Waldner, John Wiley & Sons, 1992. Reproduced with author's authorization

Schema 1. Industria basata sul sistema CIM

I vantaggi di questa tipologia di organizzazione sono:

- Diminuzione del tempo necessario alla commercializzazione del prodotto (Time to market)
- Diminuzione delle scorte di prodotto
- Diminuzione considerevole dei costi grazie ad un aumento dell'efficienza
- Aumento della qualità del prodotto mediante una più accurata pianificazione

Accanto agli studi destinati al potenziamento delle macchine industriali, a partire da uno storico convegno del 1956, i ricercatori iniziano a porsi l'obiettivo di creare una struttura basata su hardware e software "in grado di fornire prestazioni di pertinenza esclusiva dell'intelligenza umana" (Marco Somalvico, *Emula e non simula*, in "Aspettando Robot", [3])., gli esperti iniziano a parlare di Intelligenza Artificiale (IA). Le ricerche sono compiute essenzialmente a livello informatico, tentando di portare ai massimi livelli le funzionalità di un calcolatore, in modo da renderlo autonomo e molto più flessibile, sempre più vicino al cervello umano. Lo sviluppo di particolari logiche computazionali è inizialmente limitato a risolvere problemi più generici quali una più celere risposta ai sensori o una diagnostica automatica, per poi giungere fino agli anni '90 con lo studio dell'intelligenza di organizzazione e delle capacità di apprendimento.

Sostanzialmente lo sviluppo della Robotica mondiale è andato di pari passo con il progresso dell'Information Technology (informatica) e dell'AI, infatti l'evolversi del sistema di programmazione del software ha influito notevolmente sui metodi di lavoro degli elaboratori, ancorati per lunghi anni a linguaggi di basso livello. Analogamente il continuo superamento dei limiti posti dalle CPU ha contribuito a fornire agli sviluppatori un continuo miglioramento delle prestazioni tra cui il potenziamento delle capacità di calcolo e dell'interoperabilità tra terminali attraverso connessioni di rete sempre più avanzate. Nel 1965 il professore Gordon Moore notò la notevole rapidità con cui aumentavano le prestazioni dei microprocessori e formulò una legge indicante che *"la densità dei transistori sui circuiti integrati sarebbe raddoppiata ogni due anni"*, alla fine degli anni '80 è stata riformulata in base ai nuovi sviluppi, il tempo di raddoppiamento scese a 18 mesi.

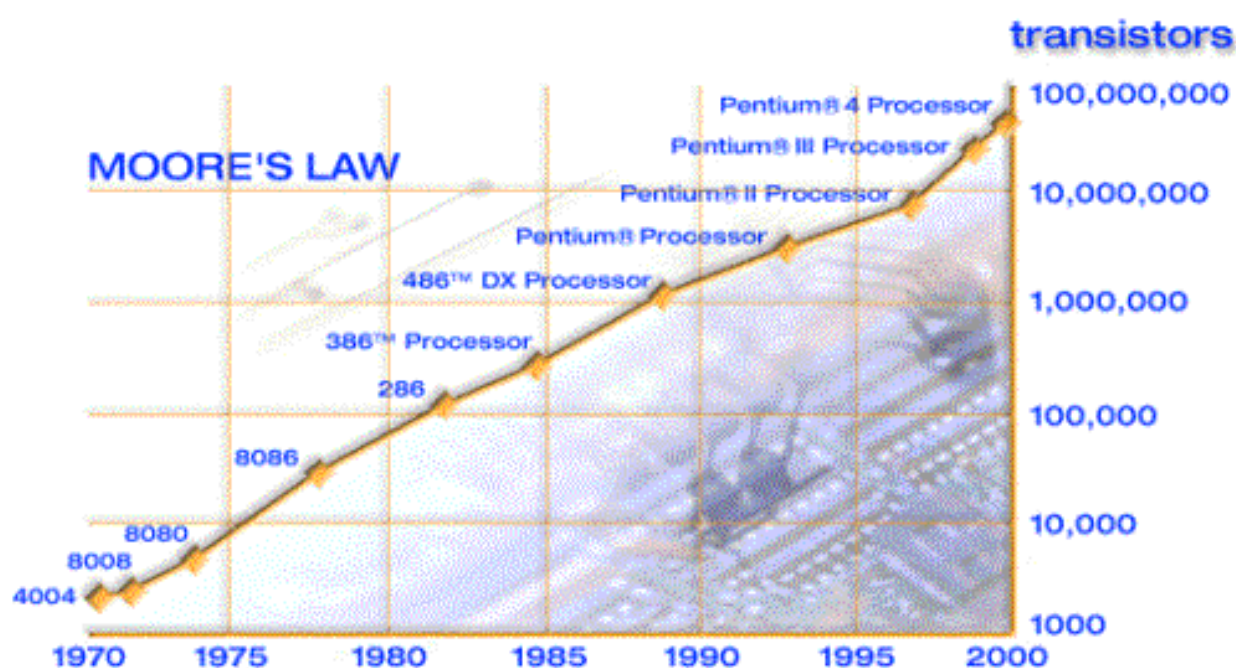


Figura 1. Legge di Moore

Seguendo tale legge un calcolatore potente quanto un cervello umano dovrebbe essere disponibile intorno al **2020** al costo di circa 1500€.

2. Intelligenza artificiale

2.1 Definizione


È l'insieme di tutti gli studi e le ricerche che hanno lo scopo di realizzare macchine in grado di comportarsi come esseri umani, ovvero di essere dotati di un'intelligenza che consenta loro di compiere operazioni con un certo grado di complessità.



L'IA corrisponde ad un programma che non si comporta diversamente da una persona, infatti, dato un "mondo" in cui vive, in ogni istante riceve informazioni da questo e lo influenza con le informazioni elaborate. Supponendo che i dati scambiati siano sempre in numero finito, si può affermare che esse corrispondano ad una quantità espressa da n bytes. (**Tesi di Church**) Si possono introdurre due funzioni **Mondo(s, d)** e **Vista(s)**, tra le quali la prima ha come argomenti la condizione del mondo e l'influenza che la nostra applicazione ha su di esso ad un dato istante. Il risultato di tale funzione è un valore che corrisponde ad una nuova condizione del mondo nell'istante successivo, mentre la seconda funzione fornisce al sistema la visione di questa. Durante la sua vita il mondo attraversa una lunga serie di stati ($s_0, s_1, s_2, s_3, \dots, s_n$), il programma elabora un certo numero di informazioni ($d_0, d_1, d_2, d_3, \dots, d_n$) e ne riceve altrettante dal mondo ($v_0, v_1, v_2, v_3, \dots, v_n$): $s_{i+1} = \text{Mondo}(s_i, d_i)$ e $v_i = \text{Vista}(s_i)$. L'elemento mancante di questo programma è ciò che probabilmente lo accomuna maggiormente all'uomo: il **significato di vita**, in particolare il concetto di piacere e dolore, che può essere implementato inserendo la capacità di valutare quanto un vettore v_a sia migliore di un v_b , quale sia una vittoria e quale una sconfitta. L'IA deve



essere quindi testata e, per evitare di ritrovarsi ad esaminare un numero infinito di candidati, si rende finito il numero dei mondi e si fornisce ai programmi un determinato limite di tempo entro il quale possa allenarsi per raggiungere un target (ad esempio rapporto molto favorevole vittorie/sconfitte), facendo attenzione a non commettere errori fatali. All'esame finale di sicuro non tutti i



programmi saranno giudicati idonei, ugualmente non tutti quelli che ci riusciranno saranno vere IA, infatti un programma scritto nello specifico per un mondo non sarà intelligente, ma alla pari di uno studente che ha imparato a memoria le risposte per un test scolastico. Rendendo elevato il numero dei mondi si facilita l'esame al programmatore, infatti la maggior parte delle applicazioni testate fallirà poco dopo l'inizio e tutte quelle che non sono IA verranno automaticamente scartate. Il programma ottenuto avrà inizialmente limitate capacità nel mondo reale e per questo motivo avrà bisogno di educazione ed allenamento con l'ausilio di insegnanti che dovranno temprarlo ed incoraggiarlo con vittorie e sconfitte. In un futuro dopo aver accudito una popolazione di questo genere molto probabilmente l'umanità stessa diventerà inutile di fronte ad una tale efficienza.


2.2 Brevi cenni storici

Il sogno dell'uomo di costruire una macchina dotato di intelligenza artificiale risale a diversi secoli fa, già nel XVIII secolo Jacques de Vaucanson realizzò un'anitra in grado di nuotare e di ingoiare chicchi di grano mentre l'orologiaio svizzero Pierre Jaquet-Droz produsse una serie di automi in grado di disegnare oppure di suonare grazie alla lettura di alcuni codici memorizzati su dischi metallici. Il filosofo e matematico Leibniz era affascinato dall'idea di realizzare una macchina in grado di risolvere qualsiasi problema a carattere scientifico, un *calculus ratiocinator*, che ispirò gli studi di J. Von Neuman e portò alla costituzione delle basi necessarie alla realizzazione dei moderni calcolatori.

In realtà le teorie fondamentali allo sviluppo del concetto attuale di IA si svilupparono tra il 1930 e il 1960 e riguardavano soprattutto studi sulla logica e sulla calcolabilità, tra questi i più importanti furono in ordine cronologico:

- Il lavoro sul neurone di McCulloch e Pitts(1943), ancora oggi oggetto di studio e ricerche
- Gli studi sull'apprendimento di Hebb(1949), da cui nacque il concetto di "apprendimento hebbiano", basato sulle connessioni tra neuroni
- La costruzione del primo computer neurale da parte di Minsky(1951)

Nell'estate del 1956 J. McCarthy organizzò al Dartmouth College un seminario della durata di due mesi a cui parteciparono tutti coloro che in quel periodo si occupavano di studi sulle reti neurali e sull'intelligenza. Di sicuro non si pose una soluzione al dilemma sull'IA in così breve tempo ma l'incontro di esperti di fama mondiale fece in modo che tutte le ricerche proseguissero in una



comune direzione ed in maniera molto più rapida. Nel 1957 grazie allo studio di particolari sintassi vennero alla luce i primi linguaggi di programmazione che diedero una svolta alle ricerche sui sistemi basati su una combinazione fra meccanica ed informatica. L'invenzione del LISP(List Processing), un linguaggio derivato dal FORTRAN, ma dedicato all'elaborazione di informazione simbolica e non al calcolo numerico, e dell'Advice Taker, un programma che consentiva di compiere inferenze, diedero il via ad un'evoluzione unica nel suo genere. I nuovi software potevano compiere azioni prima inimmaginabili, come giocare a dama e a scacchi, risolvere problemi di geometria e comprendere semplici frasi in linguaggio naturale.

Nel 1965 J. Weizenbaum scrisse il programma ELIZA, famoso ancora oggi per l'esempio che è riuscito a dare di intelligenza artificiale, tanto da turbare il suo realizzatore che preoccupato per il successo ottenuto abbandonò le ricerche nel campo. A partire dagli Anni 80 l'IA è diventata un vero business, entrando a tutti gli effetti nell'industria, sia come elemento fondamentale nella catena della produzione sia come oggetto di sviluppo da parte di compagnie. Fino ad oggi i fondi investiti nella ricerca sono gradualmente aumentati e si sono orientati verso lo sviluppo di intelligenze ispirate alla natura in quanto in grado di evolversi singolarmente e collettivamente.

2.3 Fondamenti teorici dell'IA

La prima formulazione della teoria dell'intelligenza artificiale si fa risalire al pensiero del filosofo seicentesco Thomas Hobbes che nella sua opera principale il *Leviatano* riportò la frase "Ragionare non è nient'altro che calcolare", paragonando il ragionamento ad una manipolazione di corpi fisici definiti "particelle mentali". Questa operazione avveniva sulla base di regole ben precise e con un procedimento simile a quello seguito dalle palline di un abaco, ma non così strettamente matematico. Queste considerazioni per quanto semplici ed ancora primordiali sono comunque fondamentali per comprendere che il concetto di attività mentale si basa su un insieme di calcoli e ci consentono di introdurre una serie di assiomi necessari allo sviluppo dell'IA:

- Il ragionamento in quanto risultato di un calcolo
- Il concetto di simbolo
- Il concetto di calcolo in quanto manipolazione di simboli
- l'idea della possibile esistenza di un manipolatore automatico di simboli

2.3.1 Il concetto di simbolo

Un simbolo è semplicemente un oggetto che rappresenta un altro oggetto, ma possiede la proprietà di non essere la sua copia esatta, ma il risultato di un procedimento di selezione tra le sue molteplici caratteristiche, ovvero di un'astrazione. Esistono due tipi di rappresentazioni in base al rapporto tra simbolo ed oggetto raffigurato: una foto di un paesaggio è il risultato dell'impressione della luce su una pellicola, un disegno, invece, è la nostra interpretazione dello stesso ottenuta tramite l'utilizzo di linee di diversa inclinazione. I due tipi che conservano un rapporto di somiglianza, sono in analogia. Analizzando attentamente la fotografia si può produrre la relativa immagine analitica basata sulla geometria di Cartesio. Tramite una serie di equazioni, ovvero di formule matematiche, si è costruita una rappresentazione simbolica dell'oggetto e lo stesso può essere fatto per ogni altro elemento della realtà.



Figura 3. Fotografia di un paesaggio montano

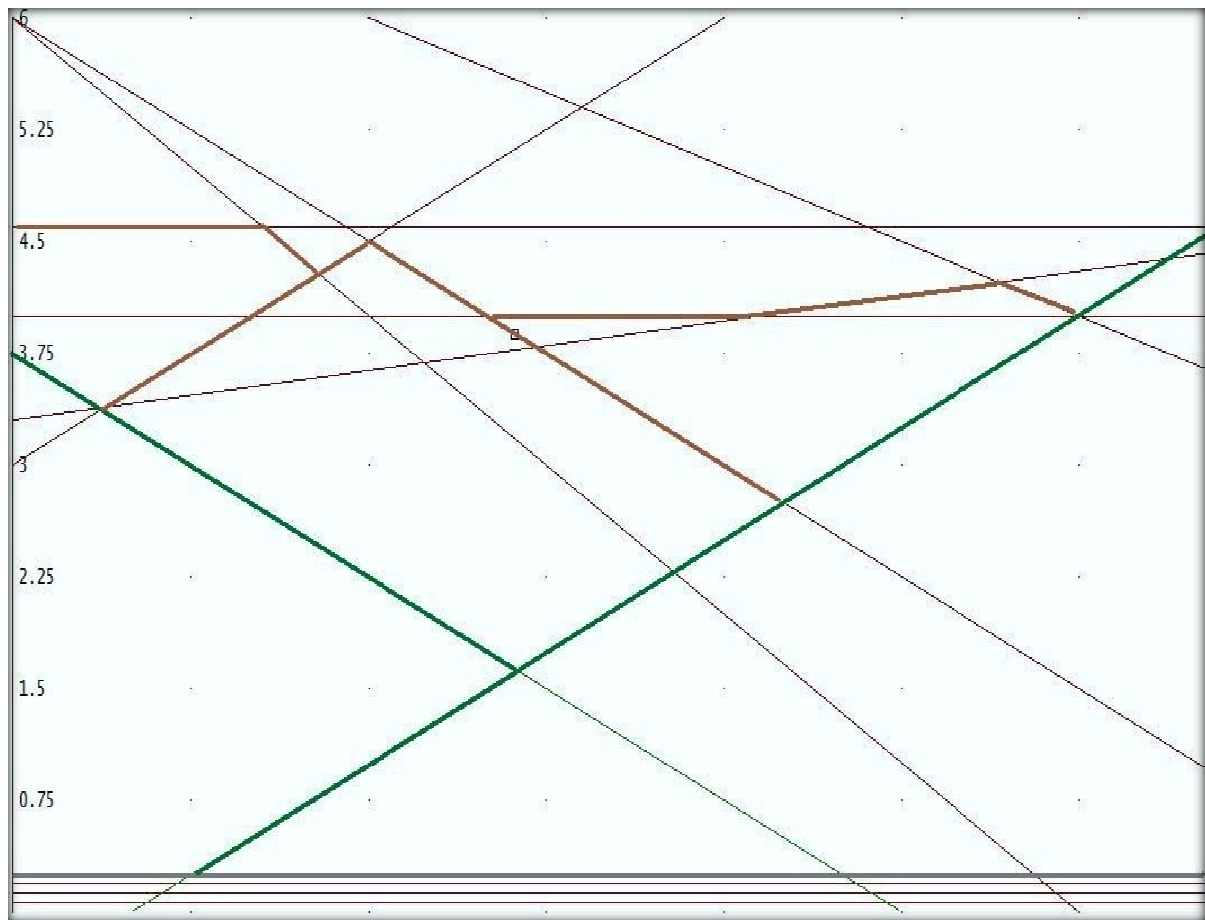


Figura 4. Rappresentazione analitica rudimentale del paesaggio montano

Equazioni delle rette	
Rette orizzontali	Rette oblique
$y = \frac{1}{4}$	$y = \frac{3}{2}x + \frac{1}{2}$
$y = \frac{1}{8}$	$y = \frac{3}{2}x + 3$
$y = \frac{3}{16}$	$y = -\frac{3}{2}x + 6$
$y = \frac{1}{16}$	$y = -2x + 6$
$y = \frac{9}{2}$	$y = \frac{1}{3}x + \frac{3}{2}$
$y = 4$	$y = \frac{1}{3}x + \frac{10}{3}$
	$y = -\frac{3}{2}x + \frac{15}{4}$

2.3.2 Il calcolo come manipolazione di simboli

L'utilità della figura analitica si basa soprattutto sulla possibilità che offre di studiarla ed analizzarne alcune caratteristiche tramite operazioni sulle rette componenti. Attraverso l'applicazione di alcune formule è possibile giungere a risultati inaspettati, con semplici ma opportuni spostamenti, aggiunte ed eliminazioni di simboli si ottengono informazioni aggiuntive, che non sono altro che ulteriori simboli. Tali elementi sono le basi costitutive delle equazioni del grafico, pur non conoscendo l'algebra, ma possedendo un manuale di istruzioni su come compiere tali trasformazioni, saremmo in grado di calcolare.

Avendo effettuato queste semplici considerazioni, ora si è in grado di estendere il concetto di calcolo ad una serie di fenomeni da cui spesso lo si escludeva a priori. Uno degli esempi più banali è quello degli scacchi, un gioco diffuso in tutto il mondo in cui è necessaria una scacchiera composta da 64 caselle e 16 pezzi per giocatore di colore differente, lo scopo è quello di eliminare il Re avversario, ponendosi al suo posto sulla sua casella e attuando il cosiddetto SCACCO MATTO. L'analisi del gioco porta alla luce alcuni elementi basilari ma fondamentali:

- i simboli che differiscono per la loro forma
- uno stato iniziale del gioco: la scacchiera con i pezzi nelle posizioni di partenza
- un insieme di regole che regolano lo spostamento dei pezzi in determinate situazioni
- uno stato finale del gioco dato dalla posizione finale dei pezzi e l'eventuale vincitore



Per applicare in maniera adeguata la nozione del calcolo al gioco degli scacchi diviene molto utile assegnare alla scacchiera una semplice notazione basata sulla numerazione da 1 a 8 per le righe e la serie di lettere dalla A alla H per le colonne. Ogni casella risulta identificata biunivocamente dalla coppia ordinata formata dalla lettera della colonna e dal numero della riga.

Allo stesso modo ad ogni pezzo corrisponde un carattere particolare uguale alla prima lettera del proprio nome a differenza dei pedoni (pezzi

posizionati sulle righe 2 e 7) che non richiedono alcun identificativo.

I giocatori si alternano nell'attuare ognuno la propria mossa identificata anch'essa biunivocamente dall'eventuale lettera del pezzo da muovere seguita dalla sua casella di destinazione.



Il giocatore 1, che possiede i pezzi di colore bianco, decide di muovere il pedone nella casella f4, l'unico pedone che in base alle regole può effettuare questa mossa è quello indicato nella figura.

1.f4



Il giocatore 2, che possiede i pezzi neri, risponde muovendo un suo pedone nella casella e6.

2.e6



Il giocatore 1 muove nuovamente un pedone, questa volta in g4.

3.g4



Il giocatore 2, notando l'errore dell'avversario, decide di muovere la Donna(D) in h4.

4.Dh4#

Il simbolo # indica la posizione di scacco matto, in quanto il giocatore 1 non può evitare in alcun modo che alla prossima mossa dell'avversario il proprio Re resti incolume.

Questo è lo stato finale della partita identificato da questa sequenza di mosse dalla quale si decreta la vittoria del giocatore 2.

Allo stesso modo sequenze differenti di mosse sviluppano partite completamente differenti, dando la possibilità ad ognuno dei due giocatori di aggiudicarsi la vittoria.

L'esempio seppur molto semplice illustra quanto sia estesa la nozione di calcolo basata sulla manipolazione di simboli in particolare nello studio di sistemi che si basano su un insieme finito di elementi distinti e su un insieme finito di regole che ne specificano le trasformazioni. Tali sistemi si basano unicamente sul proprio insieme di formule e di simboli e non implicano assolutamente un particolare tipo di materiale o di supporto, essi vengono definiti **sistemi formali**. A questa categoria, oltre ad un gran numero di giochi, è possibile includere l'intelligenza artificiale ed in maniera più complessa anche il pensiero.

2.4 La macchina e il test di Turing

Sulla base delle considerazioni sui sistemi formali e grazie agli sviluppi dell'algebra booleana, che operava operazioni di calcolo grazie alle leggi della logica, il matematico inglese Alan Mathison Turing(1912-1954) nel saggio *Sui numeri computabili con una applicazione al "problema della decisione"* del 1936 teorizzò il funzionamento di una particolare macchina in grado di eseguire qualsiasi tipo di algoritmo, essa venne in seguito definita **macchina di Turing**.

Una macchina di Turing è costituita da un dispositivo simile ad una testina, in grado di spostarsi per leggere e scrivere su un nastro di lunghezza infinita diviso in celle.

Stabilito un insieme di regole da fornire alla macchina, la macchina compie per passi successivi una lettura del simbolo contenuto nella cella ed in base a questo e al suo stato interno lo sostituirà con un nuovo simbolo. La macchina di Turing potenzialmente non presenta limiti infatti con la presenza di adeguate regole può svolgere qualsiasi operazione ed i suoi simboli non devono essere obbligatoriamente numeri, ma possono essere simboli qualsiasi in modo da poter simulare qualsiasi sistema formale.

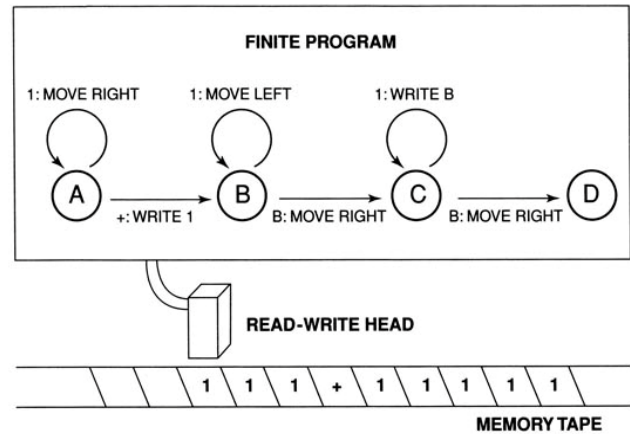


Figura 5. Macchina di Turing

Il più sofisticato dei moderni calcolatori digitali si basa sul funzionamento della macchina di Turing, ma essendo anche il nostro cervello un abile manipolatore di simboli nell'atto di pensare, ragionare e capire si può concludere che anch'esso ne è una complessa implementazione. Quest'ultima affermazione rappresenta il fondamento della *teoria rappresentazionale della mente*, indispensabile nello studio dell'intelligenza artificiale, secondo la quale la possibilità di conferire ad un computer le capacità di un cervello umano corrisponde alla trasformazione in programma delle regole con cui esso gestisce i simboli mentali.

Esempio di macchina di Turing in grado di effettuare una divisione realizzata in C++

```
int main(){
    /* n = variabile per l'inserimento della cifra
    r = variabile per il resto
    i = contatore
    c = quoziente*/
    int n=0,r=0,i=0,c=0;
    /*itero finche non esce 123*/
    while (n != 123){
        /*per muovermi a destra di uno ogni giro*/
        if(i!=0) c*=10;
        /*per tutti i casi possibili e per gli eventuali errori*/
        switch (n){
            case 0:
            case 1:
            case 2:
            case 3:
```

```

case 4:
case 5:
case 6:
case 7:
case 9:
case 8:
    if(( n % 2 == 0) && (r==0)) {
        c+=(n/2);
        r=0;
    }
    else if (( n % 2 == 0) && (r==1)) {
        c+=((n+10)/2);
        r=0;
    }
    if(( n % 2 != 0) && (r==0)) {
        c+=((n-1)/2);
        r=1;
    }
    else if (( n % 2 != 0) && (r==1)) {
        c+=((n-1+10)/2);
        r=1;
    }
    break;
default:
    cout<<"valore errato decimale "<<endl;
    break;
}
/*inserisce la cifra se e 123 esce dal programma e da i risultati*/
cout<<"Inserisci la cifra 123 per uscire "<<endl;
cin>>n;
/*incremento il contatore*/
i++;
}
/*stampo i risultati sullo schermo*/
cout<<endl<<"Quoziente = "<<c<<endl;
cout<<"Resto = "<<r<<endl;
system("pause");
return 0;
}

```

Considerando che sia possibile costruire una macchina di questo tipo, come si può essere sicuri che è realmente intelligente?

Una delle vie più semplici potrebbe sembrare all'apparenza quella di fornire una definizione di intelligenza e di valutare se il sistema considerato risponde ai criteri imposti da quest'ultima, ma nessuno finora è riuscito a trovare una definizione adeguata che mettesse d'accordo gli studiosi. Una pregevole soluzione è stata trovata ancora una volta dallo studioso Alan Turing che esordisce nel suo ragionamento con l'enigmatica domanda «Possono le macchine pensare?» e per la ricerca di una risposta elabora una specie di gioco, simile ad un gioco dell'imitazione.

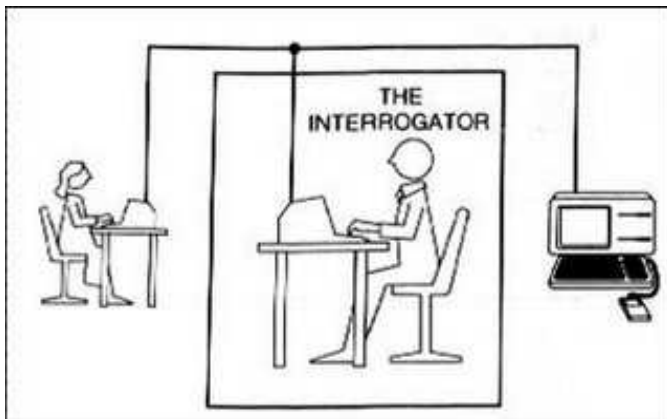


Figura 6. Simulazione del test di Turing

I partecipanti sono tre persone estranee tra loro, tra cui due, obbligatoriamente di sesso opposto, sono i candidati, mentre il terzo è colui che interroga. Quest'ultimo viene posto in una stanza, in modo da essere privato di ogni contatto con l'esterno, in particolare con gli altri due personaggi; il suo scopo è quello di capire quale di loro sia l'uomo e quale la donna, allo stesso modo

gli altri devono ingannarlo ed evitare che raggiunga il proprio obiettivo. L'unico mezzo di comunicazione tra l'interrogante e gli interrogati è un dispositivo elettronico con cui possono inviare e ricevere qualsiasi tipo di messaggio, evitando di lasciar trasparire elementi utili al loro riconoscimento come l'aspetto e la voce. Turing si convince che facendo partecipare a questo gioco una macchina, si potrà verificare la sua presunta intelligenza. Una sistema di questo genere non dovrebbe mostrare la sua abilità nei calcoli matematici, ma più che altro la sua versatilità nel rispondere a domande sui più svariati argomenti come l'amore e l'amicizia alla pari di un essere umano.

