Programmi e programmazione

Lezione n. 4

Dall'algoritmo al programma

- L'algoritmo è l'idea risolutiva con caratteristiche di rigore (non brevettabile, non tutelabile dal diritto d'autore)
- Si rappresenta mediante tecniche formali (pseudocodifica, diagrammi a blocchi, diagrammi di attività, etc.)
- Il formalismo si traduce in un linguaggio simbolico
- Il linguaggio simbolico si traduce in linguaggio macchina (0 e 1 – codice binario)

Il Programma

- Descrizione di un algoritmo
  - effettuata tramite un linguaggio artificiale detto linguaggio di programmazione
  - dove l'esecutore è un dispositivo tecnologico come un elaboratore elettronico
  - ottenuto tramite un procedimento di traduzione (algoritmico) effettuato da programmi traduttori classificabili in compilatori e interpreti
  - Il linguaggio di programmazione è dato da dati e istruzioni organizzate in strutture di controllo

Il computer esecutore di programmi

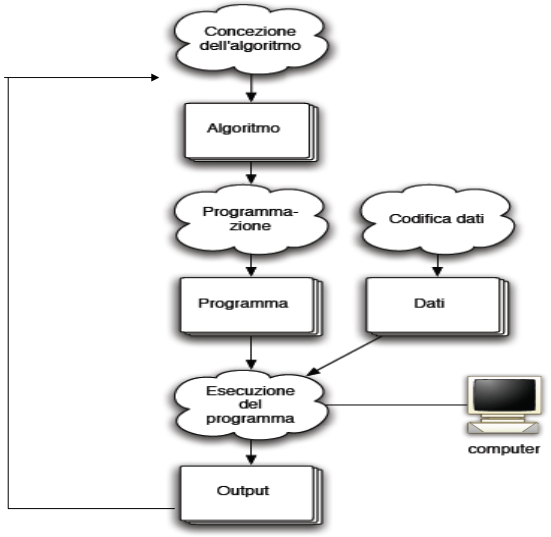


Figura 3.15: *Il computer quale mero esecutore*

- Diagrammi a blocchi
- Editor per scrivere il programma
- Traduzione in linguaggio macchina
- Esecuzione detta *run-time*
- Correzione dei bugs

Ciclo di creazione del codice eseguibile

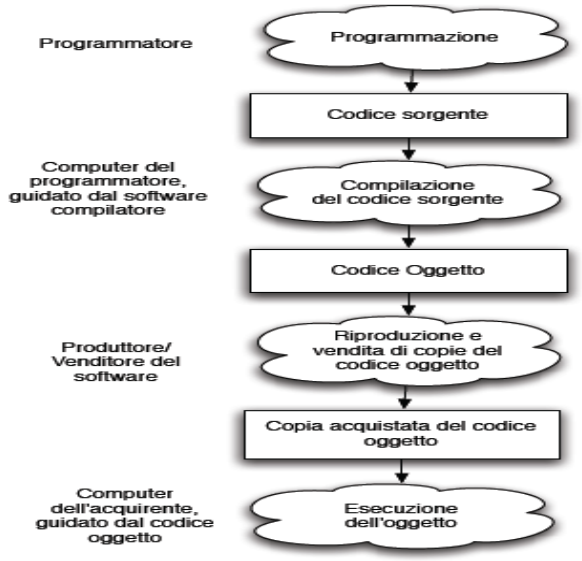


Figura 3.11: *La compilazione del software*

Evoluzione della programmazione

- Programma: algoritmo formulato in un linguaggio di programmazione
- Agli inizi dell'informatica si programmava in linguaggio macchina direttamente. Successivamente si utilizzò l'Assembler e solo successivamente linguaggi di alto livello
- Computer + linguaggio di programmazione = macchina virtuale che esegue i programmi scritti nel linguaggio di programmazione

Il linguaggio macchina

- Il linguaggio comprensibile ed eseguibile da parte dell'hardware
- Consiste in:
  - comandi eseguibili dall'hardware
  - espressi in codici binari
- Il linguaggio macchina è
  - dipendente dall'hardware (un linguaggio macchina può essere usato solo su un computer)
  - difficile da comprendere per l'uomo

Il linguaggio assembler

- Assembler: abbreviazione simbolica del linguaggio macchina
  - le istruzioni consistono di brevi parole dette "codici mnemonici" (che possono essere ricordate più facilmente dei corrispondenti valori binari).