Questo simpatico gioco ha preso il nome di **test di Turing** ed ancora oggi costituisce un punto cruciale nello studio dell'intelligenza artificiale, non essendo finora nessuna macchina riuscita ad implementare le seguenti funzionalità:

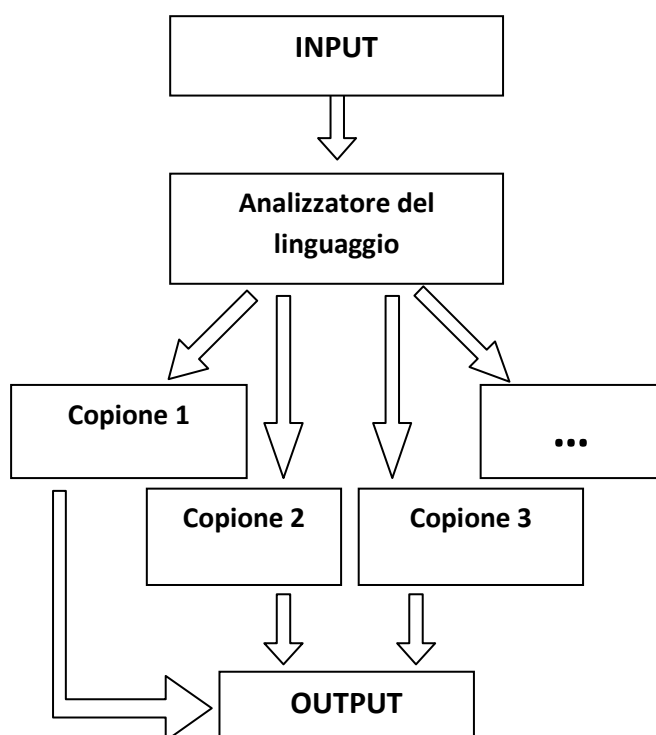
- Elaborazione del linguaggio naturale
- Rappresentazione della conoscenza
- Ragionamento automatico
- Apprendimento automatico

2.5 I primi programmi

Il caso di Eliza e di Parry

Nel 1964 un ricercatore del MIT Joseph Weizenbaum, effettuando ricerche nel campo dell'intelligenza artificiale, sviluppò un programma conosciuto come *Eliza*. All'apparenza il programma sembrava essere dotato di notevoli capacità cognitive ed abilità dialogiche, in realtà la sua era una struttura incredibilmente semplice in grado di dimostrare solo una ristretta competenza linguistica in inglese. Il programmatore intendeva emulare le funzioni base del cervello umano, quindi sviluppò il suo software su due livelli: il primo costituito da un **analizzatore del linguaggio** in grado di comprendere l'argomento del discorso e rimandare al secondo livello, in particolare al corrispondente **copione** da seguire.

Schema 2. Funzionamento e dialogo di Eliza



Ragazza: Gli uomini sono tutti uguali.

Eliza: **In che senso?**

Ragazza: Trovano sempre il modo di tormentarci.

Eliza: **Può farmi un esempio?**

Ragazza: Beh è il mio ragazzo che mi ha fatto venire qui.

Eliza: **È il suo ragazzo che l'ha fatta venire?**

Ragazza: Dice che sono depressa.

Eliza: **Mi dispiace sentire che è depressa.**

Ragazza: È vero. Sono infelice.

Eliza: **Pensa che il fatto di venire qui l'aiuterà a non essere infelice.**

Eliza sarebbe stato in grado di sostenere qualsiasi conversazione purché supportato dal corrispondente copione di cui il primo esemplare rispecchiava la parte di uno psicoterapeuta di scuola rogersiana, il quale durante una seduta interviene il meno possibile e nella maggior parte dei casi ripetendo le parole del paziente, questo fattore rendeva ancora più verosimile

l'interazione col software. Inaspettatamente il programma ebbe un enorme successo, tanto da indurre persone con problemi mentali a servirsene per le proprie terapie, da cui sortirono numerosi effetti positivi, e da stimolare la ricerca nel campo. Pochi anni dopo Kenneth Colby portò alla luce il suo nuovo programma *Parry* capace, a differenza del suo predecessore, di simulare i discorsi di un paranoico in maniera così realistica da ingannare i numerosi psichiatri che lo avevano intervistato telematicamente. Sembrava che entrambi i software avessero superato il test di Turing in realtà nessuno dei due era dotato di una vera intelligenza, ma di semplici trucchi di programmazione che la voglia di sfogarsi del gran numero di persone che lo avevano provato avevano spiritualizzato.

Parte del codice di Eliza realizzato in BASIC

```

10 '
20 '           Eliza/Doctor
30 '           Original author: Joseph Weizenbaum
40 '           This version: Frederick B. Maxwell, Jr.
50 '           Version 1.0   2/12/85   Initial CP/M (MBASIC) release.
60 '           Version 2.0   6/13/89   Initial PC (GWBASIC) release.
70 '
80 '*****
90 '
100 '           This version of Eliza is released into the public domain.
110 '
120 '*****
130 '
140 DEFINT A-Z           :' no floating point is used.
150 DIM REPLIES$(300)    :' up to 300 responses.
160 DIM KWD$(200)        :' up to 200 keywords.
170 DIM FIRST(200)       :' first reply for keyword number in subscript.
180 DIM LAST(200)        :' last reply " " " " " "
190 DIM OFFSET(200)      :' offset from first reply for each keyword.
200 '
210 '*****
220 '
230 '           -Initialization-
240 '           We will read in data from the file ELIZA.DAT in the following format:
250 '           KEYWORD 1
260 '           KEYWORD N           all keywords which will get the same responses
270 '           !                   indicates end of keywords
280 '           RESPONSE 1          all responses for this/these keywords.
290 '           RESPONSE N
300 '           .                   indicates end of responses
310 '
320 '*****
330 '
340 OPEN "I",1,"ELIZA.DAT"      :' file with keyword/response data.
350 MAXKEY = 0                  :' number of keywords
360 MINREPLY = 1                :' first reply for first keyword.
370 '
380 WHILE NOT EOF(1)
390   NUMKEYS = 0                :' number of keys with same responses.
400   LINE INPUT#1, F$          :' get keyword or !
410   IF F$ = "!" THEN 480       :' if ! then get replies.
420   MAXKEY = MAXKEY + 1        :' we've got one more keyword
430   PRINT ". ";               :' let user know we're still alive.
440   NUMKEYS = NUMKEYS + 1      :' 1 more keyword with same replies.
450   KWD$(MAXKEY) = " " + F$ + " " :' put in a keyword bounded with spaces.
460   GOTO 400

```

```

470 '
480 LINE INPUT#1 , F$      :' get the reply or .
490 IF F$ = "." THEN 540   :' check for end of reply list.
500 MAXREPLY = MAXREPLY + 1      :' 1 more reply.
510 REPLIES$( MAXREPLY ) = F$
520 GOTO 480
530 '
540 FOR KWD = MAXKEY - NUMKEYS + 1 TO MAXKEY
550     FIRST(KWD) = MINREPLY      :' first reply for key.
560     LAST(KWD) = MAXREPLY      :' last reply for key.
570 NEXT KWD
580 MINREPLY = MAXREPLY + 1      :' set up for next keyword.
590 '
600 WEND
610 CLOSE
620 '
630 *****
640 '
650             Here we go! Send intro message to "patient".
660 '
670 *****
680 '
690 PRINT
700 PRINT "HI! I'M ELIZA. LET'S TALK. TYPE `BYE' TO END THIS SESSION."
710 '
720 *****
730 '
740             Get the user input into I$.
750 '
760 *****
770 '
780 PRINT ">"; : LINE INPUT I$      :' Get user input.
790 IF I$ = "" THEN 780    :' Just hitting return isn't good enough.
800 I$ = " " + I$ + " "      :' Put a space on each end.
810 '
820 *****
830 '
840             Get rid of punctuation/extraneous characters, and make uppercase.
850 '
860 *****
870 '
880 L=1                      :' Start at the first character
890 C$ = MID$(I$,L,1)      :' Get the character.
900 '
910             Capitalize if necessary.
920 '
930 IF C$ >= "a" AND C$ <= "z" THEN MID$(I$,L,1) = CHR$(ASC(C$)-&H20): GOTO 890
940 '
950 IF C$ = " " THEN 990      :' Spaces are OK.
960 IF C$ >= "0" AND C$ <= "9" THEN 990      :' So are numbers.
970 IF C$ >= "A" AND C$ <= "Z" THEN 990      :' So are capital letters.
980 I$ = LEFT$(I$,L-1) + MID$(I$,L+1): GOTO 890      :' Delete character.
990 L=L+1                      :' Next character.
1000 IF L <= LEN(I$) THEN 890      :' Keep on processin'
1010 '
1020 *****
1030 '
1040             Has he/she said this before? Does he/she want to terminate session?
1050 '
1060 *****
1070 '
1080 IF I$=PREVIOUS$ THEN PRINT "PLEASE DON'T REPEAT YOURSELF!":GOTO 730
1090 PREVIOUS$ = I$          :' Set up for next run.
1100 '
1110 IF I$ = " BYE " THEN PRINT "TALK TO YOU LATER! BYE!" : SYSTEM
1120 '
1130 *****
1140 '
1150             Find keyword in user input string (I$).
1160 '

```

```

1170 *****
1180 '
1190 FOR K=1 TO MAXKEY-1           :' Start search at keyword number 1.
1200   C = INSTR( I$, KWD$(K) )    :' Look for the keyword in the string.
1210   IF C <> 0 THEN 1230         :' Exit on match.
1220 NEXT K
1230 KWD = K                       :' Keyword number.
1240 IF KWD = MAXKEY THEN 1280     :' We don't need anything if no match.
1250 REMAINS$ = MID$(I$,C-1+LEN(KWD$(K)))  :' Grab remainder for reply.
1260 '
1270 *****
1280 '
1290 '           Take everything after the keyword (remains$) and conjugate it
1300 '           using the data for conjugation.
1310 '
1320 *****
1330 '
1340 RESTORE
1350 READ S$,R$                   :' Read search and replacement words.
1360 IF S$ = "." THEN 1430         :' Periods (.) indicate end of data.
1370 C = INSTR( REMAINS$ , S$ )    :' Search for string S$ in REMAINS$
1380 IF C = 0 THEN 1350           :' If no match, try the next one.
1390 TEMP$ = LEFT$(REMAINS$,C-1)   :' Replacement.
1400 TEMP$ = TEMP$ + R$           :' Word.
1410 REMAINS$ = TEMP$ + MID$(REMAINS$,C+LEN(S$))  :' Right side.
1420 GOTO 1350                   :' Next conjugation to be done.
1430 C = INSTR( REMAINS$ , "+" )    :' Strip the plus signs out.
1440 IF C = 0 THEN 1470
1450 REMAINS$ = LEFT$( REMAINS$ , C-1 ) + MID$( REMAINS$ , C+1 ):' Strip it.
1460 GOTO 1430                   :' Go for the next one.
1470 '
1480 ' Handle the special case of " I " being the last word.
1490 '
1500 IF RIGHT$( REMAINS$ , 3 ) <> " I " THEN 1540
1510 REMAINS$ = LEFT$ ( REMAINS$ , LEN(REMAINS$) - 2 ) + "ME "
1520 '
1530 *****
1540 '
1550 '           Get the reply using the keyword number (KWD).
1560 '
1570 *****
1580 '
1590 REPLY$ = REPLIES$( FIRST(KWD) + OFFSET(KWD) )           :' Get reply.
1600 OFFSET(KWD) = OFFSET(KWD) + 1                         :' Point to next reply.
1610 IF OFFSET(KWD) + FIRST(KWD) > LAST(KWD) THEN OFFSET(KWD) = 0 : ' Wrap.
1620 '
1630 '           Bump offsets on all keywords that use these replys.
1640 '
1650 FOR TEMP = 1 TO MAXKEY
1660   IF FIRST(TEMP) = FIRST(KWD) THEN OFFSET(TEMP) = OFFSET(KWD)
1670 NEXT TEMP
1680 '
1690 '           If the last character of the reply is *, append REMAINS$ to reply.
1700 '
1710 IF RIGHT$(REPLY$,1)="*" THEN REPLY$=LEFT$(REPLY$,LEN(REPLY$)-1)+REMAINS$
1720 PRINT REPLY$
1730 GOTO 730
1740 '
1750 *****
1760 '
1770 '           Data for conjugations in the following form:
1780 '           Word to replace , Replacement with + appended on end
1790 '           + is to keep the word from being switched back later and will
1800 '           be stripped before output.
1810 '
1820 *****
1830 '
1840 DATA " ARE " , " AM+ "
1850 DATA " AM " , " ARE+ "
1860 DATA " WERE " , " WAS+ "

```



```
1870 DATA " WAS " , " WERE+ "  
1880 DATA " YOU " , " I+ "  
1890 DATA " I " , " YOU+ "  
1900 DATA " YOUR " , " MY+ "  
1910 DATA " MY " , " YOUR+ "  
1920 DATA " IVE " , " YOUVE+ "  
1930 DATA " YOUVE " , " IVE+ "  
1940 DATA " IM " , " YOURE+ "  
1950 DATA " ME " , " YOU+ "  
1960 DATA " US " , " YOU+ "  
1970 DATA " WE " , " YOU+ "  
1980 DATA " . , " . "  
1990 '  
2000 END
```

2.6 L'analisi del linguaggio naturale

Alla base del funzionamento di ogni sistema sviluppato secondo i principi dell'intelligenza artificiale dovrebbe esserci la sua capacità di elaborare un linguaggio basato su simboli di cui riesce a comprendere il significato. Allo stesso modo migliaia di anni fa l'homo sapiens, dando vita ad un primordiale linguaggio verbale, riuscì a compiere un grande passo in avanti dal punto di vista evolutivo. Dalla stessa analisi dei principi su cui si basa il test di Turing si nota quanto i concetti di linguaggio ed intelligenza siano strettamente connessi ed indispensabili l'uno all'altro.

La prospettiva di creare una macchina intelligente in grado di riprodurre lo stesso linguaggio verbale umano spinse un elevato numero di ricercatori a lavorare allo sviluppo di speciali software, grazie al sostegno economico fornito da compagnie interessate al progetto per fini commerciali. I primi tentativi diedero vita a rudimentali software di traduzione automatica, che si rivelarono complessi a tal punto da far svanire quell'iniziale entusiasmo che tanto aveva animato i laboratori mondiali. Gli studi sul *natural language processing* non si sono mai arrestati ed attualmente costituiscono la *linguistica computazionale*, uno dei settori attualmente più sviluppati dell'intelligenza artificiale.

2.6.1 Le grammatiche di Chomsky e l'analisi automatica del linguaggio naturale

Noam Chomsky fu il primo a sviluppare una teoria linguistica come base per i futuri sviluppi dell'elaborazione del linguaggio naturale. Le sue idee si basavano sulla capacità di ogni uomo di poter sviluppare un discorso attraverso un bagaglio di conoscenze, nella maggior parte dei casi inconsapevole, definito *competenza linguistica*, che a sua volta si articola in:

- Competenza fonologica → capacità di comprendere e riprodurre i suoni della lingua parlata
- Competenza sintattica → capacità di riconoscere la correttezza delle strutture grammaticali
- Competenza semantica → capacità di conferire ed estrapolare il significato dalle frasi

Chomsky assegnò prioritaria importanza alla competenza sintattica, ritenendo che solo grazie ad essa si è in grado di percepire immediatamente la correttezza di una frase.

(a) "La macchina comprende il linguaggio"

è una proposizione corretta dal punto di vista grammaticale, invece

(b) "La macchina linguaggio comprende il"

non lo è. La frase (b) non risponde alle regole formali che costituiscono i fondamenti della grammatica della lingua considerata. Le regole principali sono quelle che determinano la struttura principale della frase, suddivisa in maniera ideale in gruppi funzionali definiti *sintagmi*.

$F \rightarrow SN - SV$

$SN \rightarrow DET - N$

$SV \rightarrow V - DET - SN$

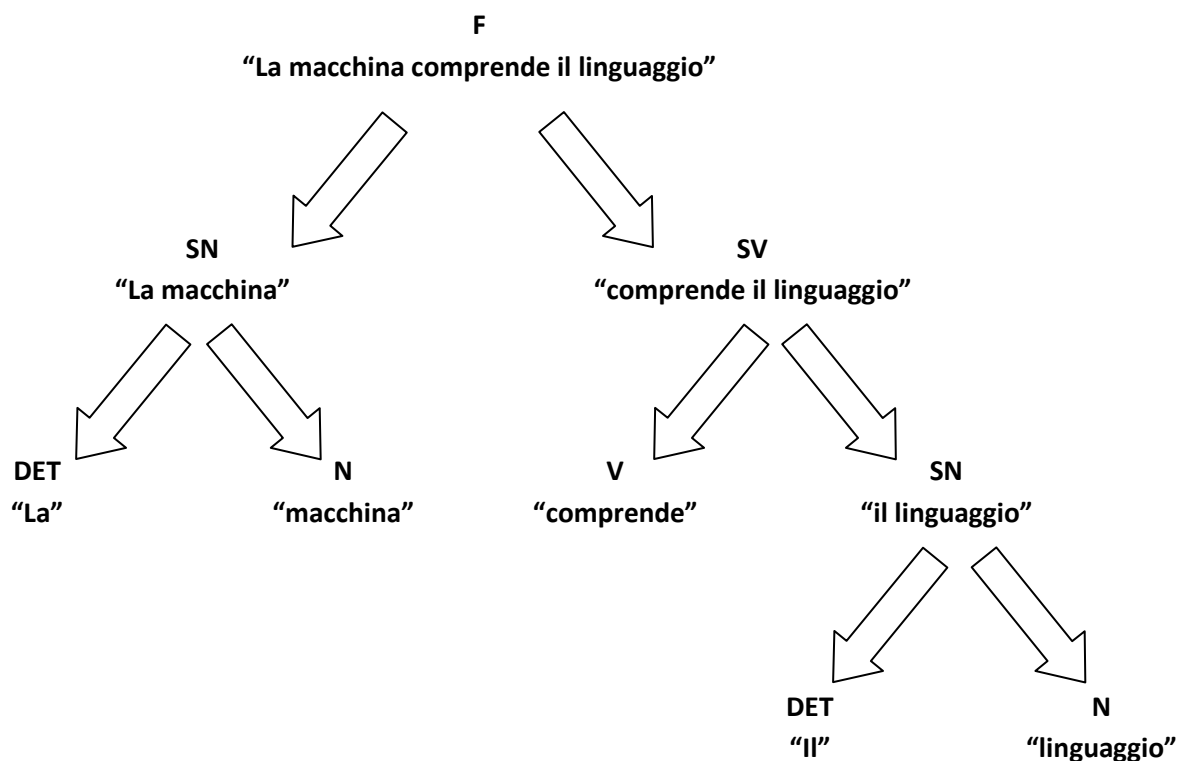
$DET \rightarrow la, il$

$N \rightarrow macchina, linguaggio$

$V \rightarrow comprende$

- La frase F è composta essenzialmente da un sintagma nominale (SN – "La macchina") e da uno verbale (SV – "comprende il linguaggio")
- Il sintagma nominale è costituito da un determinante (DET - "La") e dal nome (N - "macchina")
- Il sintagma verbale è a sua volta formato da un verbo (V – "comprende") e da un sintagma nominale (SN – "il linguaggio")
- L'ultimo sintagma nominale comprende un determinante (DET – "il") ed un nome (N – "linguaggio")

Il principio base delle seguenti regole è che esse vengano applicate sino all'arrivo ad un *simbolo terminale*, ovvero ad un simbolo che non possa essere scomposto ulteriormente.



Schema 3. Suddivisione secondo la grammatica Chomsky

Seguendo questo schema in base alle regole di una grammatica generativo-trasformatzionale, è impossibile giungere ad una frase come "La macchina linguaggio comprende il", allo stesso modo applicando ulteriori regole di Chomsky si può giungere ad ulteriori risultati come la stessa frase alla forma passiva "Il linguaggio è compreso dalla macchina".

Questi fondamenti teorici seppur molto semplici rappresentano i principi base su cui lavorano i software *parser* addetti all'analisi e alla successiva verifica grammaticale della struttura di una frase. Attualmente le maggiori difficoltà si incontrano proprio nell'applicare simili modelli di regole grammaticali al fine di formalizzare le lingue naturali.

2.6.2 Il problema del significato

Un ulteriore e non meno importante aspetto dell'elaborazione del linguaggio naturale è quello relativo al significato delle parole e delle strutture proposizionali, ovvero a tutto ciò che riguarda il campo della competenza semantica. Ogni programma dotato di una competenza sintattica basata sulla grammatica Chomsky non rilevarebbe alcuna differenza tra le due frasi "La macchina comprende il linguaggio" e "Il linguaggio comprende la macchina", sebbene la seconda, corretta

grammaticalmente, non abbia alcun senso. Fattori come questo complicano il compito degli studiosi dell'intelligenza artificiale, conferendo al problema un carattere essenzialmente filosofico e basato sull'analisi dei concetti fondamentali della teoria rappresentazionale della mente.

Le ricerche in campo teorico hanno condotto allo sviluppo di alcune strategie in grado di fornire una semantica alle macchine basate sull'intelligenza artificiale. Esse si basano principalmente sullo sviluppo di due punti:

- Conferimento di un significato lessicale alle parole
- Sviluppo delle connessioni tra le parole e le proposizioni

Il primo è facilmente attuabile tramite la scomposizione di un termine in ulteriori termini più semplici che ne esplichino il significato di base.

cane → *oggetto, fisico, animato...*

I collegamenti invece derivano dallo studio della *logica*, ovvero dalla scienza che studia lo sviluppo di un ragionamento, che conduce da premesse valide a conclusioni valide. Per secoli a partire da Aristotele i logici si sono prodigati nello sviluppo di un linguaggio formalizzato in grado di racchiudere le regole della logica di un linguaggio naturale senza mai pervenire a risultati condivisi.

La proposizione "La macchina è intelligente" attribuisce la proprietà di essere intelligente all'oggetto che è la macchina, di conseguenza il tutto può essere riscritto in un ulteriore modo:

(a) Essere intelligente (macchina)

La parte di proposizione indicata con la lettera (a) predica la proprietà dell'elemento tra parentesi e viene definita *predicato*, l'altro invece prende il nome di *argomento*. Una volta fatta questa generalizzazione è facile comprendere che la proprietà espressa può essere applicata ad un argomento qualsiasi.

(b) Essere intelligente (x)

Allo stesso modo si può notare che le stesse proprietà sono molteplici, tanto da consentire un'ulteriore generalizzazione fino ad ottenere la forma base:

(c) $p(x)$

Effettuate tali considerazioni, nel linguaggio naturale si comprende la facilità con cui è possibile sfruttare le congiunzioni per creare interconnessioni fra le proposizioni e generare , a partire da semplici enunciati, strutture gradualmente più complesse. La logica garantisce parimenti tale possibilità grazie all'impiego di particolari operatori, detti *connettivi*. Tra i principali è facile trovare i seguenti:

- AND → Congiunzione logica (latino *et*)
- OR → Disgiunzione inclusiva (latino *vel*)
- NOT → Negazione logica (latino *non*)

(e) Essere intelligente (macchina) AND essere economica (macchina, automobile)

Lo studio degli enunciati complessi parte proprio dallo studio delle unità che li compongono, ovvero da tutti gli enunciati più semplici che ne fanno parte. Accertando la falsità o la verità di questi ultimi è possibile inseguito stabilire quella della struttura di partenza.

2.6.3 I micromondi e SHRDLU

Le tecniche basate sull'analisi logica e sulla scomposizione degli enunciati, spesso molto difficili per essere applicate in maniera esauriente, hanno mostrato ben presto la loro inadeguatezza nell'applicazione a sistemi computerizzati, soprattutto per la scelta dei componenti minimi di un testo.

(a) "Marco era appena tornato a casa dalla corsa, ma quando aprì il frigorifero scoprì che era vuoto."

Un periodo di questo tipo, sebbene sia di facile comprensione per un qualsiasi moderno conoscitore della lingua, comporta una notevole quantità di informazioni non espresse esplicitamente e di sicuro non ottenibili con l'ausilio dell'analisi logica. È necessario sapere che durante una corsa si suda, con il sudore si perdono liquidi, con la perdita di liquidi giunge la sete, avere sete comporta desiderio di bere qualcosa preferibilmente di fresco, le bevande fresche si trovano in frigorifero. Questo insieme di informazioni fanno capire che un'intelligenza, per poter agire in maniera adeguata nel mondo in cui vive, deve possederne una dettagliata conoscenza.

Intorno al 1970 un gruppo di studiosi pensò di ridurre al minimo l'ampiezza di questi luoghi circoscritti e di aumentarne indefinitamente il numero ed il grado di conoscenza di essi, in questo modo diedero origine ai cosiddetti **micromondi**.

*“Un micromondo è un dominio artificiale limitato, i cui possibili oggetti, le possibili proprietà e i possibili eventi sono definiti in anticipo in modo ristretto ed esplicito. Gli scacchi, per esempio, sono un micromondo: ci sono solo un numero limitato di pezzi e mosse lecite, e tutto è chiaramente definito dalle regole. A differenza di un vero condottiero medievale, uno stratega degli scacchi non deve mai preoccuparsi di epidemie, scomuniche, malcontento fra le file dei suoi guerrieri o della comparsa di cosacchi all'orizzonte: perché nulla può accadere in un micromondo che non sia permesso espressamente nella sua definizione.” (J. Haugeland, *Intelligenza Artificiale*, Bollati Boringhieri 1988, p. 173)*

Il principio strutturale essenziale di un micromondo è quello dell'estremo dettaglio con cui è generato, al fine di allontanarsi dall'estrema complessità del mondo reale per poi riavvicinarsi gradualmente. Terry Winograd fu il primo a realizzare un micromondo, cui attribuì il nome SHRDLU, e che era costituito da un insieme di solidi geometrici appoggiati su un tavolo.

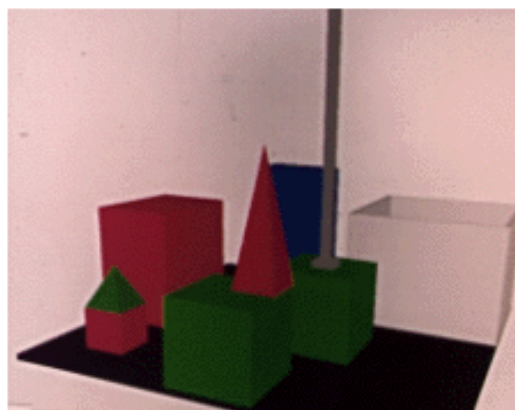
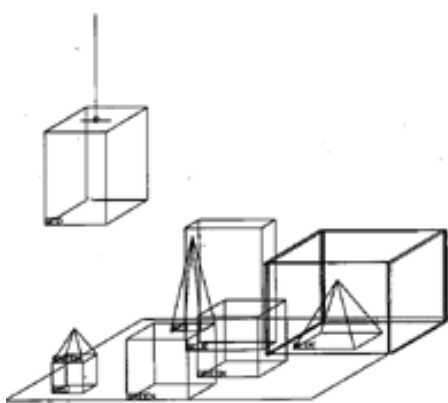


Figura 7. Porzione del Micromondo SHRDLU

Il programma conosceva perfettamente il suo mondo ed era in grado di intraprendere discussioni su di esso e perfino di agire al suo interno, modificando la struttura e la disposizione dei blocchi attraverso un braccio virtuale e fornendo le motivazioni del suo operato.

Interrogazione del Micromondo SHRDLU

Persona:	how many things are on top of green cubes? <i>[quante cose ci sono sopra i cubi verdi?]</i>
SHRDLU:	i'm not sure what you mean by "on top of "in the phrase "on top of green cubes" . Do you mean: 1 - directly on the surface 2 - anywhere on top of? <i>[non sono sicuro su cosa intendi con 'sopra' nell'espressione 'sopra i cubi verdi'. Intendi</i> <i>1- direttamente sulla superficie</i> <i>2- ovunque sopra ai cubi?]</i>
Persona:	2
SHRDLU:	three of them <i>[ci sono tre cose]</i>
Persona:	had you touched any pyramid before you put the green one on the little cube? <i>[avevi toccato qualche piramide prima di mettere quella verde sul cubo piccolo?]</i>
SHRDLU:	yes, the green one <i>[sì quella verde]</i>

SHRDLU in alcune occasioni mostra anche la capacità di imparare nuovi termini, se definiti dalla persona in maniera completa e in base a termini conosciuti dal programma, dimostrando una netta evoluzione rispetto ai programmi ad esso precedenti. L'efficienza di SHRDLU si basa sulla limitatezza del suo mondo e sulla sua estrema semplicità di funzionamento, che consente al software di ricordare ciò che accade al suo interno con particolare facilità.

2.6.4 Le reti semantiche

La creazione dei micromondi non risolse con efficacia i principali problemi derivanti dallo sviluppo di programmi intelligenti, anzi contribuì ad allontanare i ricercatori da quella che poteva essere una soluzione diretta alla conoscenza del mondo esterno.

Nel 1968 lo studioso Ross Quillian teorizzò il modello di *rete semantica*, che avrebbe facilitato l'organizzazione dei significati delle parole e l'associazione dei concetti. Il suo scopo era di consentire un elevato numero di inferenze tra le molteplici rappresentazioni lessicali fino ad arrivare alla simulazione delle capacità basilari della mente umana, come il confronto tra due parole o la comprensione di un testo. Il modello di Quillian si sviluppa secondo uno schema ramificato basato su due tipi di nodi: *nodo tipo* (*type node*) e *nodo esemplare* (*token node o token*). Il significato lessicale di ogni termine è rappresentato mediante un *type node*, cui corrisponde nella rete un *piano* (*plane*), la struttura che rappresenta la descrizione del relativo significato.

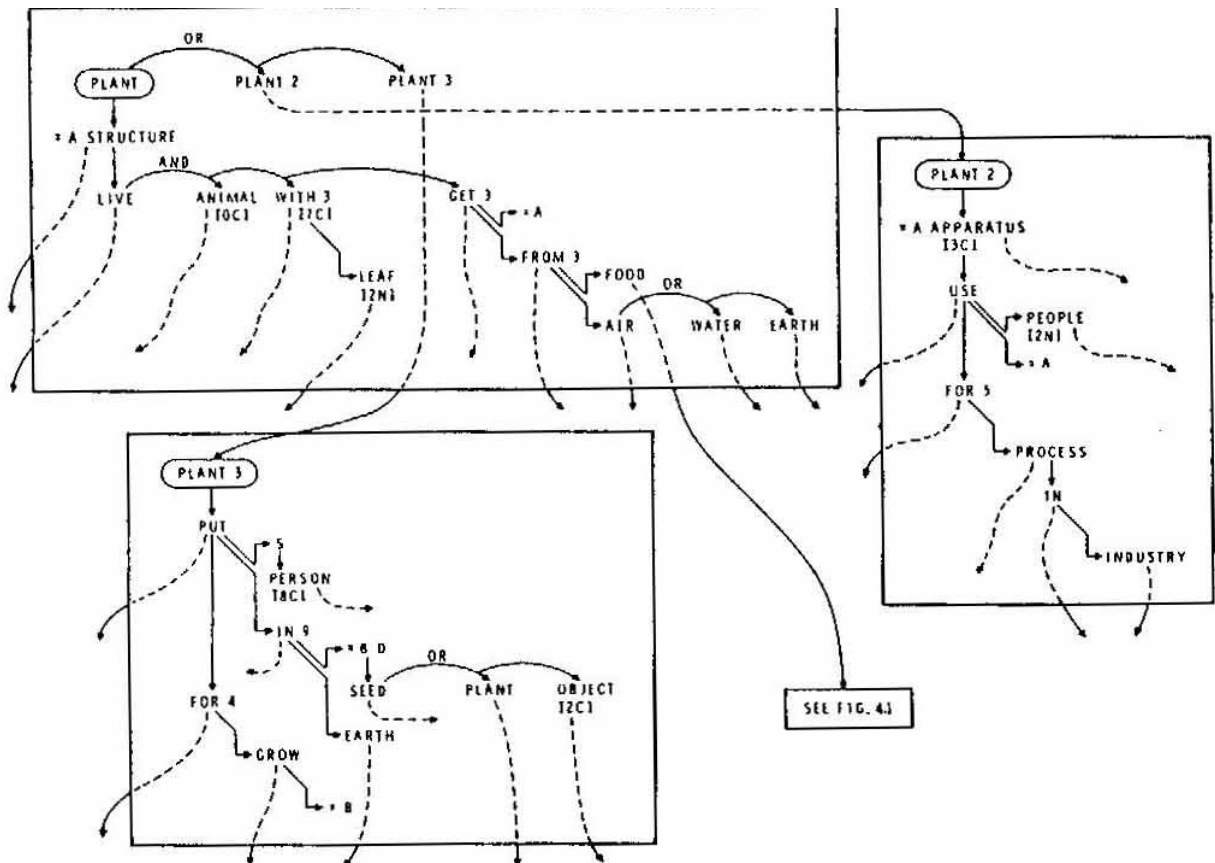


Figura 8. Esempio di rete semantica

I type node sono rappresentati attraverso ellissi. Dato che ad ogni voce lessicale possono corrispondere più significati, è facile vedere come ad ognuna possano corrispondere più type node. In figura alla parola PLANT corrispondono i tre nodi PLANT, PLANT 2 e PLANT 3, che conducono ai tre significati distinti (in italiano, "pianta", "impianto" e al verbo "piantare"). I rispettivi piani sono rettangoli che circoscrivono la parte di rete relativa a quella definizione, ottenuta con l'ausilio di ulteriori termini rintracciabili allo stesso modo nella rete, come avviene in un comune dizionario. I nodi token, raffigurati con archi di frecce tratteggiate, consentono questi riferimenti all'interno della rete. Di conseguenza, mentre ad ogni significato corrisponde un unico type node, non avviene lo stesso per i token, in quanto essi sono tanti quanti il numero di volte la voce viene impiegata in altre definizioni.

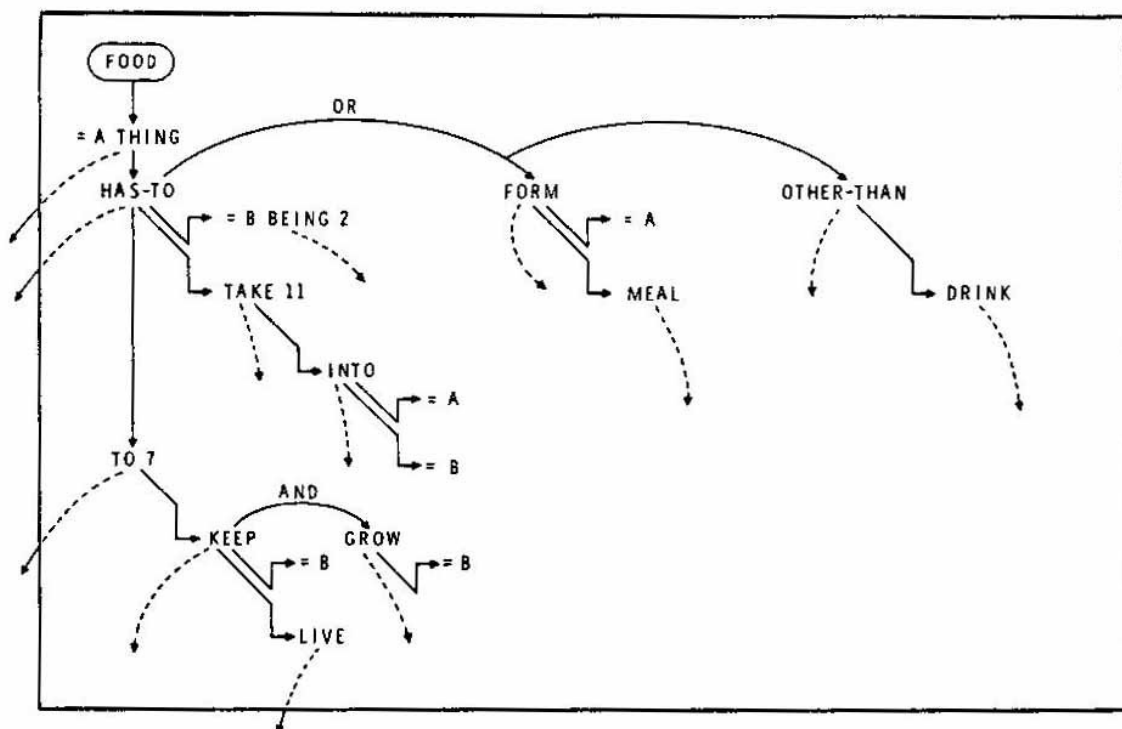


Figura 9. Esempio di rete semantica

La struttura di una rete semantica può divenire molto articolata, a seconda del numero di nodi creati e di collegamenti fra le voci presenti all'interno della struttura. La figura offre una definizione della parola FOOD attraverso una serie di connessioni:

- una relazione di sottoclasse, che lega il nodo FOOD al nodo THING(il cibo è un particolare tipo di cosa)
- un modification pointer, ovvero un arco che tramite token aggiunge proprietà al nodo(in questo caso lega THING e HAS-TO)
- archi che creano disgiunzioni fra nodi, al fine di rappresentare i molteplici significati attribuibili ad una voce(l'arco OR lega HAS-TO, FORM e OTHER-THAN)
- archi che creano congiunzioni(l'arco AND abbina KEEP e GROW)
- frecce doppie che rappresentano relazioni arbitrarie che esiste fra due token
- puntatori come =A e =B che fanno riferimento a nodi superiori, in modo da svolgere la funzione di pronomi o simboli sostitutivi

Visionata analiticamente l'intera voce è possibile parafrasarne una definizione precisa, come avviene per FOOD.

FOOD: è una **THING A** tale che un **BEING 2 B**(riferimento al secondo significato della parola being, così come per **TAKE 11**, undicesimo significato di take) la ingerisce per vivere e crescere, per formare un **MEAL** oppure tale che è **OTHER THEN** un **DRINK**.

Lo scopo del modello proposto da Quillian era di simulare la comprensione di enunciati espressi in linguaggio naturale e forniti come INPUT alla macchina, che attraverso un software specifico avrebbe percorso la rete semantica fino ad ottenere la rappresentazione completa del significato ricercato.

"nel modello della memoria il concetto completo di una parola è definito come l'insieme di tutti i nodi che possono essere raggiunti tramite un processo esaustivo di percorrimiento avente origine nel corrispondente type node patriarca, assieme alla somma totale delle relazioni fra questi nodi specificate da archi fra token e token di uno stesso piano" (Quillian 1968, p. 238).

Le prestazioni effettive sono in realtà molto limitate, in quanto il modello è in grado solamente di individuare le principali affinità e differenze riscontrate tra due voci lessicali, per poi esprimerle in un enunciato in un inglese ridotto e semplificato.

2.6.5 I frame di Minsky

Il modo estremamente rigido con cui le reti semantiche rappresentano una conoscenza regolata da collegamenti obbligatori non si presenta adeguato all'estrema versatilità di ragionare di un essere umano. Per comprendere un discorso in linguaggio naturale e riconoscere una determinata situazione è necessaria un'estrema rapidità con cui raccogliere informazioni dalla memoria ed utilizzarle in maniera corretta.

La seguente frase

(a) “L’uomo entrò in ufficio ed iniziò a scrivere.”

è di facile comprensione per ogni essere umano, infatti essa implica il possesso di un certo numero di informazioni da parte del lettore, come l’implicita presenza di una penna all’interno di un ufficio. La memoria umana in un intervallo infinitesimo analizza la parola “ufficio” ed associa ad essa una struttura ben più ricca del semplice significato lessicale.

D’altra parte la frase

(b) “L’uomo entrò in ufficio per annotare l’appuntamento, ma non trovò nulla per scrivere.”

non presenta alcun errore, né può indurre ad alcuna obiezione. Il concetto di ufficio prevede di norma la presenza di strumenti per scrivere al suo interno, ma non è indispensabile per la comprensione del testo.

Lo studioso Marvin Minsky fu uno dei primi a tentare di dotare un computer di questa capacità mettendo a punto un nuovo modello teorico, basato questa volta sul concetto di *frame* (in inglese significa cornice).

Un frame è una struttura che organizza in maniera organica tutte le informazioni necessarie, probabili ed opzionali utili a definire un particolare concetto.

Ufficio

Sottoclasse di: **stanza**
Funzioni (sempre): **lavorare, leggere, scrivere**
Funzioni (probabile): **conversare, usare il computer**
Ha come costituenti (sempre): **pareti, finestre, soffitto, pavimento**
Contiene (sempre): **scrivania**
Posizione(quasi sempre): **vicino alle finestre**
Contiene (probabilmente): **computer**
Posizione (probabilmente): **sulla scrivania**
Etc.

Il frame ipotizzato da Minsky dovrebbe contenere un numero assai maggiore di informazioni con un ancor più precisa determinazione delle proprietà degli elementi contenuti e delle relazioni esistenti tra di essi.

Un sistema dotato di frame, ricevendo come input la parola “ufficio”, richiama rapidamente il frame corrispondente e con il sopraggiungere di nuove informazioni inizia un confronto sulla base delle caratteristiche conosciute fino a pervenire ad una conferma o ad una possibile contraddizione con conseguente cambio di frame. Un ulteriore aspetto fondamentale del modello basato sui frame è la loro interconnessione: il componente “scrivania” relativo al frame ufficio è a sua volta collegato col frame scrivania. Una struttura di questo tipo favorisce la formazione di una rete di concetti, in grado di condurre ragionamenti semplici come la deduzione ma anche più complessi quali l’induzione o l’analogia.

2.6.6 Gli script di Schank ed Abelson

Un ulteriore modello, che ben si affianca a quello dei frame, è quello elaborato da Roger Schank e dal suo collaboratore Abelson, basato sulla nozione di *script*, in grado di riconoscere eventi tipici in situazioni particolari.

La serie di frasi

(a) “Marco si recò in biblioteca. Prese il libro di fisica e lo aprì. Dopo due ore lo richiuse e ne aprì un altro.”

rappresenta una breve storiella di facile comprensione. Chiunque capirebbe che Marco per due ore ha studiato fisica, sebbene nel testo non si stato espressamente specificato. Ciò avviene perché ognuno possiede una certa quantità di informazioni legate a questo contesto che consentono di comprendere pienamente la situazione presa in esame. Schank fu il primo a pensare di poter condensare questi dati nella struttura dello script per consentire ad un sistema di acquisire un buon livello di conoscenza della realtà.

“... uno script è un insieme di scanalature pronte a ricevere certi eventi e non altri. Lo script usato per comprendere qualcosa ci aiuta a sapere che cosa aspettarci. Uno script dice che cosa probabilmente seguirà in una catena di eventi stereotipati. Non solo, ci permette di capire la rilevanza di ciò che di fatto succede subito dopo: garantisce il collegamento fra gli eventi.”(R. C. Schank, *Il computer cognitivo*, Giunti 1989, p. 123.)

Il concetto di script, pur essendo molto simile a quella di frame, se ne differenzia per l'accento posto sulla successione degli eventi piuttosto che sulle proprietà degli oggetti.

Lo script specifica con estremo dettaglio le situazioni, indicando i luoghi, i tempi ed i personaggi coinvolti, non tralasciando al tempo stesso le cause ed i risultati di ogni avvenimento. Schank in alcuni casi suddivide uno script in diversi sotto-script (definiti "binari" con cui arricchisce ancor più di dettagli i diversi componenti dello script iniziale.

Esempio di script della "biblioteca":

Script:	Biblioteca
Oggetti:	Scaffali Libri Scrivanie Sedie
Personaggi:	Bibliotecaria Studente
Condizioni di entrata:	Lo studente necessita di informazioni
Risultati:	Lo studente ha ottenuto le informazioni ricercate

Scena 1: <u>Ingresso</u> Entrare in biblioteca Chiedere informazioni sul libro cercato alla bibliotecaria Ricevere informazioni sul libro cercato	
(libro presente in biblioteca) Scena 2	(libro non presente in biblioteca) Uscire dalla biblioteca

Scena 2: <u>Recupero del libro richiesto</u> Recarsi nel settore indicato Individuare lo scaffale indicato Recuperare il libro richiesto
--

Scena 3: <u>Ricerca del posto di lettura</u> Entrare nella sala lettura Guardare le scrivanie Decidere dove sedersi Controllare che il posto sia libero Andare alla scrivania Mettersi a sedere
--

Scena 4: <u>Lettura</u> Lettura e analisi delle informazioni	
(le informazioni ottenute sono state esaurienti) Memorizzare le informazioni cercate Scena 5	(le informazioni ottenute non sono state esaurienti) Scena 1

Scena 5: Abbandono delle biblioteca

Riposizionare del libro sul relativo scaffale

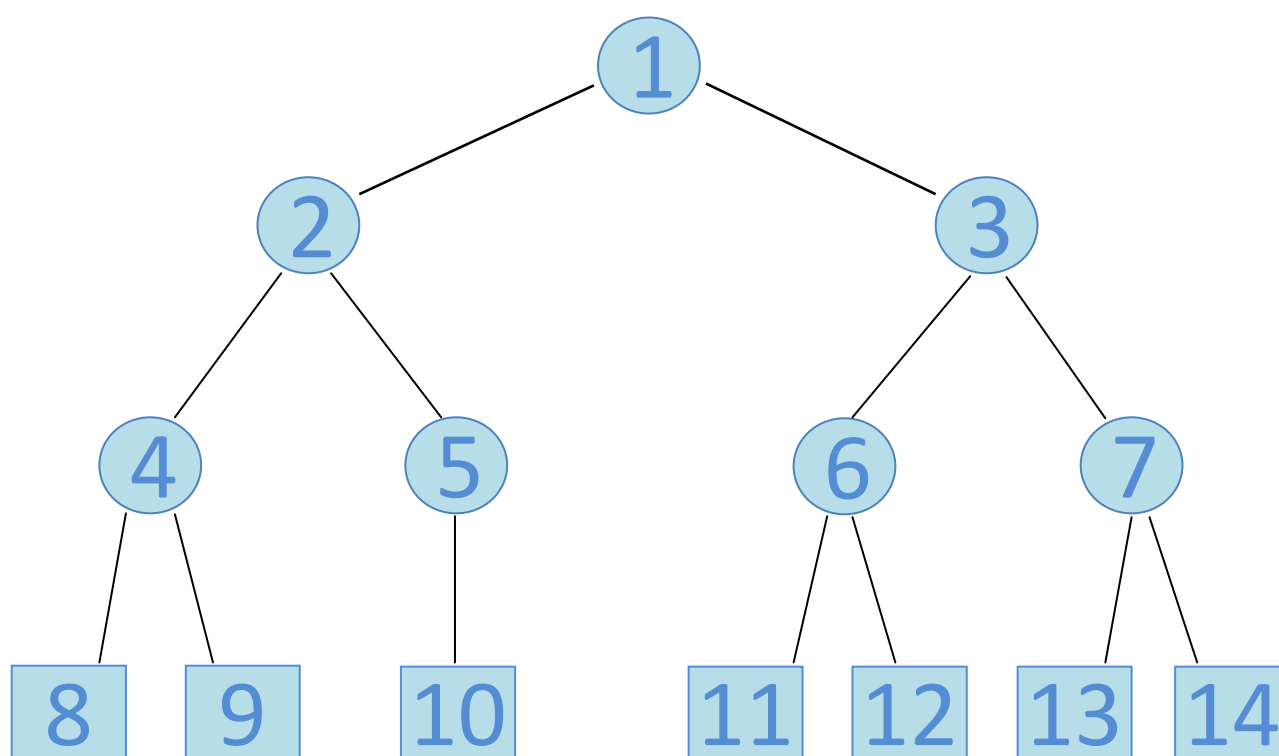
Comunicare alla bibliotecaria l'attuale stato del libro

Uscire dalla biblioteca

2.7 Problem Solving

Le numerose difficoltà riscontrate dalla società moderna nell'affrontare procedimenti a difficoltà crescente riguardanti la *risoluzione di problemi (problem solving)*, la *dimostrazione automatica* e la *ricerca selettiva* hanno spinto all'applicazione di determinati modelli a sistemi informatici basati sull'intelligenza artificiale.

Gli algoritmi utilizzati, spesso molto complessi, seguono differenti tipi di approccio al problema da risolvere: quello *sistematico*, quello *euristico* e quello *pianificato*.



Schema 4. Ricerca sistematica

Le due principali strategie di attuazione della ricerca sistematica sono la *ricerca in profondità (depth-first)* e la *ricerca in ampiezza (breadth-first)*. Attraverso la ricerca in profondità ogni percorso viene seguito (a partire da sinistra) fino al punto finale. Nel caso in cui la soluzione non sia stata trovata si ritorna alla precedente biforcazione per poi ripetere l'operazione di ricerca.

Seguendo il grafico, il primo percorso esplorato è 1-2-4-8. In caso di esito negativo si torna al punto 4 per poi seguire 4-8, allo stesso modo, in caso di mancata soluzione, si ritorna questa volta al punto 2 per poi esplorare 5-10 e così via. Questo tipo di ricerca richiede una gran quantità di tempo, soprattutto nel caso in cui la soluzione si trovi in uno dei nodi più in alto a destra o se uno dei percorsi procede all'infinito, ma in altre situazioni può risultare ugualmente improduttiva la ricerca in ampiezza, in particolare se un nodo è posto molto in basso.

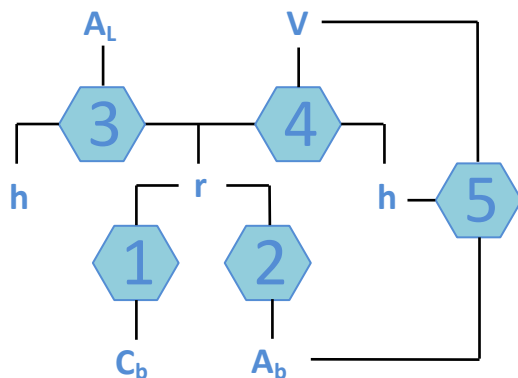
Molto più utilizzato è l'approccio euristico, basato sul tentativo diretto delle possibili strade con la memorizzazione degli errori commessi. Questo metodo, pur essendo comunque lento e di non sicura riuscita, offre almeno una quasi completa conoscenza della situazione attraverso i vari tentativi. Un esempio semplice ma efficace è quello di un robot che cerca la via di uscita da un labirinto provando le varie strade possibili e tornando indietro in caso di vicolo cieco.

L'approccio pianificato, a differenza dei primi due, necessita di una complessa elaborazione preventiva, che può essere comunque modificata in corso di risoluzione nel caso che la situazione dovesse mutare.

Attraverso un modello di questo genere è possibile risolvere un qualsiasi semplice problema di geometria solida. Considerata una figura solida, quale può essere un cilindro, gli si assegna una raccolta di regole di base.

1. Il raggio del cerchio di base è calcolabile attraverso la misura della circonferenza di base ($C_b \rightarrow r$)
2. Il raggio del cerchio di base è calcolabile attraverso la misura area di base ($A_b \rightarrow r$)
3. L'area laterale è calcolabile attraverso la misura del raggio di base e la misura dell'altezza ($r, h \rightarrow A_L$)
4. Il volume è calcolabile attraverso la misura del raggio di base e l'altezza ($r, h \rightarrow V$)
5. Il volume è calcolabile direttamente attraverso l'area di base e la misura dell'altezza ($A_b, h \rightarrow V$)

Volendo calcolare la superficie laterale ed il volume da una situazione di partenza, in cui sono disponibili area di base o circonferenza di base ed altezza è possibile eseguire un *Backward-Chaining*, secondo lo schema rappresentato in figura.



Lo schema consiste nella successione delle operazioni, nel verso che va dalle foglie alla radice:

- 1 – 3 o 2 – 3 (Area laterale)
- 1 – 4, 2 – 4 o 5 (Volume)

Un algoritmo che ha svolto un ruolo fondamentale nella storia del settore è lo STRIPS, il quale è strutturato sulla base di tre liste di asserzione:

- i “PRE”, pre-requisiti necessari per applicare l’operatore
- gli “ADD”, asserzione da aggiungere allo stato attuale
- i “SUB”, asserzione da togliere dallo stato

Il codice dello STRIPS è in grado di percorrere la strada a ritroso alla ricerca degli operatori che hanno nella lista ADD le asserzioni dello stato finale (S_n). Lo stato precedente ad esso è di conseguenza così strutturato

$$S_{n-1} = S_n - ADD + SUB$$

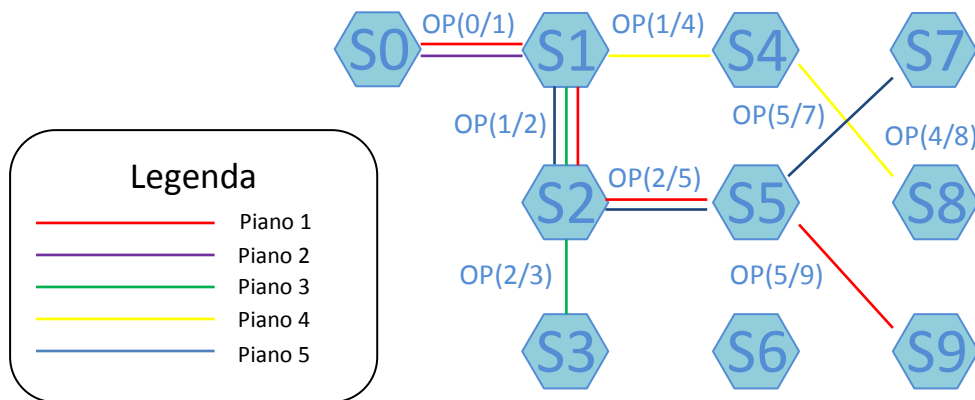
Comunque l'operatore considerato può essere applicato solo se S_{n-1} contiene i suoi PRE. Allora, S_{n-1} viene corretto in

$$S_{n-1} + PRE$$

A questo punto la ricerca ricomincia, trattando S_{n-1} come ultimo stato. In ogni istante della ricerca possono essere presi in esame più operatori, imponendo la ricerca in tutti i rami coinvolti, ma allo stesso tempo dando precedenza a quelli che minimizzano la distanza dallo stato iniziale.

Infine, individuata la successione S_n, S_{n-1}, \dots e la lista degli operatori $OP_{n-1/n}, OP_{n-2/n-1}, \dots, OP_{2/1}$, che li collegano, il piano consiste in questa ultima lista invertita: $OP_{2/1}, OP_{3/2}, \dots, OP_{n-2/n-1}, OP_{n-1/n}$.

I pianificatori riescono anche ad apprendere dall'esperienza attraverso il processo del *CHUNKING*, ovvero nella creazione automatica di macro-operatori, che lega tratti del percorso risolutivo comuni a molti piani, in precedenza separati.



S_1 ed S_5 sono collegati, mediante i sottopiani $OP_{1/2}$, $OP_{2/5}$, e lo stesso avviene per altri punti. Pertanto può essere realizzato un macro-operatore $MOP_{1/5}$, tale che se si dovesse effettuare di nuovo nel seguito il piano numero 5, esso sarebbe molto più semplice: $OP_{1/2}$ ed $OP_{2/5}$.