Istruzioni in linguaggio assembler

- Somma i numeri memorizzati agli indirizzi di memoria 8 e 5, e memorizza il risultato all'indirizzo 10.

LOAD 8	Prendi il numero registrato all'indirizzo 8 della memoria e caricalo in ALU
ADD 5	Aggiungi il numero all'indirizzo 5 a quello in ALU
STORE 10	Prendi il numero in ALU e registralo in memoria, all'indirizzo 10

Linguaggio assembler e linguaggio macchina

LOAD 8 = 010100 001000

LOAD (010100) =	codice dell'operazione (opcode)
8 (001000) =	Indirizzo dell'operando

Linguaggio di alto livello

- Linguaggio di alto livello
  - indipendente dalla macchina
  - opera per istruzioni complesse
  - crea una barriera fra l'hardware e l'utente
  - crea una macchina virtuale ovvero opera un'astrazione dell'hardware sottostante
  - più vicino al linguaggio umano

- Ogni linguaggi di alto livello è composto da
  un vocabolario – livello lessicale
  una grammatica – livello sintattico
  una semantica – livello semantco
  Mettere la formula di composizione dei linguaggi

In un linguaggio di programmazione di alto livello - Pascal

```
program areaRettangolo;
   var h, b, a: integer;
   BEGIN
        write ('altezza= ');
        readln (h);
        write ('base= ');
        readln (b);
        a:=b*h;
        writeln ('Area del rettangolo = ', a);
        readln;
   END.
```

C++ area del triangolo

```
* Calcolo area triangolo e rettangolo */

   #include <stdio.h>
   #include <stdlib.h>

   main ()
   {
        int scelta;
         float base, altezza, area;
         printf("inserire valore base: ");
         scanf("%f", &base);
         printf("\ninserire valore altezza: ");
         scanf("%f", &altezza);
         area = (base*altezza)/2;
        printf("\n\nArea: %f\n\n\n", area);
   }
```

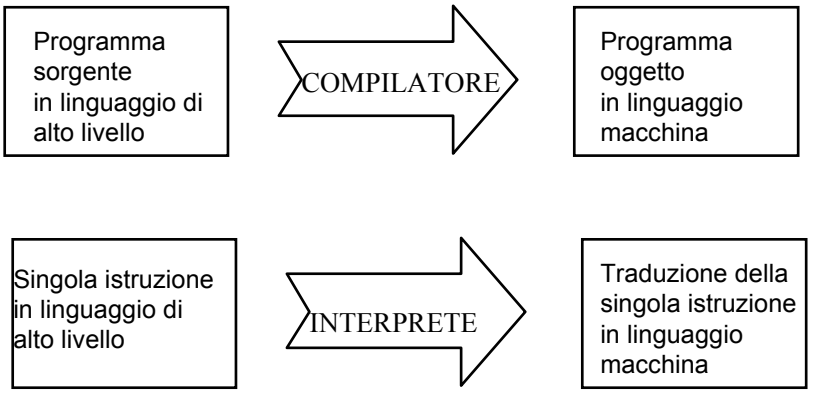
Dal linguaggio macchina ai linguaggi di alto livello

- Linguaggio macchina:
  - puo' essere eseguito direttamente dall'hardware
- Linguaggio assembler:
  - viene tradotto in linguaggio macchina da un software chiamato assemblatore (assembler)
- Linguaggio di altro livello:
  - viene tradotto nel linguaggio macchina da un interprete o un compilatore (traduttore)

Inteprete o compilatore

- Compilatore:
  - traduce l'intero programma (il codice sorgente) in un programma in linguaggio macchina (il codice oggetto o codice eseguibile)
  - Il codice oggetto viene memorizzato per essere usato nel seguito
- Interprete:
  - Finché non è stato eseguito tutto il programma
    - traduce una riga per volta
    - la esegue immediatamente l'istruzione
    - dimentica quanto tradotto
    - passa alla riga successiva