2.8 I sistemi esperti

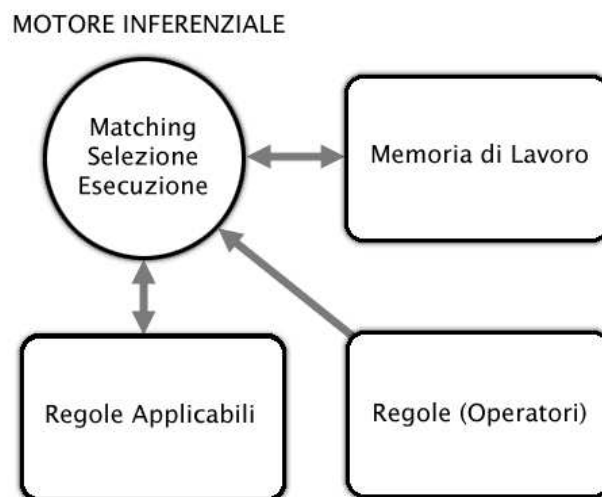
I sistemi esperti rappresentano probabilmente al momento la più importante applicazione a livello commerciale dell'intelligenza artificiale. Essi trovano impiego in particolare in quei settori dove la conoscenza non è ben strutturata ed un algoritmo non riesce a descrivere i processi decisionali:

- pianificazione o configurazione di sistemi di elaborazione
- diagnosi di malattie
- interpretazione di dati forniti da sensori
- sorveglianza di sistemi complessi
- ricerca di guasti in un sistema

Uno dei più brillanti risultati di un sistema esperto è stato l'individuazione di un giacimento di molibdeno nello stato di Washington. Eventi come questo hanno permesso alla ricerca nel settore dell'intelligenza artificiale di occupare uno dei primi posti in campo economico, discostandola dall'alone puramente teorico che lo aveva sempre coperto.


L'operatività di un sistema esperto è paragonabile all'efficienza di uno specialista dello stesso settore, infatti attraverso particolari strutture differenti dai tradizionali algoritmi la conoscenza rappresentata al suo interno può incrementarsi fino a diventare disponibile per chiunque in ogni occasione. La struttura di un sistema di questo tipo è basata su due elementi fondamentali:

- una *base della conoscenza*, ovvero una conoscenza specifica basata sui fatti e sulle regole da applicare
- un *motore inferenziale* necessario a trarre conclusioni a partire dai dati di ingresso in base anche all'eventuale mancanza di certezza di singole regole



Schema 5. Struttura di un sistema esperto

La conoscenza è rappresentata attraverso il modello dei frame, raggruppati in domini di conoscenza e contenenti informazioni collegate semanticamente in modo da consentire al programma una ricerca più rapida. Il motore inferenziale è l'elemento fondamentale che consente di esaminare le effettive basi della conoscenza per verificare se le regole sono soddisfatte al fine di trarne le relative conclusioni. Un'ulteriore struttura appartenente all'architettura di un sistema esperto è il sistema di spiegazione, attraverso il quale è possibile ricavare le informazioni che hanno condotto alla soluzione, permettendo di verificare la sua plausibilità. I gusci shell sono invece sistemi esperti con la base della conoscenza vuota, in modo da essere strumenti utili allo sviluppo di sistemi simili in campi di applicazione diversi.



Un aspetto fondamentale dei moderni sistemi esperti è che essi spesso raggiungono e superano le capacità degli umani, essendo dotati delle conoscenze di un elevato numero di specialisti di alto livello. Il programma Internist/Caduceus, sviluppato dall'Università di Pittsburg, raccoglie più dell'80% delle conoscenze di medicina interna.


Altri storici esempi di sistemi esperti possono essere identificati in:

- **MACSYMA**, un esperto di matematica, realizzato nel '68, è in grado di effettuare differenziazioni ed integrazioni simboliche meglio degli specialisti umani.
- **DENDRAL**, progettato da Feigenbáum nel 1969, determina la struttura molecolare a partire dai dati forniti da uno spettrografo di massa, superando in molti casi le capacità di molti esperti di chimica
- **MYCIN**, sviluppato all'inizio degli anni '70, riesce a produrre complete diagnosi mediche per determinate infezioni batteriche e fornisce proposte di terapie con relativo dosaggio di antibiotici. Le sue capacità sono ben al di sopra di quelle di un comune medico e possono essere paragonate solo a quelle di alcuni specialisti universitari
- **PROSPECTOR**, impiegato in ambito industriale per la ricerca di giacimenti di minerali strategici, si è distinto per aver scoperto l'importante giacimento di molibdeno nello stato di Washington, che fu valutato oltre 100 milioni di dollari.

L'utilizzo di questi sistemi automatizzati è stato necessario per garantire un minor costo alle operazioni, oltre ad una riduzione degli eventuali ritardi e del numero degli errori inevitabili nel caso di una esecuzione manuale. Allo stesso modo però non si sono evitati molti aspetti negativi, legati principalmente ai limiti del sapere degli esperti, che hanno obbligato ad una semplificazione dei risultati ottenuti, nonché all'impossibilità di condensare alcuni settori in una teoria di facile utilizzo da parte di elaboratori.

2.8.1 Creazione di un sistema esperto

Iniziando da un guscio di partenza (*shell*), ovvero da un sistema esperto ancora privo di informazioni ma dotato di una struttura completa, è possibile affinarlo con una serie di procedimenti in grado di renderlo un sistema esperto dedicato. L'obiettivo essenziale da perseguire è quello di sviluppare la base della conoscenza, ovvero il settore costituito



principalmente dalle regole e dai fatti, per poi occuparsi dei componenti aggiuntivi che consentono la manipolazione e la verifica dei dati, il dialogo con l'utilizzatore e la gestione delle eventuali spiegazioni delle decisioni.

Gli elementi che influenzano necessariamente il processo di creazione di un sistema esperto sono:

- l'ingegnere della conoscenza, colui che si occupa della gestione della conoscenza del sistema
- l'esperto umano
- l'organizzazione impiegata dal sistema
- il settore specifico di applicazione


L'ingegnere della conoscenza deve acquisire la conoscenza dal notevole patrimonio di esperienza dell'esperto umano, rappresentarla ed inserirla in un programma, dopo la sua ottimizzazione e una precisa definizione degli obiettivi da raggiungere.

Le fasi essenziali dello sviluppo di un sistema esperto sono quattro:

1. Scelta dell'architettura e del modo di rappresentare la conoscenza in modo da adeguarla alla concezione del programma
2. Realizzazione di una serie di prototipi (rapid prototyping) della conoscenza tradotta in una versione funzionante da aggiornare in caso di correzioni o suggerimenti dell'esperto
3. Prova del sistema con l'aiuto di eventuali altri esperti, che giudicano la qualità delle soluzioni e delle risposte fornite
4. Trasferimento del sistema alla macchina definitiva, frequentemente a una macchina LISP del tipo single-user, ed in seguito all'ambiente di impiego futuro

La fase di acquisizione della conoscenza è solitamente affrontata con l'impiego di diversi metodi, tra cui il più usato è quello di fare colloqui e porre domande all'esperto, ma è già in fase di sperimentazione la somministrazione al sistema di un insieme di casi da risolvere.

La realizzazione del prototipo consente di effettuare i ritocchi finali prima dell'inizio della produzione in serie ed è necessario soprattutto quando i requisiti del software non possono essere



definiti senza uno studio approfondito del problema. La tecnica del prototipo veloce offre molti vantaggi, infatti oltre a fornire una proficua interazione tra l'utilizzatore e l'ingegnere della conoscenza, consente di giungere gradualmente alla realizzazione di un sistema esperto complesso attraverso una serie di procedure di affinamento delle sue componenti.

Allo stesso modo si rivela necessaria una valutazione dei sistemi esperti in fase di costruzione, sebbene non esistano criteri fissi per attuare questa procedura. In buona parte dei casi si considera la percentuale dei problemi risolti correttamente dal sistema, ma un adeguato processo di verifica va effettuato considerando alcune questioni aperte:

- La forma scelta per la rappresentazione della conoscenza è adeguata alla risoluzione dei problemi oppure può essere vantaggiosamente modificata?
- La conoscenza immessa nel sistema corrisponde a quella dell'esperto?
- Il sistema pone le domande secondo una corretta tempistica?
- Le risposte e le spiegazioni fornite sono coerenti?
- L'interfaccia utente è di facile utilizzo?
- Cosa manca per completare il sistema?

Allo stesso modo, dopo la realizzazione del primo prototipo o della prima versione del sistema bisogna chiedersi se la velocità di risposta del sistema è adatta al settore in cui viene impiegato e se il rapporto spesa/guadagno è favorevole.

Attualmente è facile trovare sistemi esperti adatti ad un Personal Computer basati su meno di 1000 regole, ma la vasta panoramica di scelte impone una maggiore esperienza da parte dell'acquirente. L'attuale stato delle ricerche ha permesso di adattare sistemi di questa portata a settori come il collaudo degli strumenti, la simulazione del volo e a varie applicazioni nel campo dei circuiti. Inoltre la possibilità di riadattarli ad altri settori senza dover toccare il codice, ma semplicemente cambiando la conoscenza di base, ha fatto sperare di potersi addentrare nella comprensione dei linguaggi naturali.

2.9 Le reti neurali

2.9.1 Limiti nello sviluppo di algoritmi

Gli studi sull'intelligenza artificiale, giunti al massimo livello evolutivo con lo sviluppo dei più avanzati sistemi esperti, nel tentativo di allontanarsi da quel modo di ragionare ancora troppo primitivo ed incomparabile con i processi percettivi umani, non potevano non incontrare dei limiti invalicabili sfruttando unicamente i basilari modelli di calcolo simbolico. Una qualsiasi macchina "intelligente" non sarebbe mai stata in grado di descrivere un'immagine, localizzando e riconoscendo oggetti significativi, dato che questa procedura avrebbe richiesto capacità di segmentazione basate sulle variazioni di luminosità, oltre a particolari conoscenze sugli oggetti analizzati da molteplici punti di vista nel mondo tridimensionale. Allo stesso modo diventa estremamente complesso per un sistema riuscire a discernere le parole di un discorso ascoltato a causa della variabilità dovuta alla velocità di pronuncia, alla prosodia, al parlatore e alle differenti condizioni di rumore. È molto difficile, se non impossibile, per un algoritmo riuscire ad interpretare queste strutture sotto-simboliche e fornire loro una chiara caratterizzazione simbolica. Il cervello umano si è rivelato l'unico sistema, frutto di millenni di processi evolutivi, in grado di aggirare facilmente questi ostacoli.

2.9.2 Il cervello umano

Il cervello umano è certamente una delle strutture più complesse esistenti in natura. Costituito da circa un centinaio di miliardi di cellule (neuroni) collegati a loro volta da milioni di miliardi di connessioni, è giustamente considerato un'estesa *rete neurale*. Un neurone è costituito da un corpo cellulare centrale (soma), sul quale si innestano numerose diramazioni (*dendriti*), che ricevono i segnali, e da cui parte un

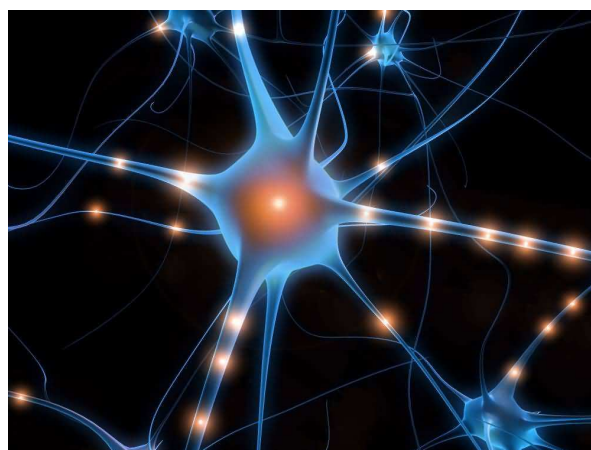


Figura 10. Neurone biologico

collegamento in uscita (*assone*), che a sua volta si dividerà nei corrispondenti dendriti.

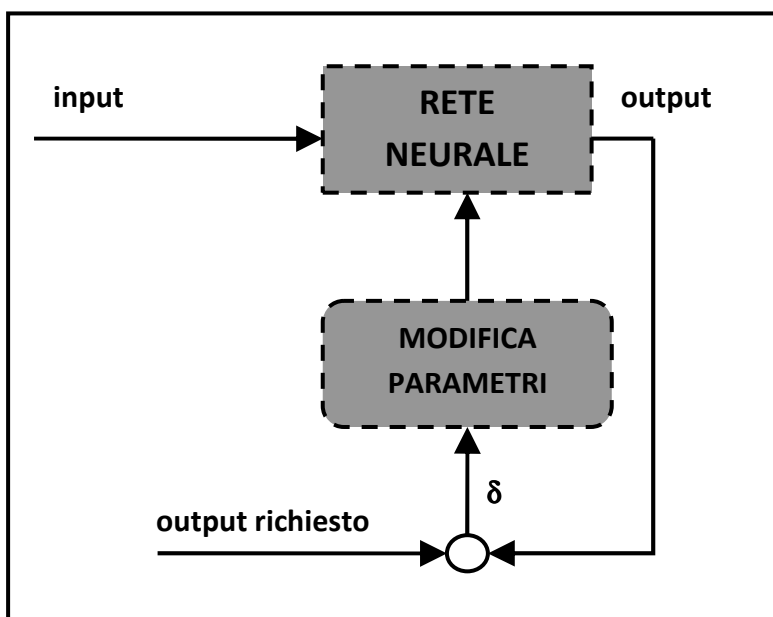
Le interconnessioni avvengono tramite le cosiddette *sinapsi*, che permettono il passaggio delle informazioni attraverso un processo elettrochimico basato su impulsi nell'ordine di alcune decine

di mV (milliVolt) con una frequenza di circa 100 Hz e le opportune modulazioni. La struttura del cervello è influenzata dalla vita dell'individuo, in particolar modo dalla sua esperienza e dal suo apprendimento, che consentono la creazione di ulteriori connessioni e l'attribuzione di compiti specifici a particolari zone, che possono essere perse a causa dell'eventuale rottura dei legami sinaptici. A seguito dei numerosi studi operati appunto sulla struttura cerebrale umana, gli studiosi hanno pensato di poter trarre suggerimenti utili per la costruzione di macchine dotate di una forte capacità di elaborazione sotto-simbolica, rifacendosi al meccanismo di indebolimento e rafforzamento di un collegamento sinaptico.

2.9.3 Le reti neurali artificiali


Lo studio del comportamento intelligente delle reti neurali biologiche, generate dal gran numero di connessioni fra cellule, ha consentito lo sviluppo di modelli matematici utili alla realizzazione di *reti neurali artificiali*. Le caratteristiche di questi modelli estremamente evoluti sono molto innovative e si articolano a partire da due concetti di base:

- Capacità di apprendimento da un certo numero di esempi
- Capacità di configurare i propri parametri di funzionamento per adeguarsi agli stimoli esterni



Schema 6. Funzionamento di una rete neurale

I principi basilari di funzionamento di una rete neurale sono totalmente differenti da quelli utilizzati nell'intelligenza artificiale classica per la gestione dei dati e delle decisioni, infatti le informazioni vengono scomposte in unità elementari contenute all'interno di ogni singolo *neurone artificiale*.



La combinazione input\output non si ottiene attraverso un procedimento di programmazione, ma attraverso un complicato processo di apprendimento basato essenzialmente sulla presentazione di esempi di coppie input\output. Il sistema in questa maniera può facilmente comprendere la relazione esistente tra i dati in ingresso e quelli in uscita ed applicarla in ogni circostanza. Nella maggior parte dei casi ogni output fornito dalla rete si differenzia di una certa quantità δ dal valore richiesto, di conseguenza è utile fornire un elevato numero di esempi per avvicinare progressivamente i parametri della rete ai valori ottimali.

I parametri considerati sono i collegamenti esistenti tra i neuroni che compongono la rete, proprio come la struttura nervosa che forma il tessuto cerebrale umano. Tuttavia non sono ancora ben chiari i meccanismi di apprendimento del cervello, ma le reti neurali con i propri algoritmi di apprendimento dedicati riescono a ricreare caratteristiche sorprendentemente simili a quelle umane come una spiccata elasticità di interpretazione. Un sistema basato su una rete neurale riesce a dare una risposta abbastanza corretta anche se gli viene fornito un input limitato o impreciso.

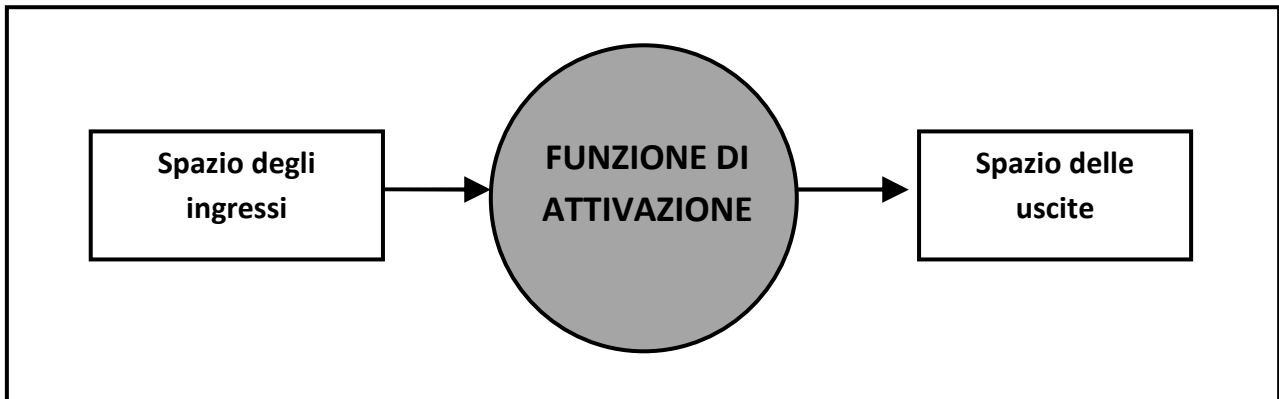
2.9.4 Caratteristiche di una rete neurale

1. La struttura è basata su un certo numero di neuroni
2. Ogni neurone deve possedere ingressi ed uscite ed una funzione di trasferimento
3. Gli ingressi e le uscite devono essere collegati attraverso collegamenti sinaptici
4. Esistenza di una precisa legge di apprendimento

Alcune reti neurali non presentano tutte queste caratteristiche, ad esempio *“il vicinato di Von Newman”*, con una disposizione a griglia, evolve modificando un parametro della funzione di trasferimento di ogni neurone sulla base delle attivazioni dei neuroni nelle vicinanze. Altre reti invece sono studiate per risolvere problemi di ottimizzazione combinatoria imponendo alcuni vincoli e tentando di minimizzare il costo energetico.

2.9.5 Neuroni artificiali

Con il termine neurone artificiale si intende un particolare modello matematico in grado di calcolare una determinata funzione, chiamata *funzione di attivazione*, i cui *ingressi* corrispondono agli stimoli ricevuti dal neurone biologico, mentre i valori di *uscita* descrivono i segnali trasmessi lungo l'assone.



Schema 7. Funzionamento di una neurone artificiale

I valori che formano lo spazio degli ingressi vengono trasformati nello spazio delle uscite secondo i parametri del neurone che regolano la soglia di reazione ed il rafforzamento della connessione sinaptica, che regolano rispettivamente il comportamento di una singola cella e l'interazione tra coppie.

Nel 1943 McCulloch e Pitts teorizzarono il primo modello formale di neurone artificiale molto prima che gli elaboratori informatici iniziassero a diffondersi. La struttura di questo iniziale modello matematico era alquanto semplice. La funzione di attivazione può assumere due valori:

- Neurone attivo, 1
- Neurone silente, 0

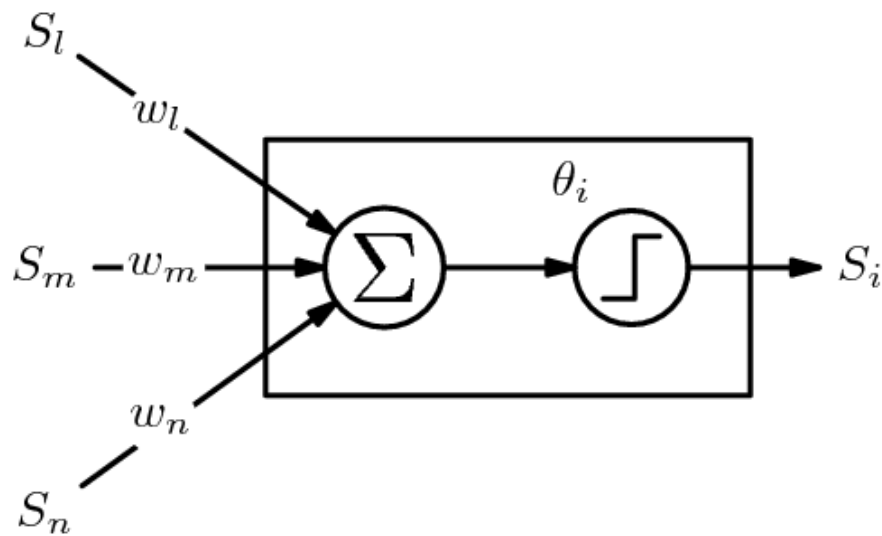
Allo stesso modo i *pesi sinaptici* che regolano la forza della connessione esistente tra una coppia di neuroni possono essere:

- Eccitatorio, 1
- Inibitorio, -1

Il neurone calcola la somma pesata degli stati dei neuroni ad esso connessi ed assume lo stato attivo o passivo se tale valore supera oppure no la soglia stabilita. La seguente funzione di attivazione descrive lo stato del neurone i -esimo:

$$S_i = \begin{cases} 1, & \sum_j w_{ij} S_j \geq \theta_i \\ 0, & \sum_j w_{ij} S_j < \theta_i \end{cases}$$

dove S_j è lo stato del neurone j -esimo, w_{ij} è il peso del contributo dell'uscita del neurone j al neurone i , e θ_i è la soglia del neurone i -esimo. Le sommatorie sono estese all'insieme di neuroni che sono connessi al neurone considerato, i .



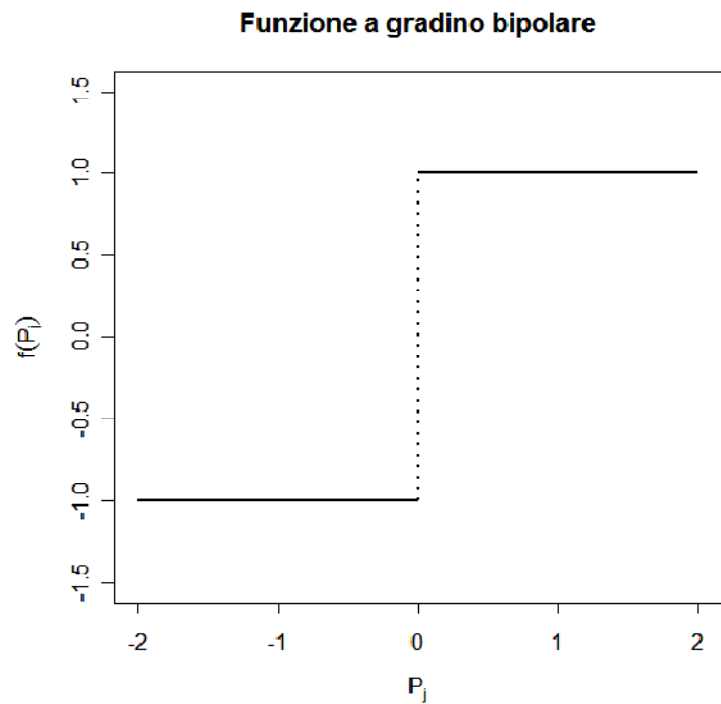
Il modello ottenuto è una particolare semplificazione del neurone biologico, che presenta ulteriori complessità nella sua struttura, come la capacità di integrare nel tempo i contenuti asincroni, anziché basarsi semplicemente sulla loro somma. L'uscita è del tipo tutto-o-niente e non è modulata nel tempo, ma riesce ugualmente a calcolare qualsiasi funzione binaria.

McCulloch e Pitts hanno comunque formulato diverse varianti del loro modello iniziale, definito *a soglia* o *a gradino*), sfruttando molte tipologie di funzioni di attivazione.

Ponendo $x = \sum_j w_{ij} S_j - \theta_i$ si ottiene il seguente catalogo di funzioni:

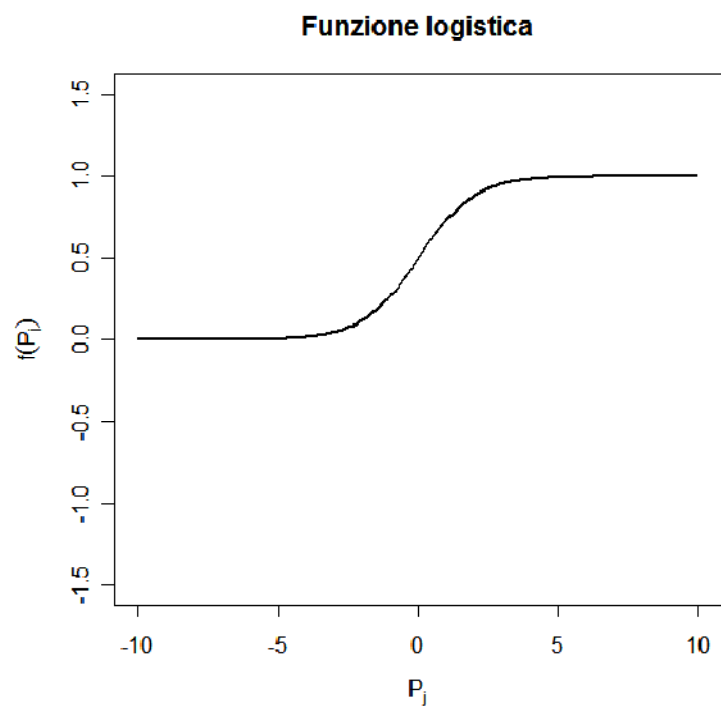
- *Gradino (bipolare)*

$$f(P_j) = \begin{cases} 1, & P_j \geq 0 \\ -1, & P_j < 0 \end{cases}$$



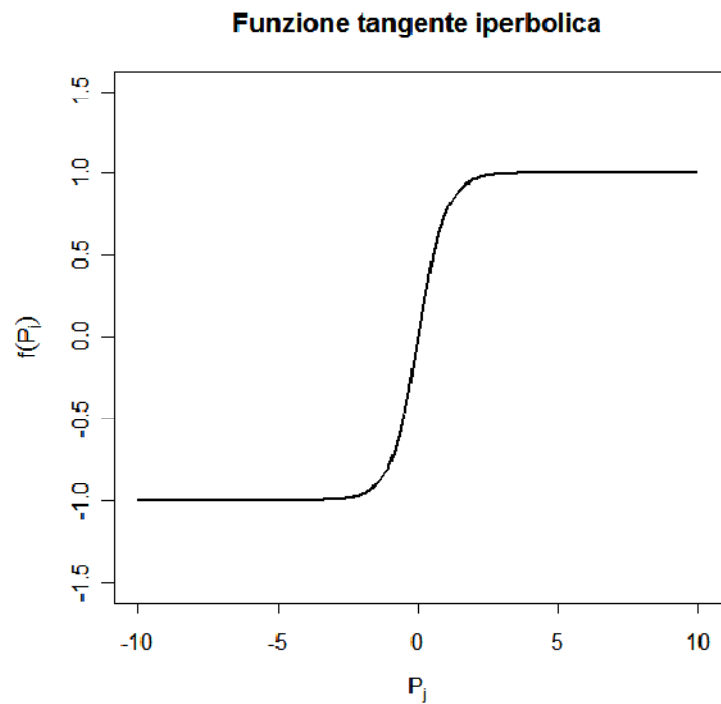
- *Sigmoide*

$$f(P_j) = \frac{1}{1 + e^{-x}}$$



- *Tangente iperbolica*

$$f(P_j) = \tanh x$$



- *Lineare*

$$f(P_j) = x$$

$$f(P_j) = \alpha + \beta P_j$$

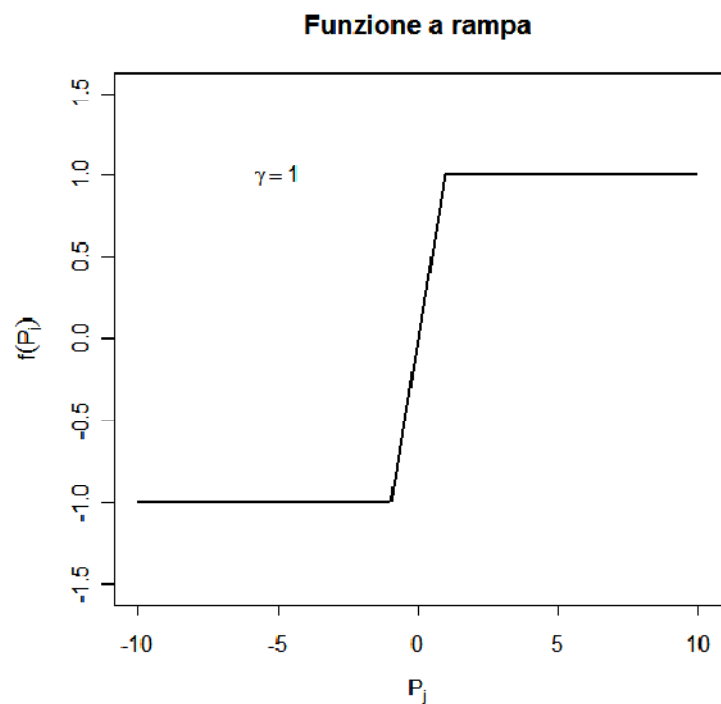
$$\alpha = 0$$

$$\beta = 1$$

per ottenere l'IDENTITA'

- *Lineare a tratti (rampa)*

$$f(P_j) = \begin{cases} \gamma, & P_j \geq \gamma \\ P_j, & -\gamma < P_j < \gamma \\ -\gamma, & P_j \leq -\gamma \end{cases}$$



Tutte le funzioni considerate, ad esclusione di quella lineare, sono generalizzazioni della funzione gradino, infatti aggiungendo un parametro di modifica della pendenza è facile ritornare alla funzione di partenza.

Alcuni neuroni biologici, come quelli specializzati della corteccia visiva, sono dotati della proprietà della località, infatti riprodotti artificialmente sono in grado di produrre una risposta significativa solo in un intorno di un punto dello spazio degli ingressi. Il punto considerato è definito dal parametro c_j , *centro* del neurone, che rappresenta la posizione del neurone nello spazio di ingresso. L'ampiezza dell'intorno di risposta può essere modulata dal parametro σ , detto *fattore di scala*. La risposta del neurone a uno stimolo è proporzionale alla *distanza*, r , tra lo *stimolo*, s , applicato al neurone e il suo centro, c_j : $r = \|s - c\|$.

A questi neuroni possono essere efficacemente applicati modelli basati su funzioni a simmetria radiale:

Gaussiana:

$$h(x) = e^{-\frac{(x-c_j)^2}{\sigma^2}}$$

Multiquadratica:

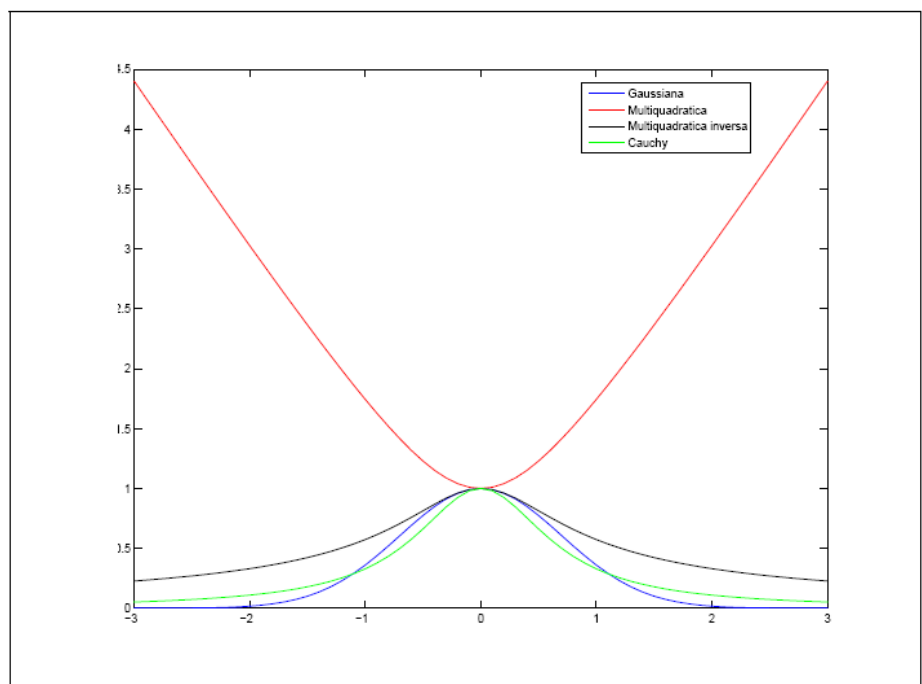
$$h(x) = \frac{\sqrt{(x-c_j)^2 + \sigma^2}}{\sigma^2}$$

Multiquadratica inversa:

$$h(x) = \frac{\sigma^2}{\sqrt{(x-c_j)^2 + \sigma^2}}$$

Cauchy:

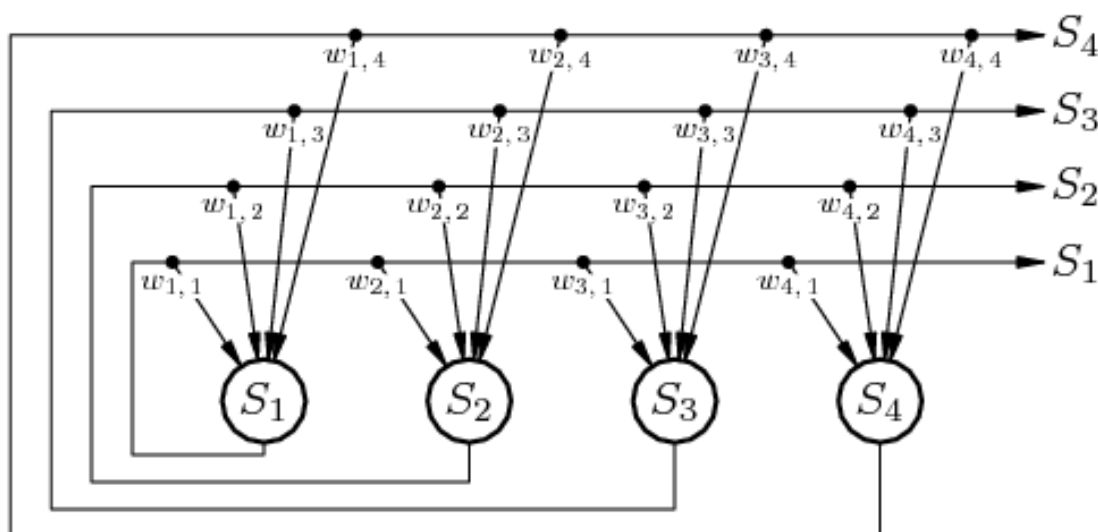
$$h(x) = \frac{\sigma^2}{(x-c_j)^2 + \sigma^2}$$



In ultima analisi è importante almeno citare le reti impulsive (*pulsed neural networks*), molto complesse in quanto basate su neuroni *integra-e-spara*, in grado di ricalcare più fedelmente il comportamento delle cellule cerebrali umane grazie alla loro codifica temporale delle informazioni.

2.9.5.1 Reti di Hopfield

Le Reti di Hopfield sono reti completamente connesse, in cui ogni neurone è connesso ad ogni altro neurone della rete. La funzione di attivazione di questa tipologia di rete è di tipo bipolare ed è regolata dall'insieme di soglie $\{\theta_i \mid i = 1, \dots, n\}$ e dall'insieme dei pesi sinaptici $\{w_{ij} \mid i, j = 1, \dots, n\}$. Il valore di uscita di un neurone è definito *stato del neurone*, che ampliato a tutta la rete individua lo *stato della rete*.



La rete può funzionare in tre modi:

- Parallelo
- Serie
- Misto

In parallelo lo stato di ogni neurone i al tempo t si modifica contemporaneamente a tutti gli altri ed è definito dalla funzione:

$$S_i(t) = \begin{cases} 1, & \sum w_{ij} S_j(t-1) - \theta_i \geq 0 \\ -1, & \sum w_{ij} S_j(t-1) - \theta_i < 0 \end{cases}$$

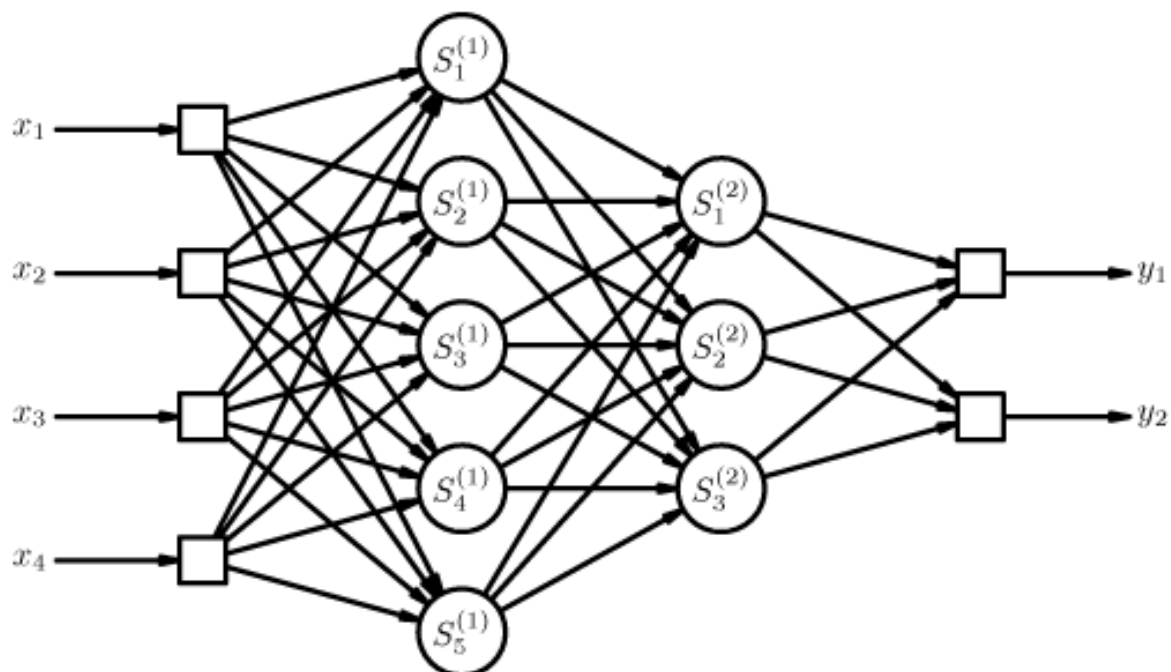
Seguendo un funzionamento di tipo seriale, un neurone per volta modifica il proprio stato, al

contrario secondo uno schema misto si modificano a gruppi. Una rete di Hopfield attraverso un'evoluzione ciclica riesce in ogni caso a raggiungere uno stato stabile, ma allo stesso modo a partire da una configurazione iniziale ad esso somigliante può raggiungerlo. La rete emula una *memoria associativa*, basata sulla memorizzazione di alcune informazioni, che possono essere richiamate dall'analisi di un loro sottoinsieme.

2.9.5.2 Reti feed-forward multistrato

L'informazione che attraversa questo modello di rete procede secondo un flusso unidirezionale, infatti i neuroni al suo interno non sono totalmente connessi, ma organizzati in *strati (layer)*. I dati attraversano gli strati interni consecutivi (*hidden layer*), che mettono in comunicazione uno strato di ingresso, pronto a ricevere input dall'esterno, con uno strato di uscita, che fornisce la risposta.

È molto utile sapere che le reti neurali sigmoidali multistrato con una struttura composta da almeno tre strati può essere adoperata come un approssimatore universale di funzioni continue, utile nell'eseguire studi di analisi matematica.



Il funzionamento di una rete multistrato si basa sull'utilizzo di differenti funzioni di attivazione per ogni differente strato, mentre il valore di uscita di ogni neurone i del k -esimo strato corrisponde al risultato della seguente espressione:

$$S_i^{(k)} = f^{(k)} \left(\sum_j w_{i,j}^{(k)} S_j^{(k-1)} - \theta_i^{(k)} \right)$$

dove lo stimolo per il livello di ingresso, $S_j^{(0)}$, corrisponde agli esempi forniti alla rete, e $f^{(k)}$ è la funzione di attivazione utilizzata per lo strato k -esimo.

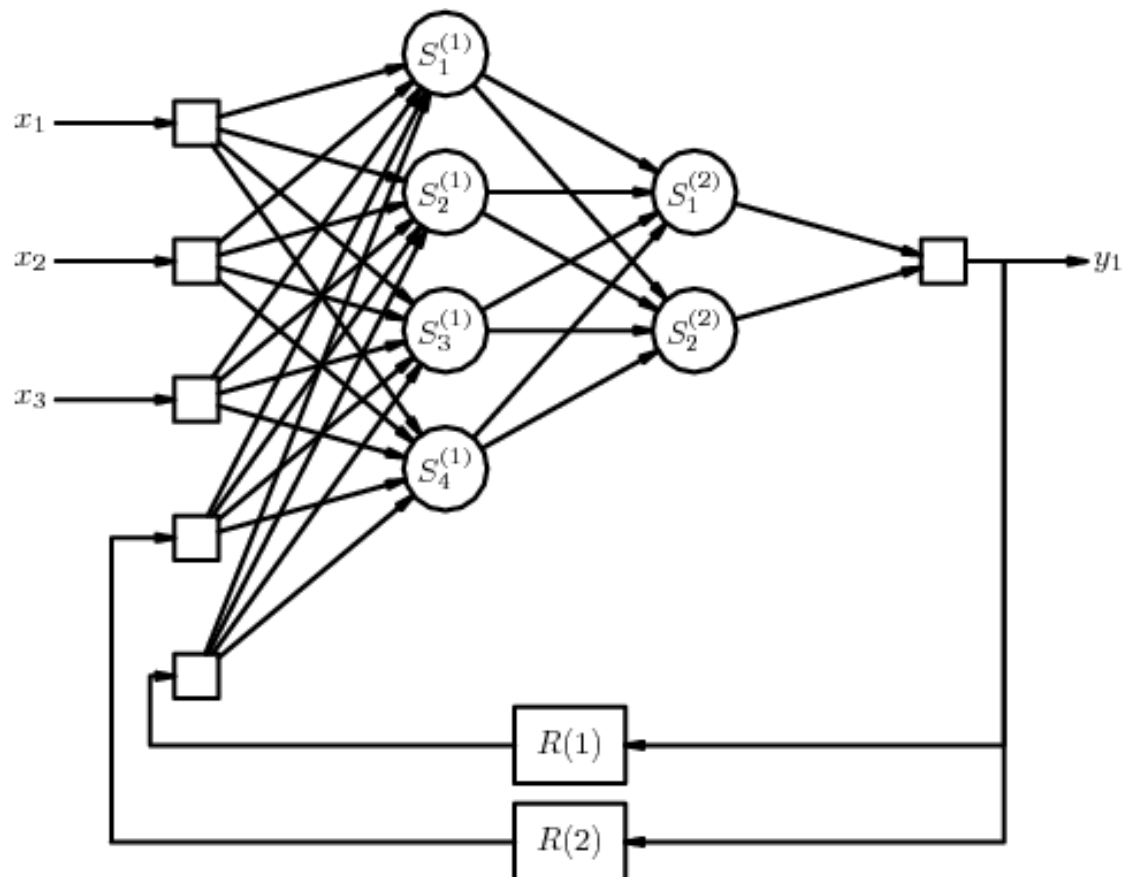
Esistono reti feed-forward basate sulla simmetria radiale (Radial Basis Function, RBF), in particolare su funzioni gaussiane e dotate nella maggior parte dei casi di un solo strato, sufficiente ad approssimare ogni funzione continua. La funzione G , propria di una rete RBF, risulta una combinazione lineare di funzioni radiali:

$$G(s) = \sum_i w_i S_i(c_i, \sigma_i)$$

dove c_i e σ_i sono i parametri caratteristici dell' i -esimo neurone. I parametri strutturali (centri, c , e fattori di scala, σ) caratterizzano la copertura dello spazio di ingresso da parte dei neuroni, cioè la regione dello spazio degli ingressi alla quale la rete risponde significativamente, regolando il valore di uscita in base a determinati pesi sinaptici.

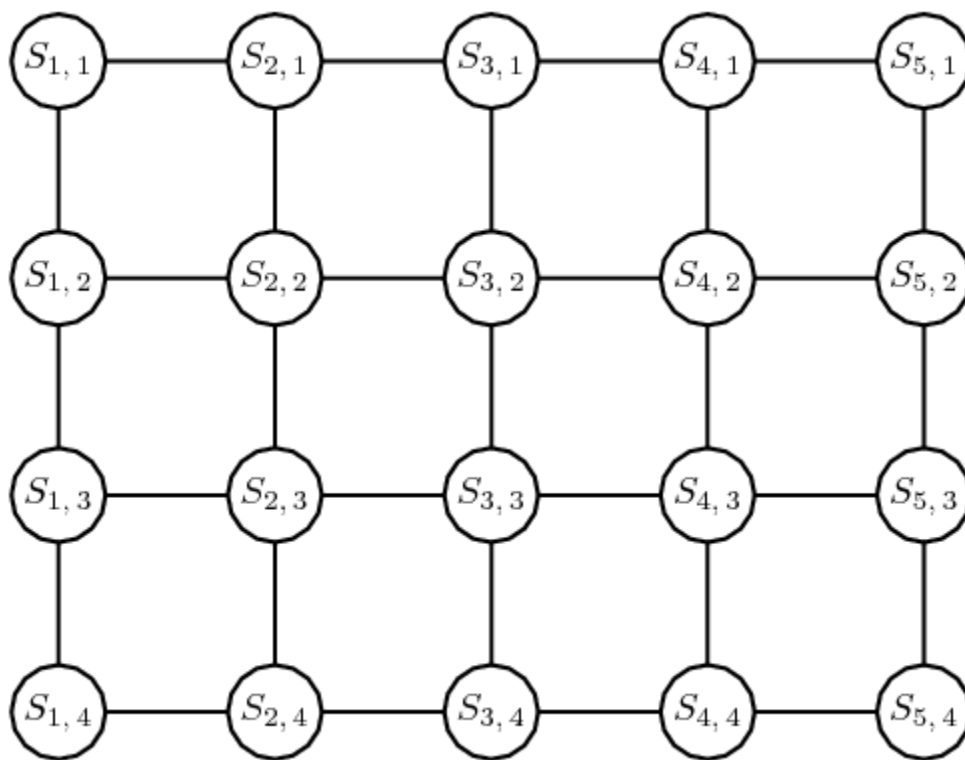
2.9.5.3 Reti ricorrenti

Le reti ricorrenti sono particolari reti multistrato connesse solo in maniera parziale, in modo da consentire la conservazione delle risposte di alcuni strati ed il loro futuro riutilizzo come valori in ingresso. Le connessioni basate su meccanismi retroattivi con differenti ritardi consentono alla rete di richiamare gli stati precedenti e le conferiscono un certo dinamismo, come in figura in cui $R(1)$ ed $R(2)$ corrispondono a due diversi ritardi nelle connessioni.



2.9.5.4 Reti competitive

Le reti competitive sono modelli che fundamentalmente differiscono dalle altre architetture neurali, infatti riescono ad auto-organizzarsi per individuare gli elementi che differenziano gli esempi di un insieme analizzato. I neuroni di tale rete entrano in competizione nel tentativo di rappresentare ognuno un sottoinsieme sempre più esteso di elementi in apprendimento. Gli esempi sono definiti nello *spazio delle caratteristiche* e le sue coordinate sono dette appunto *caratteristiche*, inoltre una speciale funzione riesce a determinare il grado di somiglianza o differenza tra gli elementi di una coppia.



La rete, una volta configurata, comincia a rispondere alle stimolazioni con l'attivazione del neurone i più vicino, la cui uscita S_i corrisponde alle qualità medie degli elementi dello spazio delle caratteristiche appartenenti al sottoinsieme del neurone. La rete raccoglie esempi molto simili in sottoinsiemi che avranno ciascuno al suo interno caratteristiche medie quasi identiche

2.9.5.5 Apprendimento

Una rete neurale assume un determinato comportamento in base all'impostazione dei suoi parametri, che avviene attraverso un algoritmo specifico di *apprendimento (learning)* o *addestramento (training)* in grado di analizzare e generalizzare il comportamento di un insieme di esempi proposti. La tipologia di algoritmo adottato dipende dal modello del neurone e dallo schema strutturale della rete, ma sostanzialmente è possibile individuare due differenti categorie di apprendimento: quello *supervisionato* e quello *non supervisionato*.

- Nell'apprendimento supervisionato vengono forniti alla rete neurale coppie esemplari formate da un valore in ingresso e dal corrispondente valore in uscita. È una modalità molto utile nella configurazione delle reti addette all'approssimazione di funzioni.
- Nell'apprendimento non supervisionato sono considerati invece solo i valori relativi agli ingressi, in modo che la rete riesca ad organizzarsi in maniera autonoma, come avviene per le memorie associative.

2.9.5.5.1 Apprendimento supervisionato

Tale tipologia di apprendimento presenta un elevato numero di algoritmi dedicati, tra cui quelli particolarmente significativi sono i seguenti:

- Algoritmo di retropropagazione (backpropagation)
- Apprendimento ibrido (hybrid learning)
- Apprendimento per rinforzo (reinforcement learning)

L'algoritmo di *retropropagazione* è il più utilizzato nella realizzazione delle reti neurali. Il suo principio di funzionamento si basa sulla verifica dell'errore commesso secondo i parametri impostati e nella sua successiva correzione in base ad un determinato gradiente ottenuto derivando la funzione di attivazione rispetto ai parametri da configurare. Passo dopo passo, grazie ad un elevato numero di esempi, la rete riesce a regolare correttamente i suoi valori fino a completare il suo apprendimento.

La configurazione di una rete feed-forward multistrato a due strati nascosti, utile all'approssimazione di una funzione reale, necessita un insieme di esempi costituito da coppie di numeri reali $\{(x_i, y_i)\}$. Il neurone dello strato di ingresso tramite la sua funzione lineare di

attivazione $S_1^{(0)} = x$ fornisce al resto della rete il valore iniziale. Il primo strato genera un output regolato dalla funzione:

$$S_j^{(1)} = f^{(1)} \left(\sum_k w_k^{(1)} x - \theta_j^{(1)} \right)$$

Il secondo strato a sua volta produce il seguente valore in uscita:

$$S_i^{(2)} = f^{(2)} \left(\sum_j w_{i,j}^{(2)} S_j^{(1)} - \theta_i^{(2)} \right)$$

L'uscita della rete neurale corrisponde all'output dello strato finale, ovvero:

$$\tilde{y} = \sum_i w_i^{(3)} S_i^{(2)}$$

La coppia di esempio (x, y) genera in ultimo un errore di approssimazione E :

$$E = (y - \tilde{y})^2$$

Il quadrato elimina il segno dalla differenza ottenuta e la rende facilmente derivabile.

L'algoritmo di backpropagation, utilizzando la regola di derivazione parziale delle funzioni composte in più variabili, consente di aggiornare i pesi sinaptici con le seguenti regole:

$$\begin{aligned} \Delta w_i^{(3)} &= -\eta \frac{\partial E}{\partial w_i^{(3)}} = -\eta \frac{\partial E}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial w_i^{(3)}} \\ \Delta w_{i,j}^{(2)} &= -\eta \frac{\partial E}{\partial w_{i,j}^{(2)}} = -\eta \frac{\partial E}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial S_i^{(2)}} \frac{\partial S_i^{(2)}}{\partial w_{i,j}^{(2)}} \\ \Delta w_j^{(1)} &= -\eta \frac{\partial E}{\partial w_j^{(1)}} = -\eta \frac{\partial E}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial S_i^{(2)}} \frac{\partial S_i^{(2)}}{\partial S_j^{(1)}} \frac{\partial S_j^{(1)}}{\partial w_j^{(1)}} \end{aligned}$$

Il valore η costituisce il *fattore di adattamento* (o *tasso di apprendimento*), addetto al controllo della velocità con cui si cerca di discendere verso il minimo dell'errore. Analoghe formule possono essere derivate per gli altri parametri della rete, come per le soglie.

L'apparente semplicità ed efficacia dell'algoritmo di backpropagation in realtà nasconde diversi problemi non lievi e che non presentano una facile risoluzione:

- I minimi locali delle funzione errore causano un aumento considerevole dell'errore, che η non riesce a correggere adeguatamente
- Non esiste certezza sul numero di iterazioni necessarie per giungere al minimo dell'errore

Nel tentativo di ovviare a queste complessità i ricercatori hanno messo a punto tecniche basate sul *simulated annealing* e sull'uso dei *momenti*. La prima consiste nell'aggiunta di un valore casuale alla funzione errore all'inizio dell'apprendimento si riesce a privarla dei minimi locali per individuare direttamente quello globale. La tecnica dei momenti invece comporta l'aggiunta di un fattore moltiplicativo al tasso di apprendimento, in modo da regolarlo adeguatamente in caso di crescita o riduzione dell'errore.

L'*apprendimento ibrido* è sicuramente una tecnica più efficiente per l'addestramento delle reti neurali, infatti consentono di adottare due metodologie differenti per agire sui parametri strutturali o sui pesi sinaptici. I primi sono definiti tramite un algoritmo che con meccanismi di ricerca identifica le regioni coperte dagli esempi di apprendimento. I pesi sinaptici $\{w_i\}$ invece possono essere calcolati con minor sforzo attraverso un sistema lineare.

Una rete dotata di m neuroni, sfruttando un insieme di apprendimento formato da M coppie $\{(x_k, y_k) | k=1, \dots, M\}$ può adoperare un sistema lineare di M equazioni in n incognite. Per ogni coppia (x_k, y_k) dovrebbe essere valida la seguente relazione:

$$y_k = \sum_{i=1}^n w_i g_i(x_k)$$

dove $g_i(x_k)$ è la risposta dell' i -esimo neurone all'ingresso x_k . A partire dalle seguenti considerazioni è possibile ricavare il seguente sistema:

$$\begin{cases} y_1 = w_1 g_1(x_1) + w_2 g_2(x_1) + \dots + w_n g_n(x_1) \\ \vdots \\ y_M = w_1 g_1(x_M) + w_2 g_2(x_M) + \dots + w_n g_n(x_M) \end{cases}$$

in cui le incognite sono i pesi sinaptici, $\{w_i\}$. Poiché il numero di esempi, M , è usualmente

maggiore del numero di neuroni, n , il sistema risulta sovradimensionato.

L'ultima tipologia di apprendimento è quella per *rinforzo* che si basa sulla modifica dei parametri a seconda dell'adeguatezza della risposta fornita dalla rete, ricreando un addestramento sullo stile premio-punizione. Gli altri metodi seguono la tecnica tipica di un insegnante, mentre quest'ultimo è molto più vicino alla figura di un critico, che giudica le risposte.