Compilatore ed inteprete

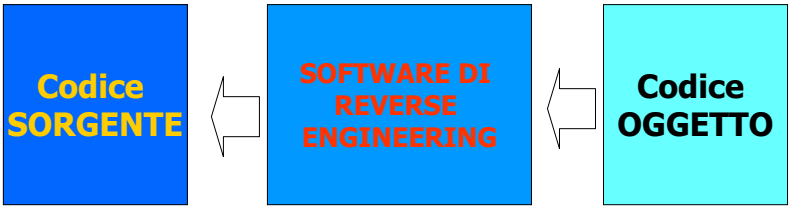


Codice sorgente e oggetto

- Il programma scritto in linguaggio Assembler o in linguaggio di alto livello viene detto programma sorgente o CODICE SORGENTE
- Il programma sorgente è l'input dato ai traduttori che producono in uscita il programma in linguaggio macchina o CODICE OGGETTO

Reverse engineering

- Il programma scritto in linguaggio macchina viene mediante di tool (programmi) convertito in programma sorgente
- Vietato salvo casi particolari: art. 64, quater L. 633/41 garantire l'interoperabilità dei software ma solo se non è possibile ovviare in altro modo



Art. 64-quater L. n. 633/41

Art. 64-quater

1. L'autorizzazione del titolare dei diritti non é richiesta qualora la riproduzione del codice del programma di elaboratore e la traduzione della sua forma ai sensi dell'art. 64-bis, lettere a) e b), compiute al fine di modificare la forma del codice, siano indispensabili per ottenere le informazioni necessarie per conseguire l'interoperabilità, con altri programmi, di un programma per elaboratore creato autonomamente purché siano soddisfatte le seguenti condizioni:
   a) le predette attività siano eseguite dal licenziatario o da altri che abbia il diritto di usare una copia del programma oppure, per loro conto, da chi é autorizzato a tal fine;
   b) le informazioni necessarie per conseguire l'interoperabilità non siano già facilmente e rapidamente accessibili ai soggetti indicati alla lettera a);
   c) le predette attività siano limitate alle parti del programma originale necessarie per conseguire l'interoperabilità.

Art. 64-quater L. n. 633/41

Art. 64-quater

2. Le disposizioni di cui al comma 1 non consentono che le informazioni ottenute in virtù della loro applicazione:
   a) siano utilizzate a fini diversi dal conseguimento dell'interoperabilità del programma creato autonomamente;
   b) siano comunicate a terzi, fatta salva la necessità di consentire l'interoperabilità del programma creato autonomamente;
   c) siano utilizzate per lo sviluppo, la produzione o la commercializzazione di un programma per elaboratore sostanzialmente simile nella sua forma espressiva, o per ogni altra attività che violi il diritto di autore.
3. Le cause contrattuali pattuite in violazione dei commi 1 e 2 sono nulle.
4. Conformemente alla convenzione di Berna sulla tutela delle opere letterarie ed artistiche ratificata e resa esecutiva con legge 20 giugno 1978, n. 399, le disposizioni del presente articolo non possono essere interpretate in modo da consentire che la loro applicazione arrechi indebitamente pregiudizio agli interessi legittimi del titolare dei diritti o sia in conflitto con il normale sfruttamento del programma.