2.9.5.2 Apprendimento non supervisionato

L'apprendimento non supervisionato consente di istruire le reti che devono individuare le regolarità all'interno di un insieme formato da esempi molto simili tra loro. Presentando un esempio alla volta al sistema, questo sposta il neurone corrispondente nella direzione dell'esempio, anziché farglielo rappresentare. Allo stesso modo vengono spostati i neuroni topologicamente vicini per realizzare una mappatura graduale e rafforzare la procedura di addestramento.

Le reti di Hopfield sono generate con un addestramento non supervisionato, in base alla *regola di Hebb* secondo la quale se due neuroni si attivano a partire dagli stessi stimoli allora la connessione si rafforza.

La regola di adattamento che gestisce questa procedura è la seguente:

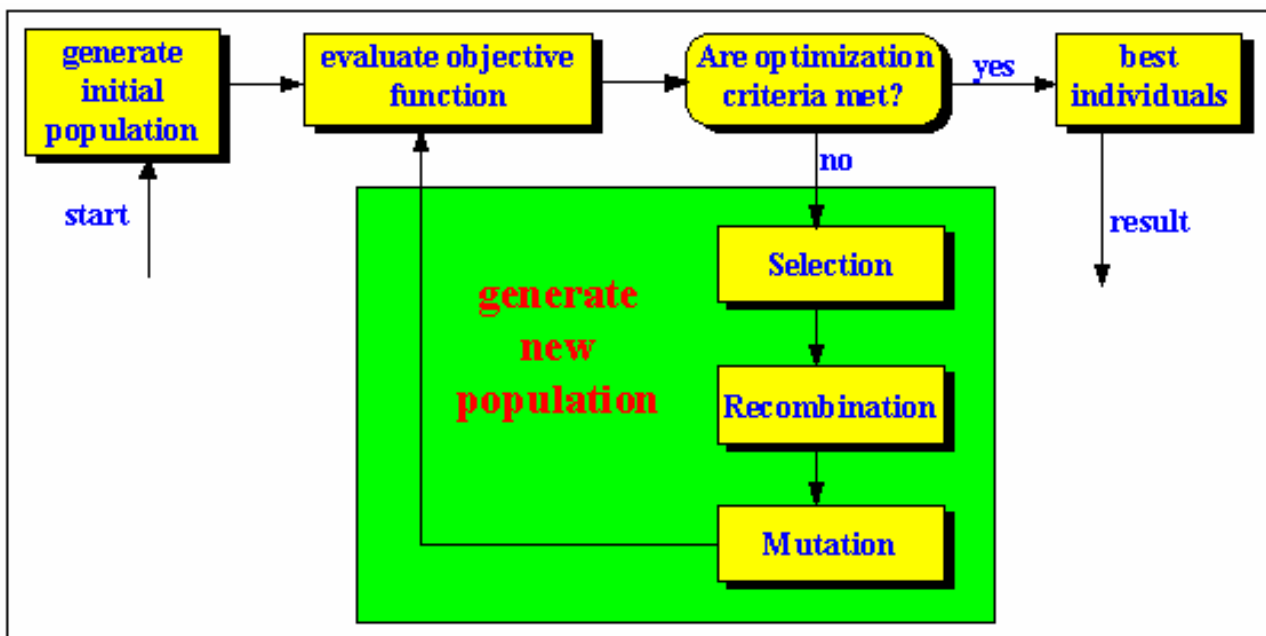
$$\Delta w_{ij} = S_i S_j$$

dove il peso della connessione riguardante l' i -esimo e il j -esimo neurone viene incrementata solo se entrambi i neuroni sono attivati dalla presentazione di un esempio da memorizzare.

2.10 Algoritmi genetici

2.10.1 Introduzione

Gli algoritmi genetici (GA) sono stati messi a punto per la prima volta da Holland, il quale intendeva simulare i processi genetici su cui si basano gli organismi biologici, in particolare quelli essenziali per i meccanismi evolutivi. Strutture di questo tipo, se applicate a sistemi informatici risolutori di problemi relativi al mondo reale, sono in grado di evolvere soluzioni adeguate, allo stesso modo di come in natura la lotta per la sopravvivenza lascia in vita solo gli individui più adatti ed in grado di riprodursi più prolificamente. Il loro patrimonio genetico è trasmesso ad un numero elevato di successori (*offspring*), creando grazie ad una serie di incroci generazioni molto adatte all'ambiente (*superfit*) e quindi con un grado evolutivo più alto rispetto agli antenati. Gli algoritmi genetici analogamente assegnano ad ogni soluzione un punteggio di adattamento (*fitness score*), in base alla sua qualità, e quindi una certa probabilità di resistere alla competizione con le altre. Se il GA è stato configurato in maniera corretta allora è facile giungere ad un'ottima soluzione del problema, come il disegno della struttura di una rete neurale e molte altre applicazioni, in cui non esistono tecniche specializzate.



Schema 8. Processo di un algoritmo genetico

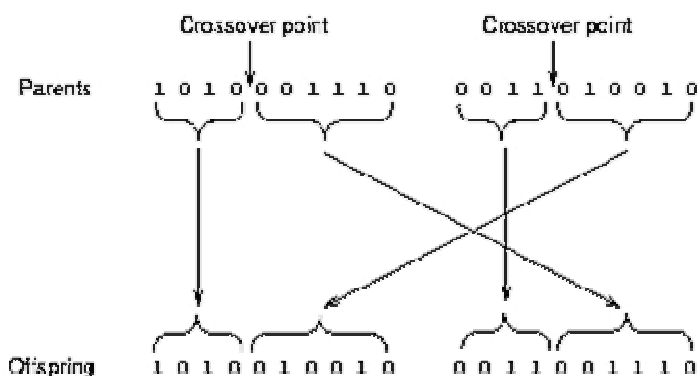
2.10.2 Principi basilari

Per consentire un adeguato funzionamento ad un algoritmo genetico deve innanzitutto essere attuata una codifica (*representation*) del problema ed è necessaria la cosiddetta *funzione fitness* che assegni i valori alle soluzioni. Il procedimento si svolge in una scelta dei genitori per la riproduzione e nella loro ricombinazione per ottenere figli.

Una soluzione è codificata da una serie di parametri (*geni*), che riuniti in una stringa di valori formano un *cromosoma* $f(x, y, z, \dots)$, in cui ogni gene è rappresentato da un numero binario. L'insieme dei parametri appartenenti alla totalità dei cromosomi costituisce il *genotipo*, mentre tutte le informazioni richieste sono riunite nel *fenotipo*. Il fitness score di una soluzione viene ricavato dal fenotipo e quindi dai cromosomi attraverso la funzione fitness.

La riproduzione è una fase cruciale nel funzionamento di un algoritmo genetico, infatti genera una discendenza da genitori selezionati tra i migliori attraverso un particolare schema. La procedura non è così banale, in quanto i cromosomi dei due individui di partenza vengono ricombinati tramite i meccanismi del *crossing-over* e della *mutazione*.

Il crossing-over taglia le stringhe di due cromosomi in posizioni casuali in modo da produrre due segmenti "testa" (*head*) e due segmenti "coda" (*tail*). Questi ultimi sono poi scambiati per produrre due nuovi cromosomi contenenti entrambi una parte di ognuno dei due genitori. Tale modalità è anche definita single point crossover e può essere applicata in base ad una probabilità compresa tra 0,6 ed 1,0. Senza il crossing-over i figli sono semplici duplicazioni dei genitori.



La mutazione è il passo successivo al crossing-over ed altera a caso ogni gene con una probabilità dello 0,001. Un esempio può essere rappresentato da una funzione fitness codificata a 10 bit e di tipo esponenziale con un massimo in $x=0,2$. L'incrocio avviene sul secondo bit e ricombina i geni fino ad ottenere figli con fitness maggiore, sebbene sia possibile anche il caso contrario.

Individual	x	Fitness	Chromosome
Parent 1	0.08	0.05	00 01010010
Parent 2	0.73	0.000002	10 11101011
Offspring 1	0.23	0.47	00 11101011
Offspring 2	0.58	0.000007	10 01010010

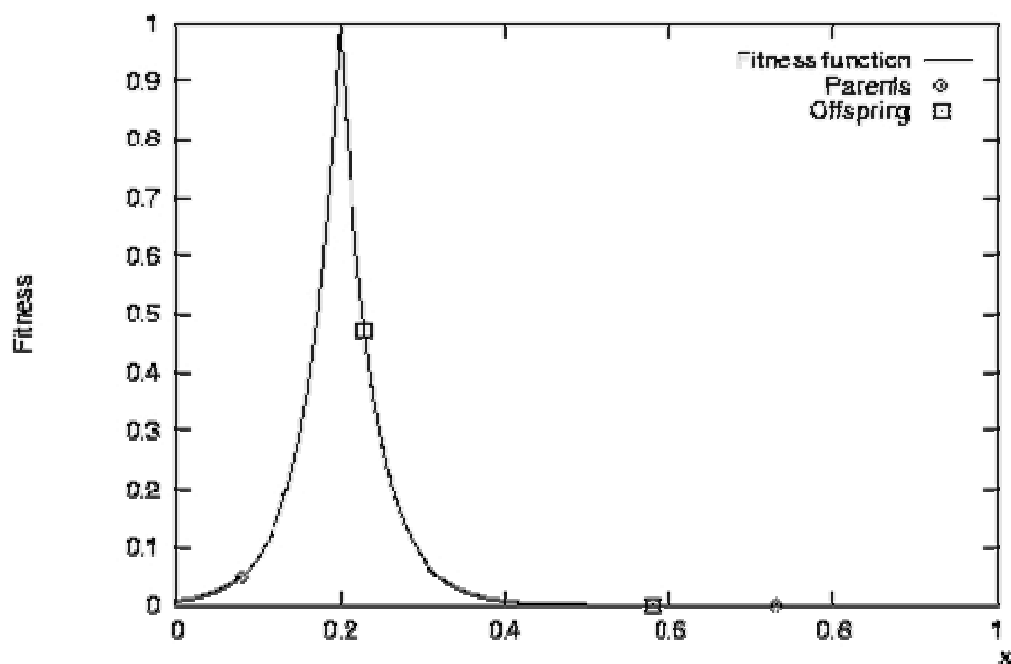


Grafico 1. Funzione Fitness

Se l'algoritmo genetico è configurato in maniera corretta allora sarà possibile ottenere un certo numero di generazioni successive con una progressiva crescita del fitness fino ad una graduale uniformità (*convergenza*). Un gene converge quando il 95% della popolazione condivide il medesimo valore.

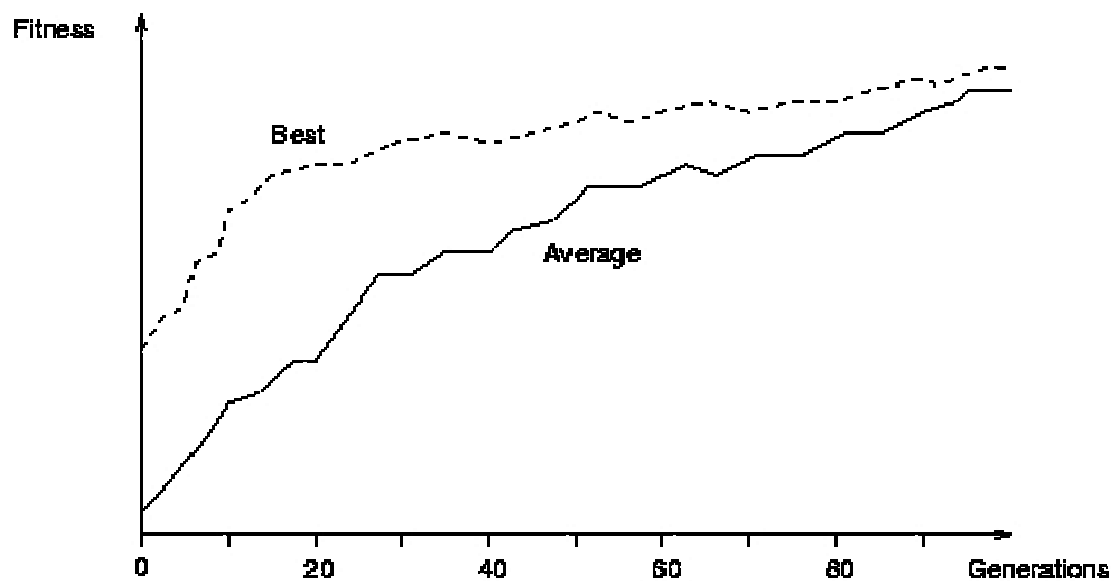


Grafico 2. Convergenza

3. Studio del PICAXE Micro-Robot Line Follower

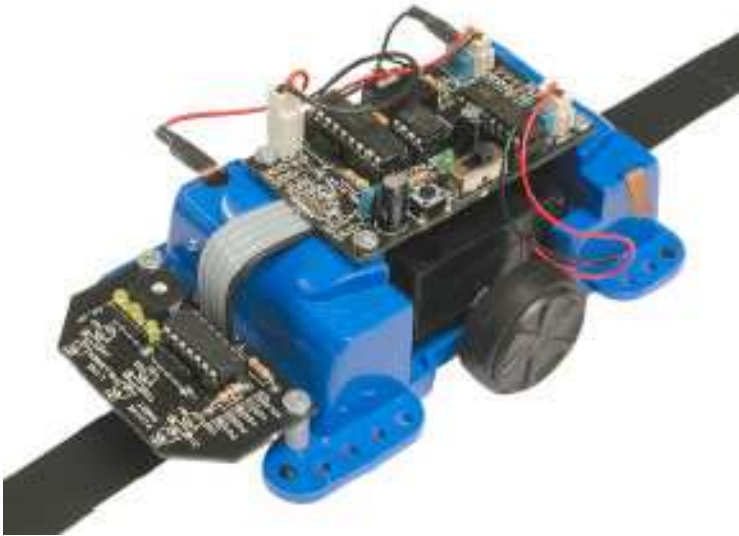
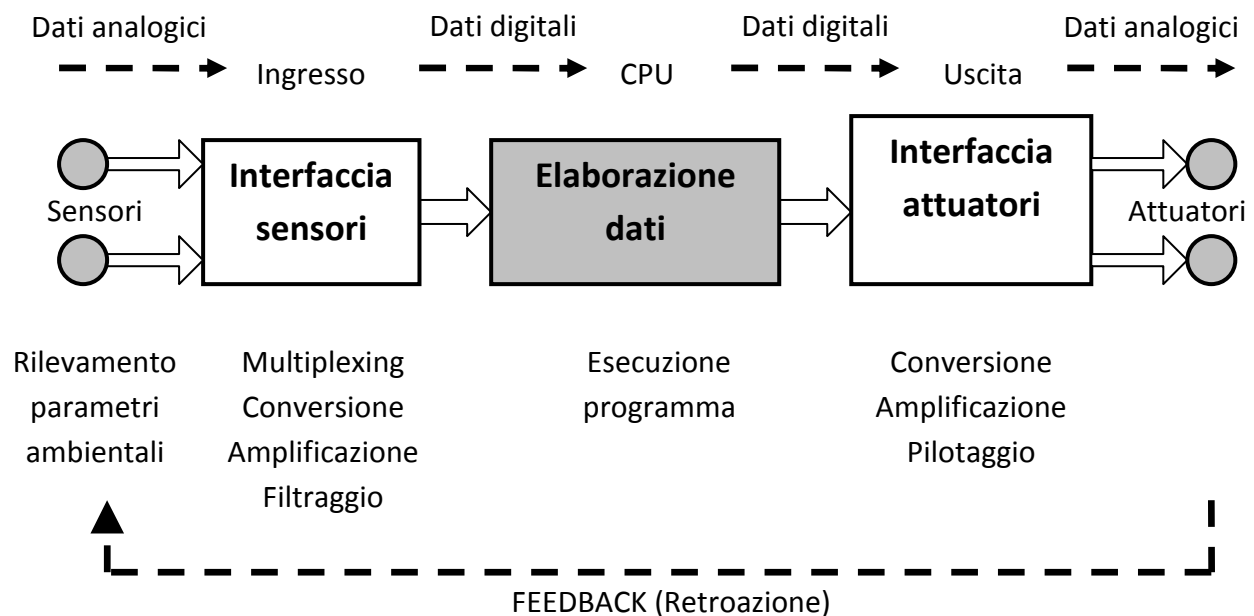


Figura 2. Micro-Robot PICAXE dotato di Line Follower module

Scheda tecnica del Micro-Robot:

- Casa produttrice: Solarbotics
- Base in plastica – 120 X 80 mm
- 2 ruote in gomma
- 2 motori (L293D) da 0,46 W
- Alimentazione con 4 Batterie AA
- Microcontrollore PICAXE 18
- Connettore 3.5 stereo per la programmazione
- Upgrade AXE121 Line Follower module



3.1 Struttura di base

Il Micro-Robot PICAXE è un robot dotato di ruote e può essere considerato un Wheeled Mobile Robot (WMR),

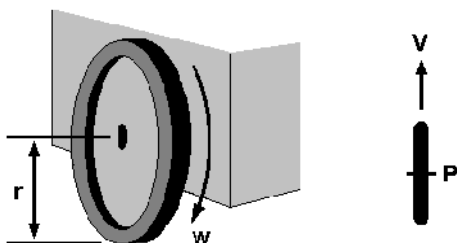
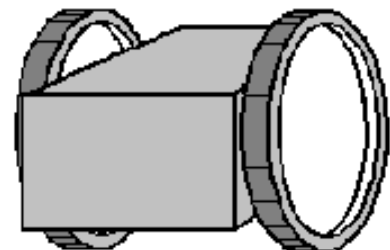
*“A robot capable of locomotion on a surface **solely through the actuation of wheel assemblies** mounted on the robot and in contact with the surface. A wheel assembly is a device which provides or allows motion between its mount and surface on which it is intended to have **a single point of rolling contact.**”* (Muir and Newman, 1986)

ovvero

“Un robot capace di locomozione su una superficie esclusivamente attraverso il movimento delle ruote montate sul robot ed a contatto con la superficie. Le ruote sono dispositivi che forniscono o permettono il movimento tra il loro sostegno e la superficie sulla quale esiste un singolo punto di contatto.”

La struttura basata su un corpo centrale e su due ruote laterali presenta le seguenti caratteristiche:

- Estrema facilità nel modellarlo
- Sensibilità alle irregolarità del terreno
- Rischio di scivolamento



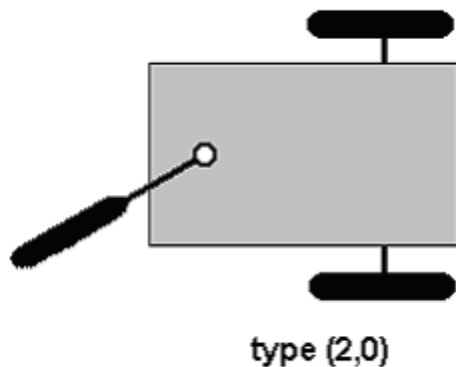
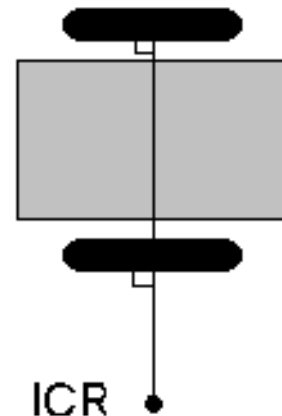
La presenza di ruote fisse impone una precisa restrizione sulla mobilità:

- Impossibilità di muoversi nella direzione perpendicolare al piano della ruota

Il punto di intersezione degli assi delle ruote, definito anche come Centro di Istantanea Rotazione (CIR) o Instantaneous Center of Rotation (ICR), corrisponde a tutti i punti della retta passante per i centri delle ruote.

I gradi di mobilità del Micro-Robot sono due, seguendo un arco variabile di movimento in base alla linea dei CIR.

Al contrario, i gradi di sterzabilità, ovvero il numero di ruote centrate orientabili che si muovono indipendentemente per sterzare il robot, sono pari a zero.



In base alle seguenti caratteristiche:

- Gradi di libertà: 2
- Gradi di sterzabilità: 0

Il Micro-Robot PICAXE può essere classificato come un **Type (2,0)**

3.2 Elementi di cinematica

Per analizzare in maniera efficace il moto del Micro-Robot sono strettamente necessarie alcune nozioni di cinematica, in particolare:

- **Cinematica diretta**→fornisce le coordinate (x, y) e l'angolo θ raggiunti dal robot, dati i parametri di controllo (velocità delle ruote e tempo)
- **Cinematica inversa**→consente di ricavare i parametri di controllo per condurre in un determinato istante il robot nella posizione identificata dalle coordinate (x, y) e con un angolo θ

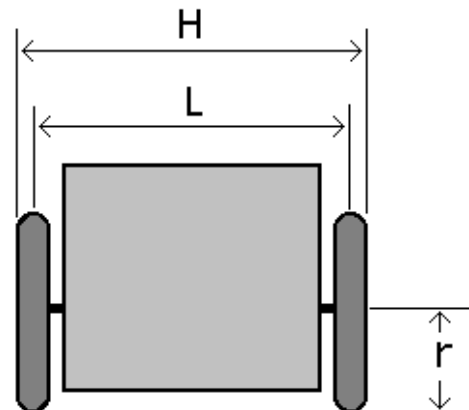
Postura P del Micro-Robot:

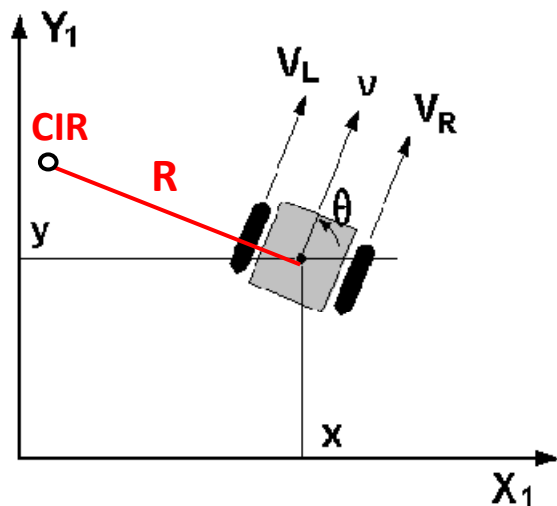
- (x, y) Posizione
- θ Orientamento

$$P = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

I parametri di controllo, ovvero le variabili controllate dall'operatore sono in questo caso:

- V_R velocità ruota destra
- V_L velocità ruota sinistra





Le due ruote realizzate in modo da avere l'asse comune si muovono attorno al CIR con una velocità angolare pari a ω e secondo un raggio di curvatura R

$L \rightarrow$ Larghezza Micro-Robot

$$\omega(R + \frac{L}{2}) = V_R$$

$$\omega(R - \frac{L}{2}) = V_L$$

$$CIR(x - R \sin \theta, y + R \cos \theta)$$

$$\omega = \frac{V_R - V_L}{L}$$

$$V = \frac{V_R + V_L}{2}$$

È possibile ottenere il valore di R uguagliando diverse espressioni del valore di $\omega(t)$

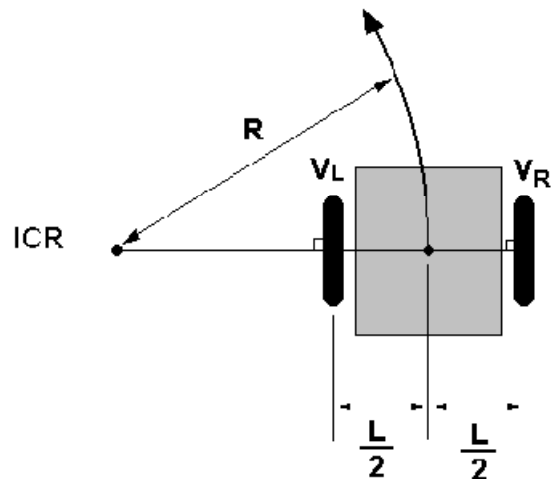
$$\frac{V_R - V_L}{L} = \frac{V_R}{R + \frac{L}{2}}$$

- Movimento dritto

$$R \rightarrow \text{infinito} \quad V_R = V_L$$

- Rotazione su sé stesso

$$R = 0 \quad V_R = -V_L$$



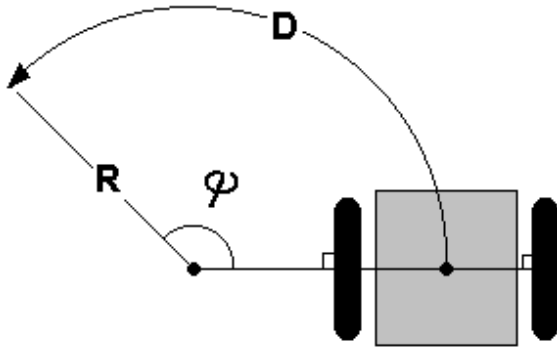
Cinematica diretta

A $t+dt$, rotazione attorno al CIR a distanza

R con angolo φ

$$\begin{aligned}(x', y', \varphi')^T &= \\ &= Rot(z, \omega dt) Trasl(x - CIR_x, y - CIR_y, \varphi) + \\ &+ Trasl(CIR_x, CIR_y, \omega T)\end{aligned}$$

Le soluzioni di tali equazioni al tempo t si ottengono attraverso la risoluzione dei seguenti integrali:



$$x(t) = \int_0^t v(t) \cos(\varphi(t)) dt$$

$$y(t) = \int_0^t v(t) \sin(\varphi(t)) dt$$

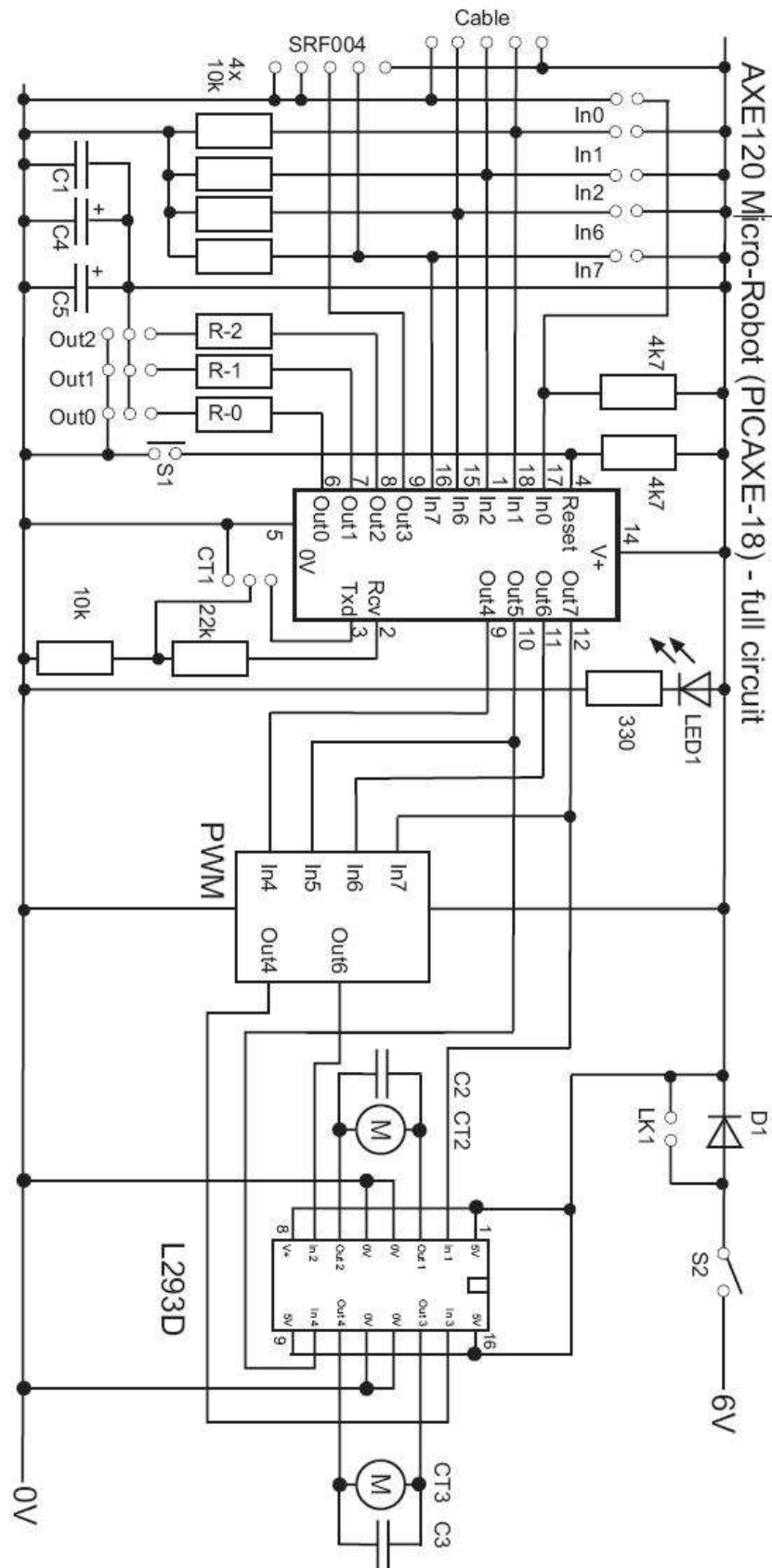
$$\varphi(t) = \int_0^t \omega(t) dt$$

Cinematica inversa

Il problema è risolto da infiniti valori di V_L e V_R che conducono il robot al punto di destinazione, quindi è possibile trovare la soluzione più adeguata tramite i casi particolari:

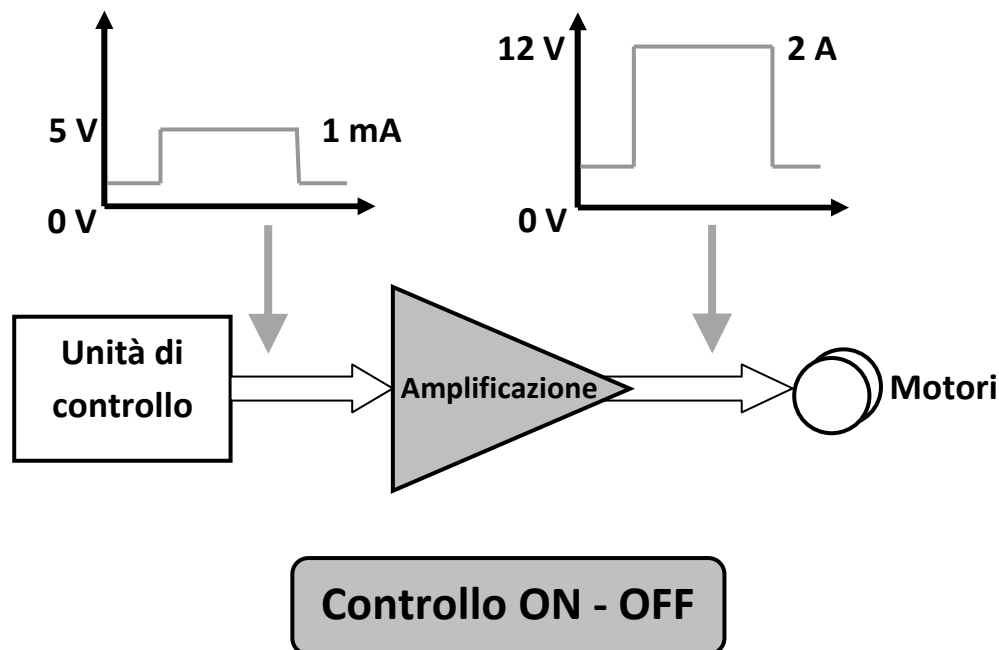
Se la destinazione è la postura (x, y, q) , allora è possibile far avanzare il robot fino alla posizione (x, y) impostando $V_L = V_R$ per poi farlo ruotare di un angolo q con $V_L = -V_R$.

3.3 Diagramma del circuito



3.4 Funzionamento dei motori elettrici

Il sistema di funzionamento dei motori del Micro-Robot PICAXE è molto semplice, infatti si basa su un'alimentazione a corrente continua (DC). La velocità è controllata attraverso una tecnica chiamata **“Modulazione di larghezza di impulso” (PWM - Pulse Width Modulation)**, che trasmette l'output ai motori sottoforma di valori ON e OFF ad alta frequenza. È possibile variare facilmente la velocità tra 50 (lento) e 255 (veloce), impostando particolari comandi a livello software che agiscono sul chip a 8 pin installato sulla scheda principale e riservato alla modalità PWM.



In figura è rappresentato il più comune esemplare di sistema di guida elettrica. La sua struttura è basata su un motore (1), una frizione (2) ed una massa guidata (I).

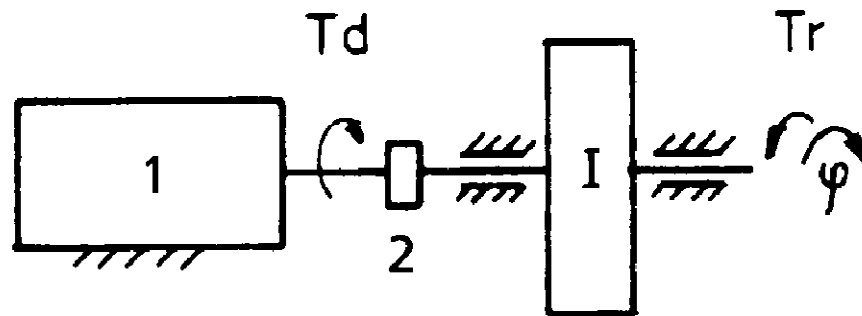


Figura 11. Schema di un Motore DC

Il moto di questa massa può essere descritto attraverso questa equazione:

$$\ddot{\phi}I = T_d - T_r$$

$\ddot{\phi}$ non è altro che l'accelerazione del robot, T_d corrisponde alla coppia di forze che attua il moto e T_r è la coppia resistente. Nel caso che tali valori si mantengano costanti nel tempo è facile per un operatore fare delle stime pressoché esatte sulle velocità raggiunte dal robot, infatti:

$$\ddot{\phi} = \frac{T_d - T_r}{I} \qquad \dot{\phi} = \frac{T_d - T_r}{I} t$$

Se le condizioni iniziali sono $t = 0$ e $\dot{\phi} = 0$ allora, secondo le leggi della meccanica newtoniana, si avrà:

$$\phi = \frac{T_d - T_r}{2I} t^2$$

Da queste semplici formule è poi possibile ricavare i valori del tempo t :

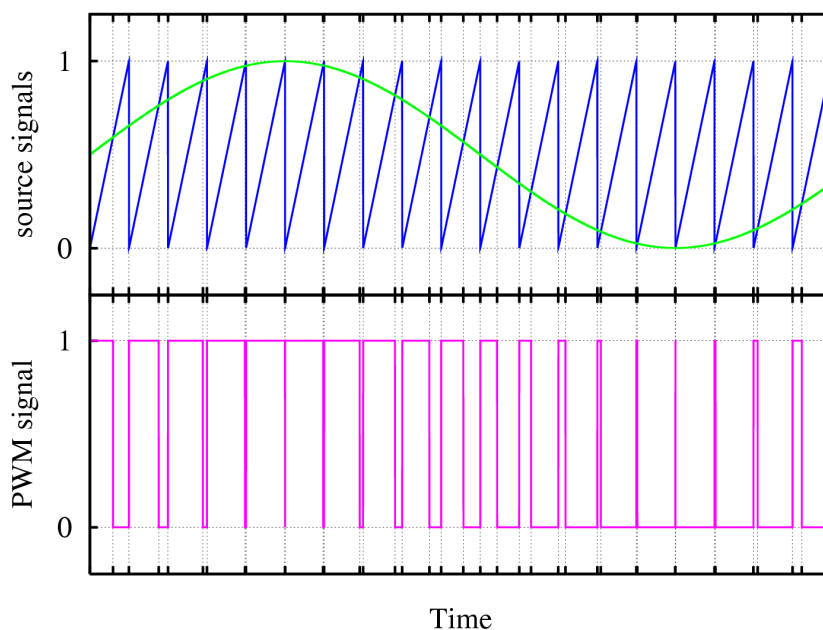
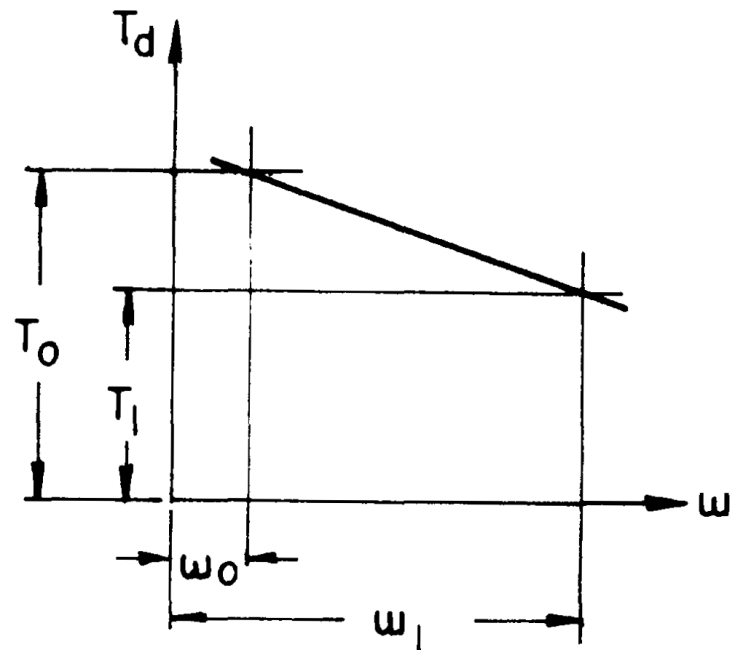
$$t = \phi \frac{I}{T_d - T_r} \qquad t = \sqrt{\frac{2\phi I}{T_d - T_r}}$$

La situazione reale è molto più complessa, infatti nessuno di questi parametri si mantiene costante nel tempo. In particolare la coppia motrice dipende fortemente dalla natura del motore, di conseguenza è molto variabile e può essere matematizzata solo attraverso un'equazione più complessa:

$$\omega = b_1 - b_2 T_d$$

$$T_d = a_1 - a_2 \omega$$

Le costanti a e b definiscono la posizione e l'inclinazione della retta nel piano e costituiscono valori tipici di un particolare motore.



La tecnica PWM consente l'invio di un impulso molto potente che dipende dal rapporto fra la durata degli impulsi ed il periodo di ripetizione:

$$p_0 = \frac{t}{T}$$

Figura 12. Conversione di un segnale analogico con PWM

3.5 Sensori



L'upgrade Line Follower module installato sul Micro-Robot PICAXE consente alla macchina di seguire il percorso definito da una linea nera riportata sulla superficie di appoggio attraverso i segnali provenienti dai tre fotodiodi installati sulla scheda aggiuntiva posta nella parte anteriore.

La tipologia di sensori installati sfrutta la variazione di intensità della luce, una caratteristica molto comune e facilmente rilevabile nei fenomeni luminosi. Il fascio di luce emesso dai diodi (LED) viene riflesso dalla superficie sottostante ed in seguito catturato dal fotodiodo corrispondente. Le difficoltà maggiormente riscontrabili non sono strettamente legate a questo fenomeno, infatti fattori come il tempo e la temperatura alterano le emissioni della sorgente complicando un'applicazione molto semplice ed efficace.

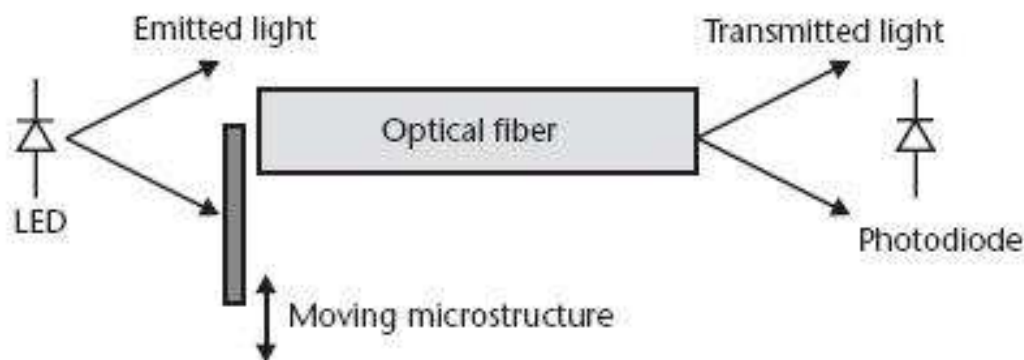
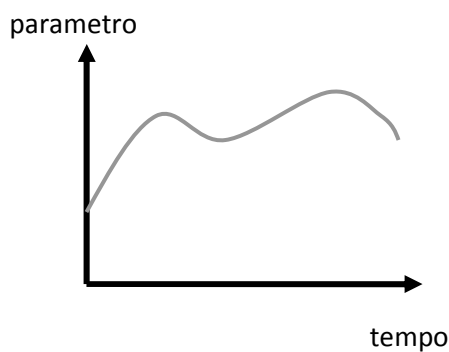
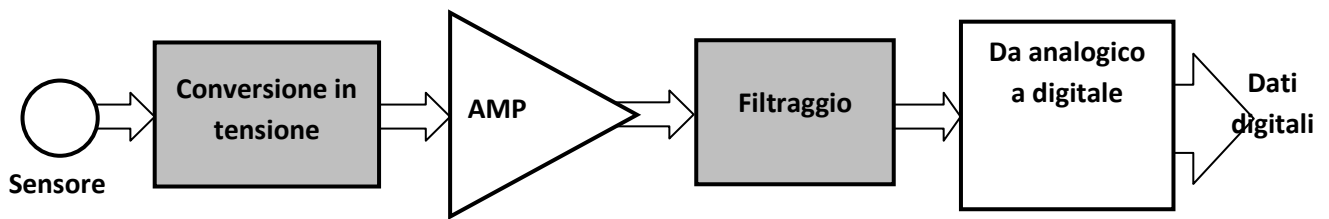
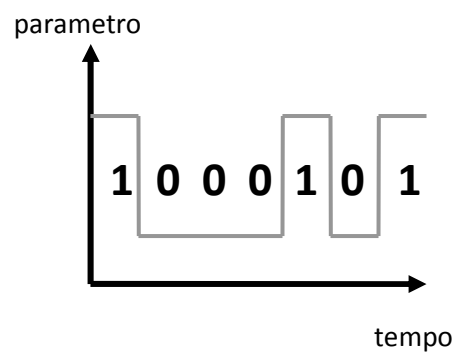


Figura 13. Principio di base dei sensori ottici



Segnali analogici

- Colorazione del supporto



Segnali digitali

- Segnali codificati in linguaggio binario

Il sistema sensoriale del Micro-Robot PICAXE segue il classico principio di trasformazione dei segnali analogici in segnali digitali attraverso operazioni di amplificazioni e filtraggio del flusso in entrata.

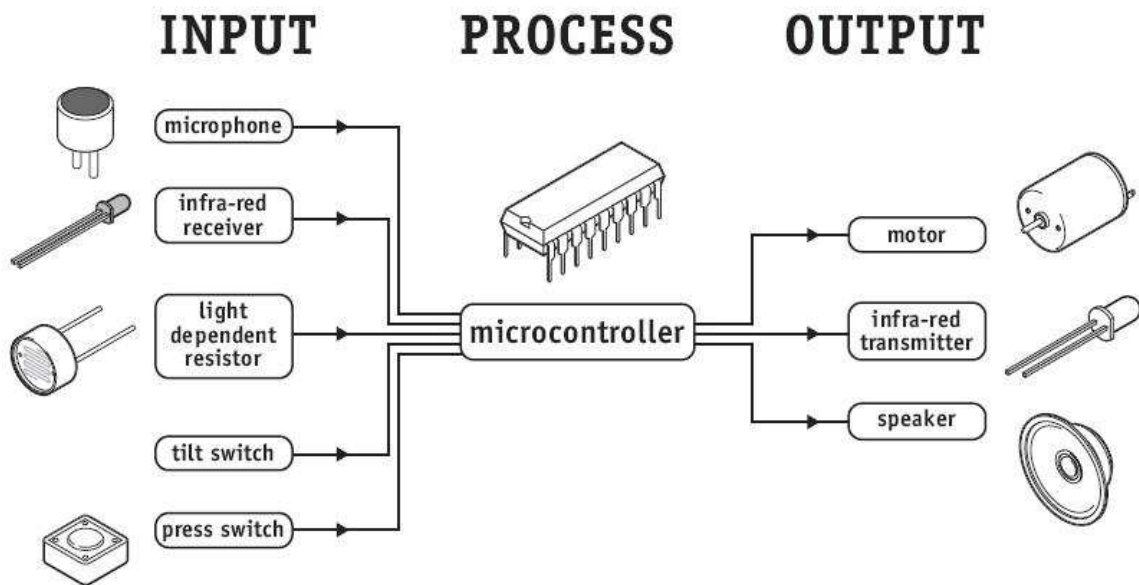
3.6 Microcontrollore PICAXE

Il sistema di elaborazione del robot si basa sulla tecnologia di ultima generazione PICAXE, a cui appartengono microcontrollori di tipo FLASH, a basso costo e a lunga durata. Possono essere programmati oltre 100 mila volte nel corso della loro vita. L'uso di un microcontrollore in primo luogo conduce ad una serie di notevoli vantaggi:

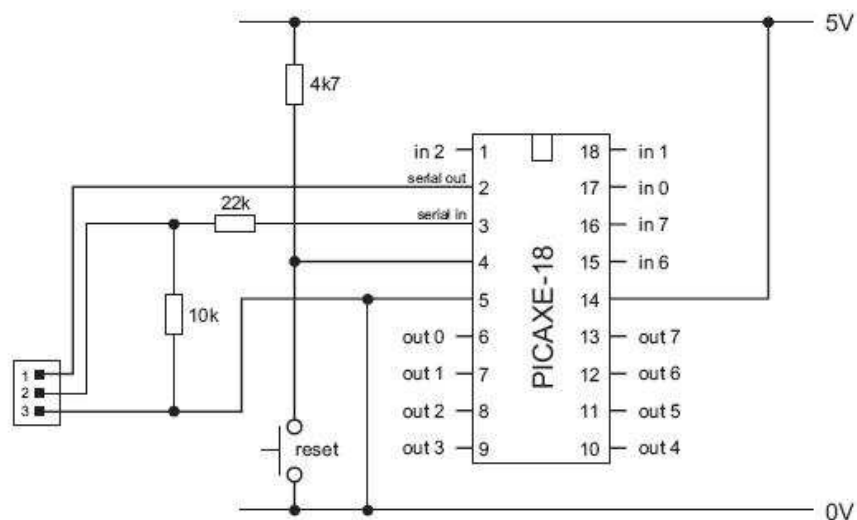
- Aumento dell'affidabilità con la riduzione delle dimensioni
- Sostituzione di un elevato numero di componenti con un unico
- Semplificazione del procedimento di assemblaggio

- Maggiore flessibilità e adattabilità del prodotto attraverso la programmazione
- Maggiore rapidità di cambiamento attraverso la sola sostituzione del software

Il sistema PICAXE viene spesso adoperato anche negli antifurti per abitazioni o in equipaggiamenti per medici, inoltre sfrutta un linguaggio di programmazione BASIC che lo rende adatto anche ad un'utenza meno esperta.



Il microcontrollore a 18 PIN di cui è dotato il Micro-Robot acquisisce gli input e, attraverso il programma memorizzato al suo interno, prende le adeguate decisioni e comanda un determinato comportamento alla macchina.



PICAXE-18

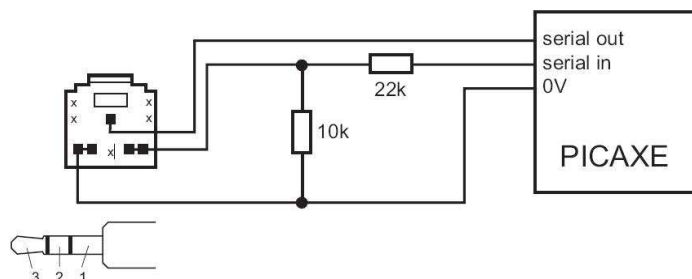
ADC 2 / Input 2	1	18	Input 1 / ADC 1
Serial Out	2	17	Input 0 / ADC 0
Serial In	3	16	Input 7
Reset	4	15	Input 6
0V	5	14	+V
Output 0	6	13	Output 7
Output 1	7	12	Output 6
Output 2	8	11	Output 5
Output 3	9	10	Output 4

La seguente tabella indica gli input disponibili sul microcontrollore a seconda dell'upgrade installato. Nel caso del Micro-Robot che sfrutta il line follower sono impiegati solo **In1**, **In2** ed **In6**.

Module	Out3	In0	In1	In2	In6	In7
line follower			X	X	X	
infrared		X				
ultrasonic	X					X

3.7 Programmazione del Micro-Robot

La programmazione della macchina si risolve in un procedimento estremamente intuitivo a cominciare dal collegamento con il PC attraverso la porta USB, che agisce direttamente sul microcontrollore.



L'ambiente di programmazione è estremamente intuitivo, infatti, oltre alla classica programmazione in linguaggio BASIC, consente di realizzare un software a partire da uno schema a blocchi (Flowchart).

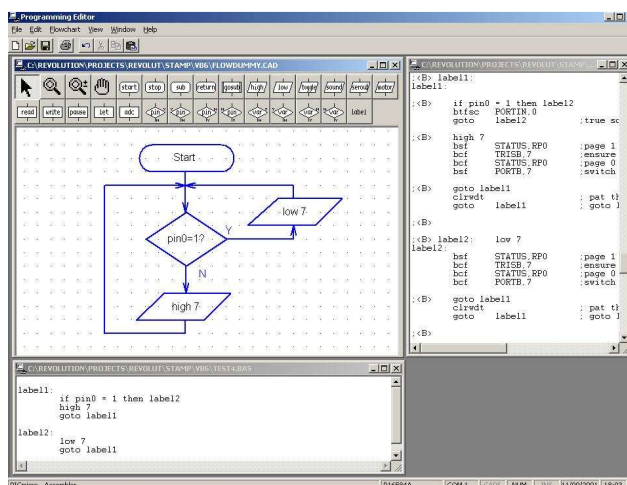
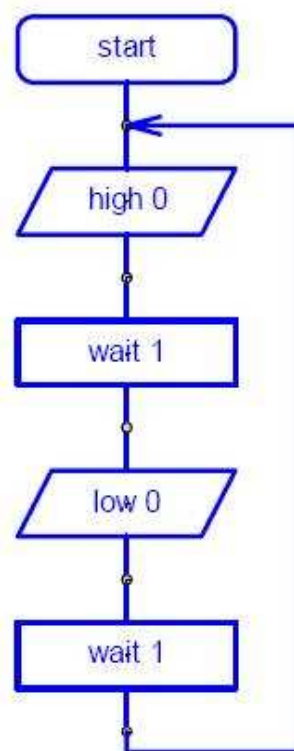


Figura 13. Ambiente di programmazione


```
main:
  high 0
  wait 1
  low 0
  wait 1
  goto main
```



Il codice ed il diagramma rappresentano lo stesso programma che esegue delle operazioni molto elementari nell'ordine dettato dalle istruzioni o direttamente dalle frecce nel grafico.

Main: *è semplicemente un'etichetta che introduce un campo*

high 0 *invia il segnale di accensione al LED collegato alla porta 0*

wait 1 *attende 1 secondo*

low 0 *invia il segnale di spegnimento al LED collegato alla porta 0*

wait 1 *attende un secondo*

goto main *ritorna all'etichetta riaprendo il ciclo*

Il programma sopra riportato è fornito come esempio nel Programming Editor e costituisce uno dei programmi più semplici realizzabili, tanto da mostrare quanto sia elementare creare codice operativo adatto al Micro-Robot PICAXE.

Il codice che consente il funzionamento ottimale dell'upgrade Line Follower è leggermente più complesso del precedente in quanto racchiude la struttura decisionale "if...then", comunissima in qualsiasi programma che possiede una minima autonomia.

Init: *'etichetta iniziale*

pause 100 *'avvio del controller con pausa di 100 ms*

main: *'etichetta principale*

if input1 is on then go_f *'se input1 è acceso si sposta sull'etichetta go_f*

if input2 is on then go_l *'se input2 è acceso si sposta sull'etichetta go_l*

if input6 is on then go_r *'se input6 è acceso si sposta sull'etichetta go_r*

goto go_s

go_f:

let pins = %10100000 *'va diritto*

goto main *'riavvia il ciclo*

go_l:

let pins = %00100000 *'va a sinistra*

goto main *'riavvia il ciclo*

go_r:

let pins = %10000000 *'va a destra*

goto main *'riavvia il ciclo*

go_s:

let pins = %00000000 *'si ferma*

goto main *'riavvia il ciclo*

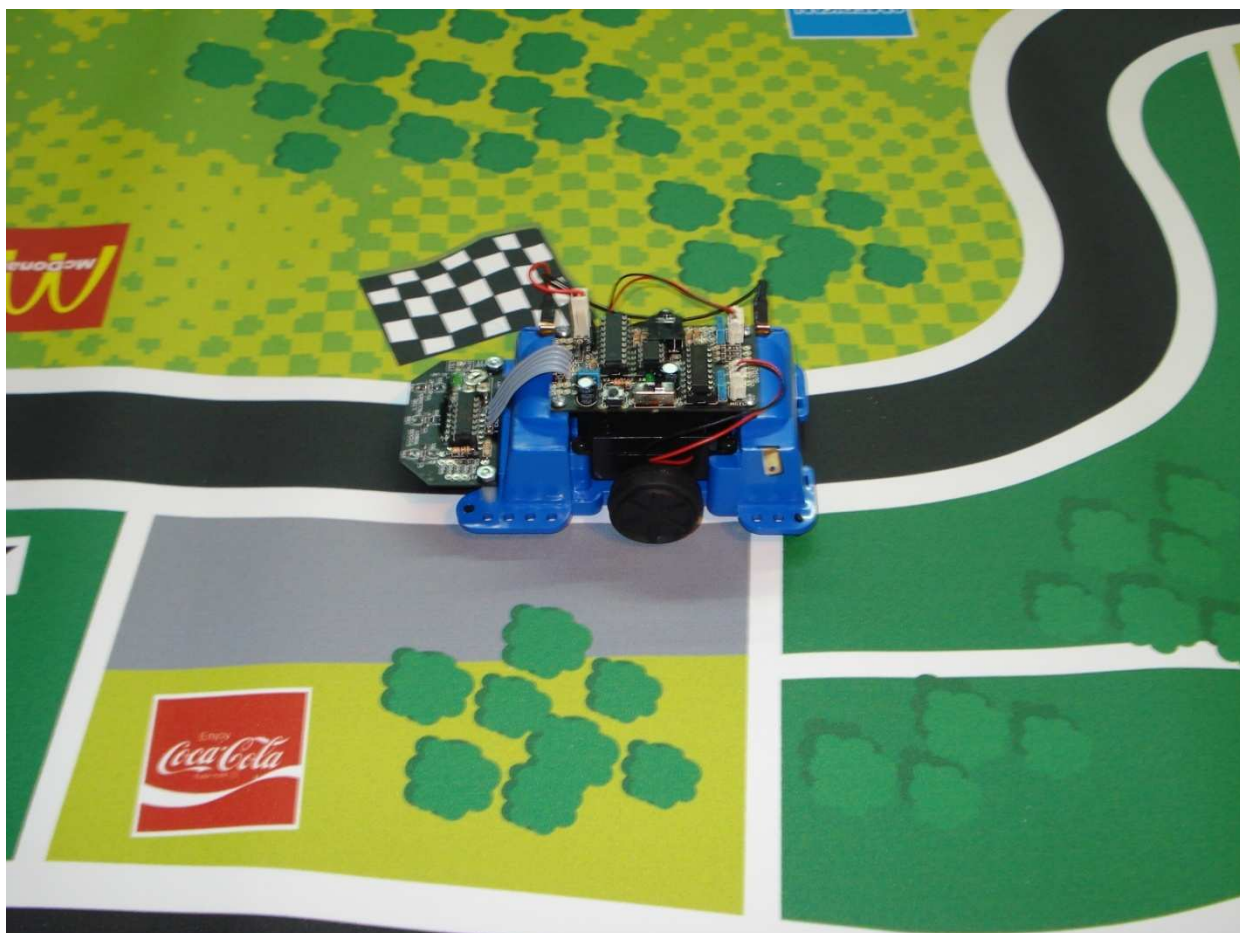


Figura 14. Micro-Robot PICAXE sul suo percorso

4. Il futuro della Robotica

Gli sviluppi straordinari compiuti dai ricercatori negli ultimi anni hanno consentito alla robotica di invadere la vita quotidiana. Ormai ogni settore ha iniziato a beneficiare di queste macchine intelligenti che riescono in maniera sempre più efficace a sostituire l'azione dell'uomo nei lavori più difficili e pericolosi. La robotica grazie alla sua interdisciplinarietà ha quindi trovato applicazione in molteplici contesti tra i quali comunque non esiste una netta differenza.