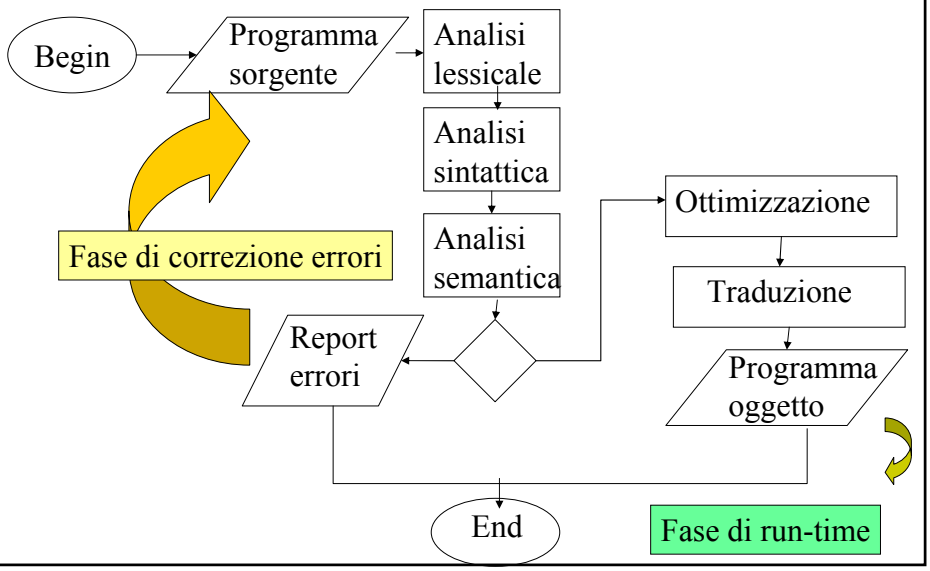
Compilatori nel dettaglio

- I compilatori eseguono la traduzione di TUTTO il codice sorgente
  - lo schema di compilazione elabora il programma nella sua interezza
  - esamina gli errori considerando tutte le istruzioni del programma e quindi in modo più completo
  - l'esecuzione avviene solo alla fine di tutta la traduzione
  - nel momento che si esegue un programma compilato si è certi che i controlli sintattici e grammaticali sono stati eseguiti e che il programma non darà errori di questo tipo durante l'esecuzione
  - ottimizzazione dell'esecuzione dal punto di vista della gestione delle risorse del calcolatore

Schema compilativo (i)

1. Analisi lessicale - viene controllato che le unità lessicali o token appartengono al linguaggio
2. Analisi sintattica - viene controllato che le unità lessicali nell'ordine dato costituiscano un'istruzione riconoscibile all'interno del linguaggio
3. Analisi semantica - viene controllato che le istruzioni date siano fra loro congrue rispetto alle regole del linguaggio
4. Ottimizzazione - vengono eliminate le inefficienze

Schema compilativo (ii)



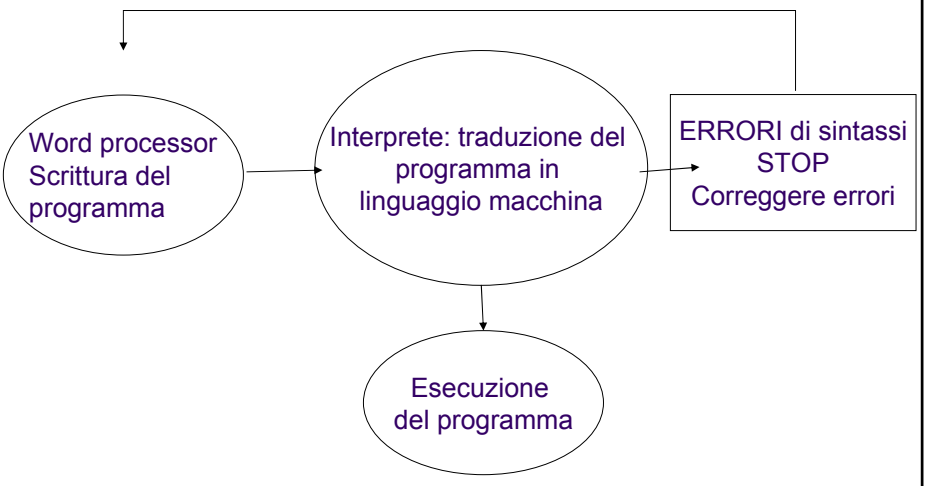
La fase di correzione, run-time, debugging

- Se la compilazione finisce con la segnalazione di errori gravi occorre eseguire il *debugging* ossia togliere i bug
- Solitamente ogni ambiente di sviluppo software SDK (software development kit) comprende al suo interno un editor e uno strumento per eseguire il *debugging* ossia vedere in fase di esecuzione le variabili che valori assumono per comprendere con uno strumento comprendere dove l'errore si è verificato, ripercorrere la soluzione dal punto di vista logico, trovare l'errore concettuale, risistemare il codice sorgente e ripetere la compilazione