4.1 Robotica biomedicale

La robotica medica ha subito un impulso significativo negli ultimi anni ed ha comportato un aumento progressivo del numero di sale operatorie assistite, dotate di apparecchiature automatizzate all'avanguardia in campo clinico.

I settori in cui ha riscosso maggiore successo sono numerosi:

- Diagnostica
- Simulatori chirurgici
- Percezione e stimolazione tattile
- Teleoperazione bilaterale
- Addestramento aptico
- Assistenza ai disabili
- Tecnologia delle protesi



Figura 15. Sala operatoria dotata di tecnologie di ultima generazione

Comunque la robotica biomedica non si limita al settore chirurgico o riabilitativo, infatti attualmente è riposta grande fiducia in quegli strumenti che possono migliorare la qualità dell'assistenza sanitaria e la vita dei cittadini, in particolare di quelli più anziani e con difficoltà di accesso alle strutture. Nuove ricerche invece hanno aperto vasti orizzonti per la replicazione dei sistemi biologici.

4.2 Robotica marina

Il mare attualmente ricopre circa l'80% della superficie terrestre e raggiunge profondità elevatissime, spesso impossibili da raggiungere per l'uomo. La robotica non poteva non essere la soluzione a questo limite, consentendoci di esplorare un ambiente misterioso ed alquanto

sconosciuto. Attraverso sistemi moderni e sempre più sofisticati è stato possibile raggiungere risultati notevoli in molti campi:

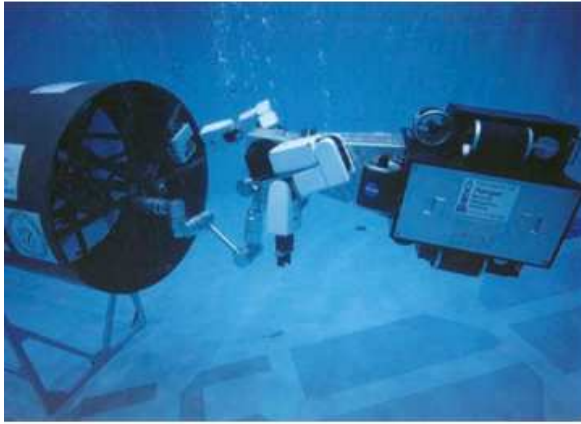


Figura 16. Braccio subacqueo in operazioni di riparazione

- Ricerca scientifica
- Recupero relitti
- Difesa
- Protezione dell'ambiente
- Archeologia sottomarina
- Sfruttamento delle risorse

4.3 Robotica militare

L'ambito militare è stato costantemente il settore che ha beneficiato per primo e maggiormente delle strutture più all'avanguardia. Gli scopi principali sono stati essenzialmente quello di garantire una maggiore sicurezza al personale ed una maggiore potenza agli armamenti impiegati in addestramento e direttamente in battaglia. Tramite la robotica è stato facile assicurare alle strumentazioni un completo controllo a distanza, ma anche la capacità di agire in autonomia per effettuare ricognizioni o ancor di più salvare vite umane. I *droni* sono stati i primi aerei privi di equipaggio, a cui si sono presto aggiunti robot in grado di disinnescare ordigni esplosivi. La ricerca non si è fermata qui, infatti nei laboratori si stanno mettendo a punto automi in grado di lavorare in gruppo per poter emulare efficacemente il comportamento di un esercito.



Figura 17. Arma in dotazione al governo americano

4.4 Robotica spaziale



Figura 18. Il rover Mars Sojourner in esplorazione sul suolo di Marte

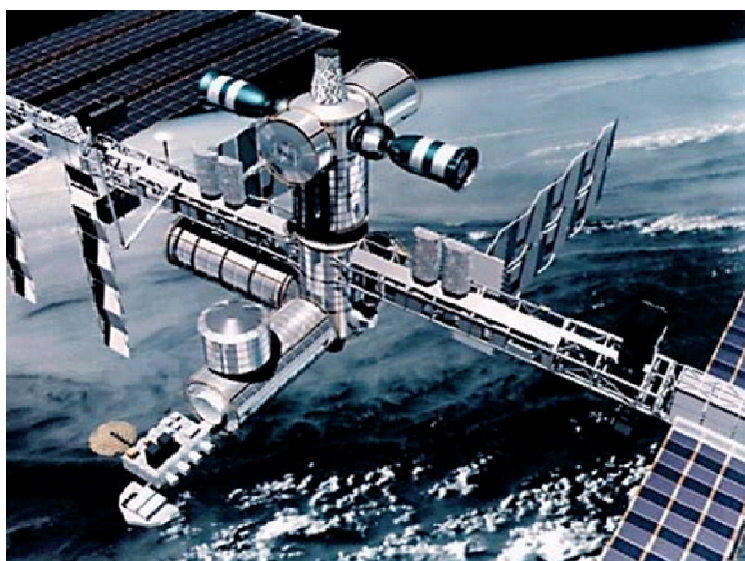


Figura 19. Stazione Spaziale Internazionale (ISS)

Lo sviluppo della robotica ha sempre avuto un legame diretto col settore dell'esplorazione dell'universo. L'uomo non essendo predisposto alla sopravvivenza in ambienti differenti da quello terrestre, in occasioni di questo genere ha dovuto in primo luogo salvaguardare la propria incolumità per poi dedicarsi con ulteriori difficoltà alla missione assegnata. Sistemi di esplorazione basati su tecnologie robotiche assicurano in primo luogo una minore spesa alle agenzie spaziali nazionali e maggiore sicurezza al personale umano, non più impiegato nelle situazioni più pericolose e difficili. Attualmente i viaggi spaziali sono molto lenti, pur viaggiando a velocità prossime a quelle della luce si impiegherebbero anni per raggiungere anche gli obiettivi più vicini al Sistema Solare, tempi impossibili per un essere umano. Un robot invece può essere facilmente sacrificato senza creare alcuno scrupolo a livello morale.

4.5 Robotica Industriale

Il settore industriale è stato sicuramente quello in cui i prodotti della robotica hanno trovato maggiore impiego per l'influenza di diversi fattori che hanno consentito di ottimizzare la produzione rispetto al passato:

- Versatilità di impiego
- Adattabilità a situazioni
- Precisione di posizionamento
- Ripetibilità di esecuzione

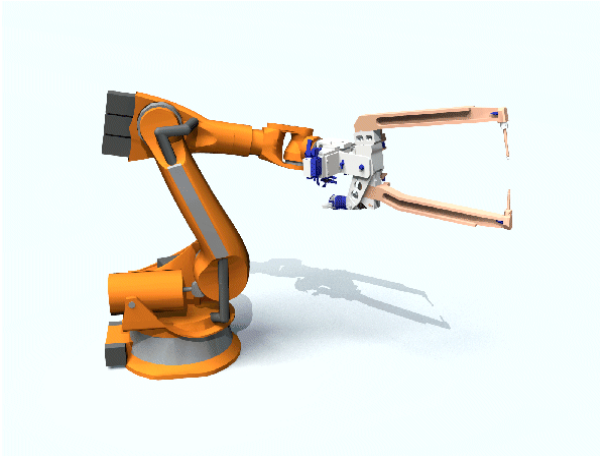
L'impiego dei robot in questo campo è divisibile in tre grandi gruppi:

- **Assemblaggio ed ispezione** → Area sviluppatasi principalmente nel corso dell'ultimo decennio, comprende oltre alle operazioni di montaggio anche l'ispezione del prodotto tramite particolari sensori di cui è dotato il robot
- **Movimentazione** → Comprende principalmente le applicazioni in cui il robot non esegue modifiche sulla struttura del prodotto, ma provvede a movimentarlo
- **Lavorazione** → Costituisce il settore più specifico della robotica industriale, in cui gli apparati robotici industriali eseguono sul prodotto operazioni quali la saldatura, il taglio e la verniciatura

Un'iniziale simulazione grafica tridimensionale dei processi industriali rende molto più efficace l'applicazione dei robot attraverso:

- analisi dei tempi operativi di ciclo
- programmazione fuori-linea e ottimizzazione
- progetto e verifica di layout 3D per evitare collisioni

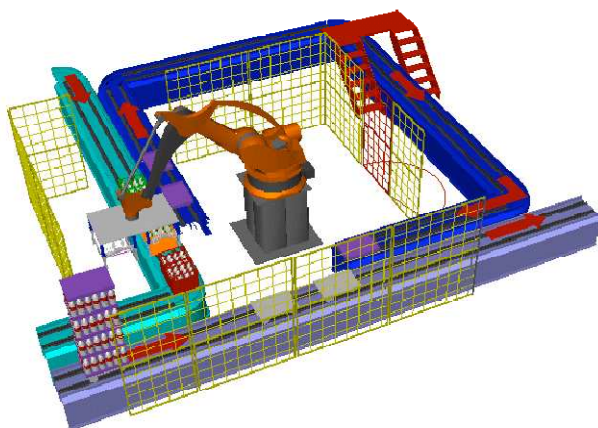
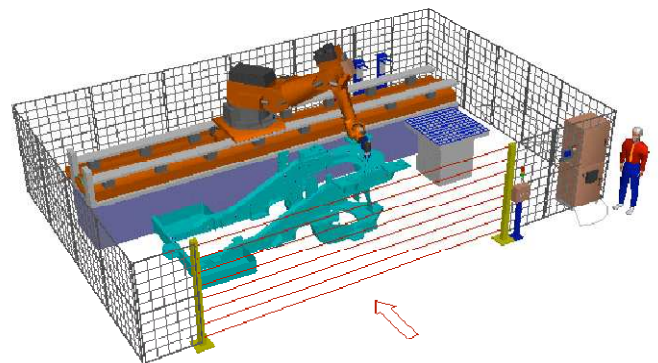




Principali operazioni di lavorazione:

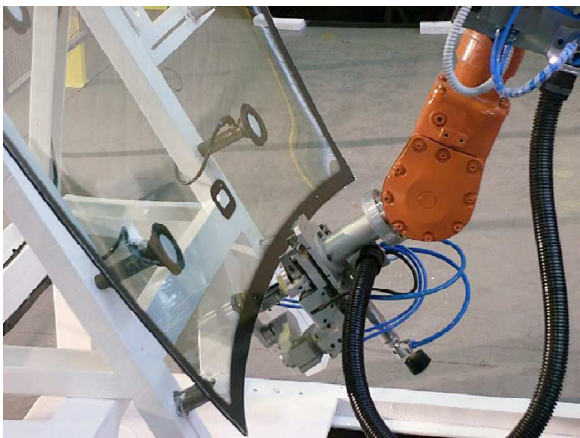
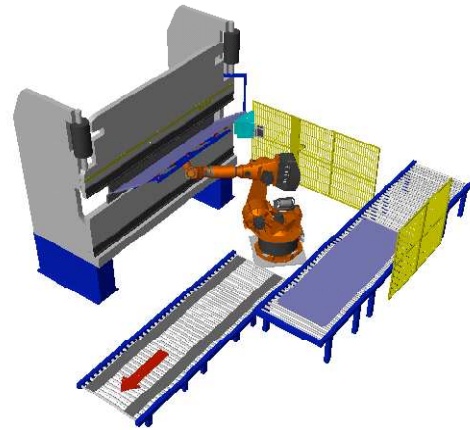
- Saldatura a punti con pistola servo-assistita

- Rivettatura (stud welding)



- Pallettizzazione

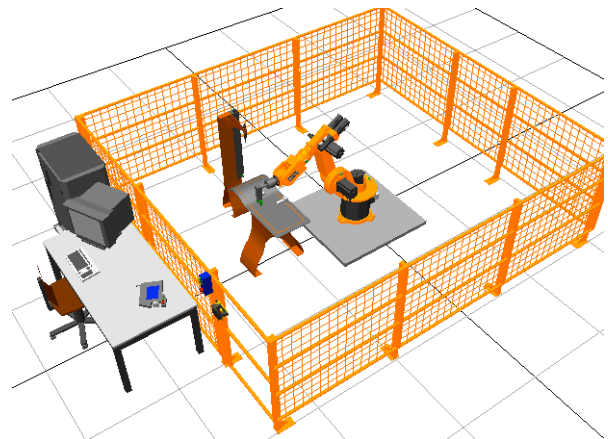
- Piegatura con caricamento della pressa



- Sbordatura di un parabrezza

Stazione di lavoro fuori linea:

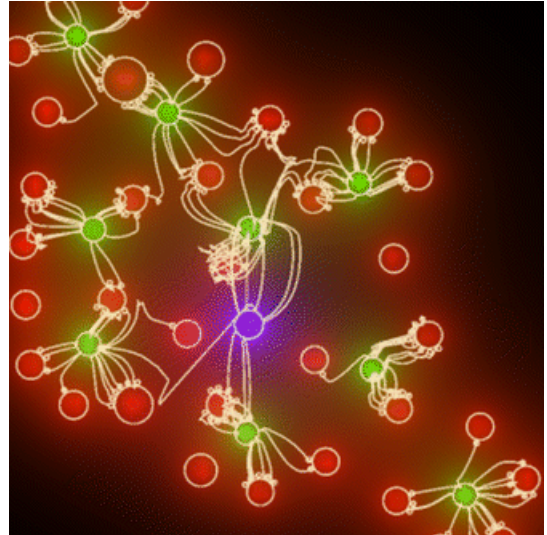
- Robot articolato in operazione di finitura di superficie



5.APPENDICE: Genetic Algorithms

5.1 Basic principles

Living organisms can solve many kinds of problems by using their brain's capacities that are the results of a slow and long process of evolution. Researchers, such as the American John Holland, have tried in many ways to give computer programs this versatility, in particular they have spent a lot of time to study a particular algorithm linked to the principles of natural selection. Software based on this type of structure would eliminate many hurdles don't let us creating an artificial intelligence.



Most organisms evolve essentially by two processes:

- Natural selection
- Sexual reproduction

The first one chooses the organisms that can survive and execute the second one by which they can create a new generation of organisms based on a recombination among the genes. Techniques, such as cross-over and mutation, create new series of chromosomes and therefore new creatures, different from their parents.

Translating these procedures to computer programs has never been an easy procedure. The first attempt was in the late 1950's and early 1960's, but it failed because it was too near to mutation and too far from mating. Only Hans J. Bremermann succeeded to produce a simple kind of mating, which was based on the sum of corresponding characteristics. It was a limit and not an improvement!

Some years later John Holland, studying mathematical models, developed a new programming technique called "Genetic Algorithms" and very similar to the principles of biology, in particular of the DNA.

The new system had a specific organization, consisting in a set of rules linked to actions through their conditions. These features were represented by strings of bits, in particular 1 for presence and 0 for absence. For example, if I consider a man I will find the value 1 in bits corresponding to “clever”, “conscientious” and “hairy”, but 0 in those ones corresponding to “metallic” or “yellow”. It is useful to program by FORTRAN or LISP languages to create such systems in which it is given to every solution of the population a degree of quality. High-quality strings mate; low-quality ones perish. After some generation it will be easy to see that only best solutions have survived.

The large variety of solutions can be divided in many regions, over which there are genetic algorithms as nets that control the population’s evolution and select the best strings.

After their mating, the program divide them randomly and exchange the two parts on the right of the point considered and called cross-over point. This method of recombination is the base of the formation of a zygote from the two gametes in a living organism.

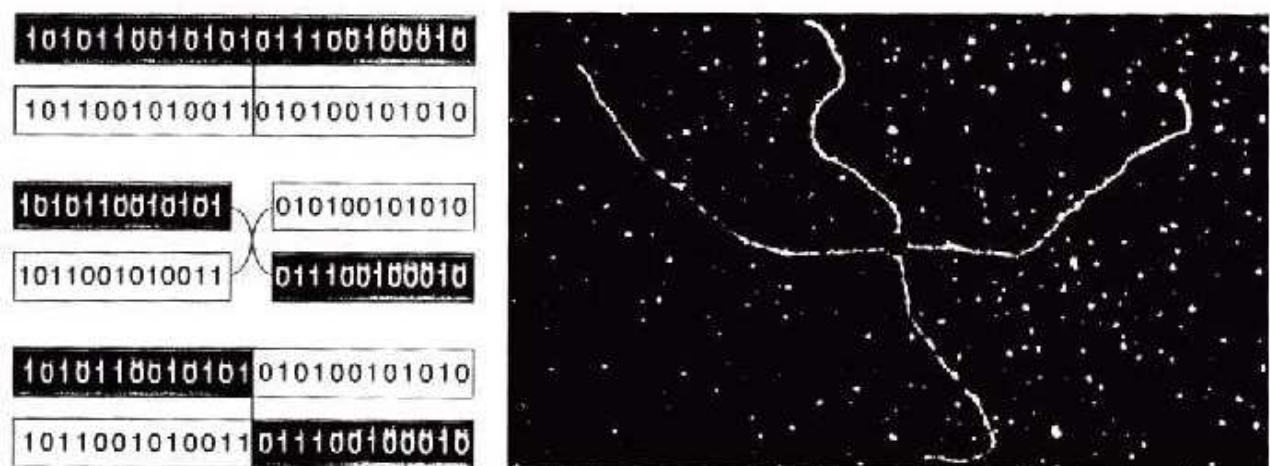


Figure 20. Example of crossover

The last process is the mutation that flips only a small fraction of the strings ($1/10000$) from 0 to 1, or vice versa. Mutation isn’t sufficient to find the last solution, but is necessary to avoid the development of an uniform population.

The genetic search is the result of a balance between **exploitation** and **exploration** of the population. They correspond respectively to a local and a general search, in fact using only one we couldn’t arrive to correct solutions. Exploitation can show an incorrect solution, exploration is more powerful but without the other process is really slow.

5.2 Representation of a problem

The function that can transform solutions in chromosomes and so in genes is:

$$C:S \rightarrow X$$

S is the range of solution of the problem and X the range of chromosomes where the algorithm have to work and give the value c equal to a particular chromosome corresponding to the solution s .

$$c = C(s), s \in S$$

5.3 Complete process of a genetic algorithm

Here is described how a new generation $P(t+1)$ is created from a population $P(t)$:

1. *Valuation*: the code evaluates the quality of every individual by the fitness function
2. *Selection for reproduction*: the best individuals are selected for the reproduction, they are put into an intermediate population P_1 and then into the mating pool
3. *Crossover*: genes of the new generation are modified by the technique of crossover so P_1 become P_2
4. *Mutation*: A little part of chromosomes is modified by the process of mutation with a probability of $[0.001, 0.01]$, P_2 is transformed in P_3
5. *Selection for replacement and survival*: $P(t+1)$ now corresponds with P_3 , but also other individuals can be in it, in fact many genetic algorithms can work at the same time

5.4 Fitness function

A fundamental feature of a genetic algorithm is the fitness function, which is a model that gives a fitness value to each chromosome. Solutions with the highest value among the population represent the best individuals that will be able to reproduce for the creation of a better generation.

Example 1.

A robot is at the entrance of a labyrinth, it must go through it and then go out. From the beginning to the exit there are 100 passes. The possible movements for each pass are four (F forward, B backward, R right, L left) so a generic sequence can be:

FRLFBLF.....F

Encoding each move as 2 bits, every possible solution will be a string of 200 bits (2 bits x 100 moves). In this case the fitness function can be the distance from the exit to the position of the robot after the 100 moves.

Example 2.

If we want to find the maximum of the function

$$y = x + |\sin(2x)|$$
$$0 \leq x \leq \pi$$

we can't derive it in fact there is the module. The value of x can vary between 0 and 3.14, so with a precision of 1/100 there will be 314 possible values and strings with 9 bits ($2^9 = 512$ e $2^8 = 256$). Fitness function coincides with y , that have to assume the biggest value.

5.5 Proportional selection

It is the most used strategy. The actual population has got n chromosomes:

$$P(t) = \{x_1, x_2, x_3, \dots, x_n\}$$

Total fitness is equal to:

$$F = \sum_{i=1}^n f(x_i)$$

Instead the probability of selection the chromosome x_i is:

$$p_i = \frac{f(x_i)}{F}$$

A wheel is divided in n sectors, each one corresponding to one chromosome and as large as the probability of his selection. The wheel have to turn n times to choose the chromosome that will be placed in the population P_1 .


5.6 Crossover


The process of crossover is really simple to understand, in fact it can be easily explained by some pictures.

There are two chromosomes (strings) of length L, so the code will choose a positive number k smaller than L. If the selected number is 3, the result is the following:

X  **1 0 1 1 0**

Y  **0 1 0 1 0**

X  **1 0 1 1 0**

Y  **0 1 0 1 0**

X'  **1 0 0 1 0**

Y'  **0 1 1 1 0**

The two parts on the right of crossover point are exchanged. This is the simplest kind of crossing over, it is based on a single point, but there are other ones more complex too.

6. Contenuto del CD

Il CD allegato al testo contiene un software di esempio da me realizzato in ambiente Visual Basic. Attraverso un codice molto semplice e di facile comprensione ho tentato di creare un banale prototipo di programma intelligente. Esso consiste in una griglia su cui si muove in maniera casuale un uomo, deciso a raggiungere il traguardo. Lo spostamento è complicato dalla presenza di alcune botole, che, posizionate dal software, riportano il personaggio al punto di partenza. Nel frattempo lo sfortunato, avendo memorizzato la posizione della trappola, al successivo passaggio sarà in grado di evitarla e quindi di terminare il percorso “sano e salvo” dopo un certo numero di tentativi.

BIBLIOGRAFIA

Baldi Livio, Cerofolini Alessandro, *La legge di Moore e lo sviluppo dei circuiti integrati*, Mondo digitale n°3, settembre 2003

Barr, A., Feigenbaum, E. A., *The Handbook of Artificial Intelligence* (vol. 1), William Kaufmann, Los Altos (Calif.) 1981

Beeby, S., Kraft, M., Ensell, G., White, N., *MEMS Mechanical Sensors*, pag. 95

Carlucci, Alello Lucia, Maurizio, Dapor, *Intelligenza artificiale: i primi 50 anni*, Mondo digitale n°2, giugno 2004

Bernstein, J., *Uomini e macchine intelligenti*, Adelphi, Milano 1990

Chomsky, N., *Saggi linguistici. Vol. 2 – La grammatica generativa trasformativa*, Bollati Boringhieri, Torino 1979

Cook, V., Newson, M., *Grammatica universale. Introduzione a Chomsky*, Il Mulino, Bologna 1996

De Luca, Alessandro, *Corso di Robotica*, Roma 2007/2008

Dobrev, Dimiter, *Intelligenza artificiale Una definizione*, PC Magazine Bulgaria

Ferrari, Stefano, Piuri, Vincenzo, *Le reti neurali*

Frixione, M., *Logica, significato e intelligenza artificiale*, Franco Angeli, Milano 1994

Gori, Marco, *Introduzione alle reti neurali*, Mondo digitale n°4, dicembre 2003

Grazia, Carlo Umberto, *Introduzione ai sistemi esperti*

Haugeland, J., *Intelligenza Artificiale*, Bollati Boringhieri, Torino 1988

Hobbes, T., *The Leviathan*, (tr. it *Il Leviatano*, Editori Riuniti, Roma 1982)

Hodges, W., *Logica*, Garzanti, Milano 1986

Locagnina, Valerio, *Algoritmi genetici*

Lazzerini, Beatrice, *Introduzione agli algoritmi genetici*

Lolli, G. (a cura di), *Mente e macchine*, «Le Scienze quaderni», n. 66, giugno 1992

McCorduck, P., *Storia dell'intelligenza artificiale. Gli uomini, le idee, le prospettive*, Franco Muzzio, Padova 1987

McCulloch, W. S., Pitt, W. S., *A Logical Calculus of the Ideas Immanent in Nervous Activity*, in «Bul. Math. Biophys.», vol. 5, 1943, p. 115-133

Melchiorri Claudio, *Robotica e Automazione*

Minsky, M., (a cura di), *Semantic Information Processing*, MIT Press, Cambridge (Mass.) 1968

Minsky, M., *Framework for Representing Knowledge*, in Winston 1975 (tr. it. in Haugeland 1989)

Mishkoff, H. C., *Understanding Artificiale Intelligence*, Sams, Indianapolis 1985

Operto Fiorella, *Introduzione alla Robotica*, Padova 2005

Parisi, D., *Mente. I nuovi modelli della Vita Artificiale*, Il Mulino, Bologna 1999

Pessa, E., *Intelligenza artificiale. Teorie e sistemi*, Bollati Boringhieri, Torino 1992

Picardi, E., *Le teorie del significato*, Laterza, Roma-Bari 1999

Pratt, V., *Macchine pensanti. L'evoluzione dell'intelligenza artificiale*, Il Mulino, Bologna 1990

Rich, E., *Intelligenza Artificiale*, McGraw-Hill Italia, Milano 1986

Rolston, D. W., *Sistemi esperti. Teoria e sviluppo*, McGraw-Hill Italia, Milano 1991

Rose, S., *Il cervello e la coscienza*, Mondadori, Milano 1977

Sandler, Ben-Zion, *ROBOTICS Designing the Mechanisms for Automated Machinery*, Academic Press, London 1999, pagg. 1-6, 75-76


Schank, R. C., Abelson, R. P., *Script, Plans, Goals, and Understanding*, Lawrence Erlbaum, Hillsdale (N.Y.) 1977

Schank, R. C., *Il computer cognitivo*, Giunti, Firenze 1989

Searle, J., *Mente, cervello, intelligenza*, Bompiani, Milano 1999

Searle, J., *Menti cervelli e programmi*, Clup-Clued, Milano 1984

Searle, J., *La mente è un programma?*, in «Le Scienze quaderni», n. 66, giugno 1992



Silvi Antonini, S., *Vita artificiale. Dal Golem agli automi cellulari*, Apogeo, Milano 1995

Smolensky, P., *Connessionismo tra simboli e neuroni*, Marietti, Milano 1992

Turing, A. M., *Computing Machinery and Intelligence*, in «Mind», Vol. 59, 433-60, 1950 (tr. it *Macchine calcolatrici e intelligenza*, in Somenzi e Cordeschi 1986)

Winograd, T., *Calcolatori e conoscenza*, Mondadori, Milano 1987

Winograd, T., *Understanding Natural Language*, University Press - Academic Press, New York 1972

Zaccaria, Renato, *Aspettando Robot*, Mondo digitale n°3, settembre 2003

SITOGRAFIA

Test di Turing: www.nemesi.it

Codice del simulatore della macchina di Turing: www.pierotofy.it

Intelligenza Artificiale: www.mediamente.rai.it

www.strategiaglobale.com

Sito del progetto di eccellenza OMAROBOT

Revolution education Ltd: www.rev-ed.co.uk