Interpreti nel dettaglio

- Gli interpreti eseguono la traduzione e l'esecuzione di una istruzione per volta
  - lo schema di interpretazione elabora il programma istruzione per istruzione
  - esamina gli errori passo passo
  - l'esecuzione avviene contestualmente dopo la traduzione
  - nel momento che si esegue un programma interpretato è possibile riscontrare errori durante l'esecuzione
  - non c'è ottimizzazione

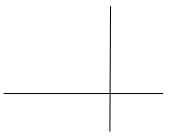
Flusso dell'interprete



Tipologia di linguaggi (i)

- Linguaggi procedurali
  - Fortran - 1954
  - Cobol - 1959
  - RPG - 1960
  - Algol - 1965
  - PL/1 -1965
  - BASIC - 1965 - interpretato e compilato
  - Pascal - 1971 - interpretato e compilato
  - C - 1972

Tipologia di linguaggi (ii)



- Linguaggi logici o descrittivi o funzionali
  - Lisp
  - Prolog
- Linguaggi orientati agli oggetti - object-oriented
  - Simula 67 - Eiffel
  - C++
  - Java
- Linguaggi di scripting
  - CGI
  - JavaScript
  - Perl
  - PHP

Generazione dei linguaggi

- Prima generazione
  - linguaggio macchina
- Seconda generazione
  - Assembler
- Terza generazione
  - linguaggi procedurali
- Quarta generazione
  - linguaggi di query
- Quinta generazione
  - linguaggi funzionali – programmazione dichiarativa
  - programmazione ad agenti

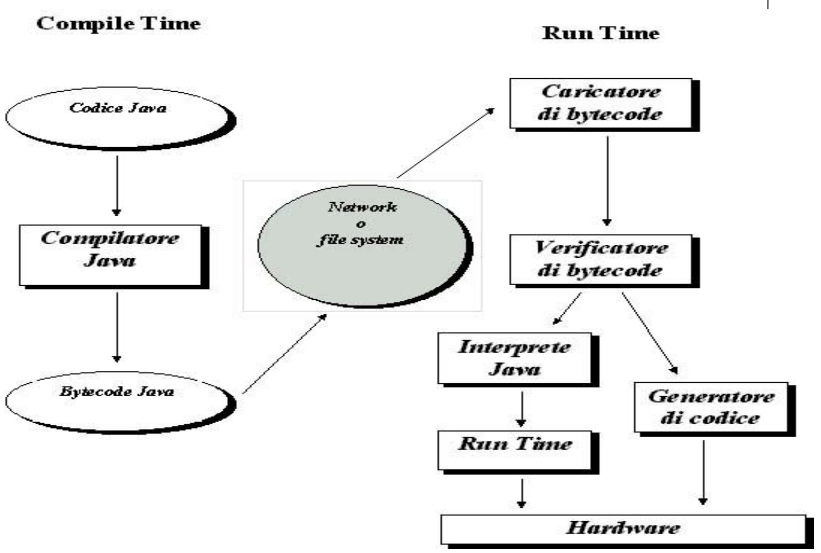
Java: un esempio di linguaggio ibrido

- Java nasce nel 1991 da un gruppo di sviluppatori della SUN (Green Team)
- È un linguaggio ad alto livello
- Orientato agli oggetti
- Indipendente dalla piattaforma ossia non dipende dal processore, ed in genere dall'hardware, e dal sistema operativo della macchina su cui dovrà funzionare
- È anche un linguaggio anche di scripting (applet) ossia si possono costruire applicazioni che funzionano tramite pagine web ed eseguibili dal browser
- Funziona sulla base di uno strato la Java Virtual Machine (JVM) che rende indipendente il codice dalla piattaforma di esecuzione

Java: un esempio di linguaggio ibrido

- Java è un linguaggio che viene compilato producendo un codice intermedio detto BYTECODE
- Il bytecode viene poi interpretato dalla JVM per trasformare tale codice in codice eseguibile sulla piattaforma di riferimento
- Esistono JVM per ogni piattaforma (Solaris, Windows, Linux, Macintosh, ecc.)
- Questo consente di creare codice sorgente di alto livello indipendente dalla piattaforma di esecuzione
- Contemporaneamente però la JVM è lenta in quanto interpretata e soprattutto l'intero processo di esecuzione è fortemente sensibile alla buona esecuzione della JVM

Processo di traduzione di Java



Sintesi

- Calcolatore come esecutore
- Calcolatore come ragionatore
- Calcolatore come modello del cervello
- Rete di calcolatori come modello della società – *social networking*
- Questi modelli sollevano nuovi quesiti giuridici: autonomia, responsabilità, perdurabilità nel tempo, etc.

Il computer esecutore di programmi

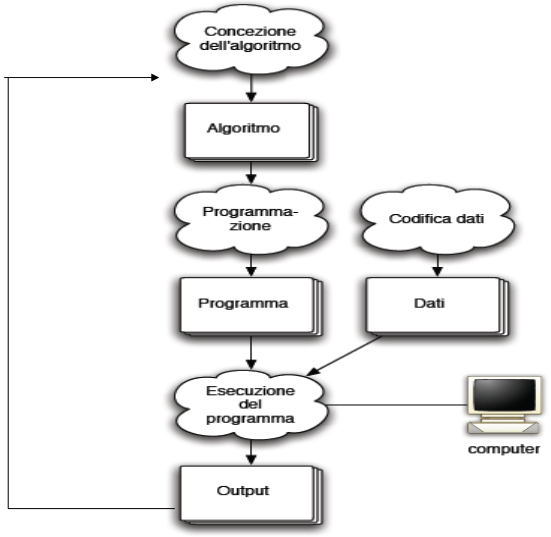


Figura 3.15: *Il computer quale mero esecutore*

- Diagrammi a blocchi
- Editor per scrivere il programma
- Traduzione in linguaggio macchina
- Esecuzione detta *run-time*
- Correzione dei bugs

Il computer come ragionatore

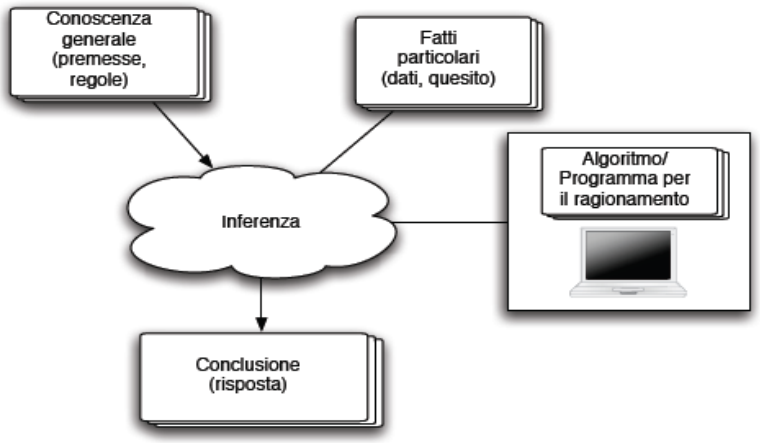


Figura 3.17: *Il computer quale ragionatore*

Il computer come ragionatore

- Il computer in questo caso è chiamato non ad eseguire operazioni ma a ragionare sopra uno specifico caso usando un modello di ragionamento logico predeterminato dal programmatore e sulla base di regole precedentemente inserite
- Inserendo un caso concreto e chiedendo se il caso rientra nelle regole poste, il ragionatore potrà dedurre una risposta positiva o negativa fornendo anche la giustificazione (dimostrazione)
- Esempio:

```
SE [la persona x è disoccupata] E
   ( [la persona x ha figli minorenni a carico] O
   [la persona x ha persone disabili a carico] ) E
   [la persona x ha un reddito annuo < di 20.000 euro]
ALLORA
   [la persona x ha diritto ad uno sgravio fiscale]
```

- Caso concreto: Maria, madre disoccupata di tre figli con reddito 17.000 euro ha diritto allo sgravio

Il computer come modello del cervello

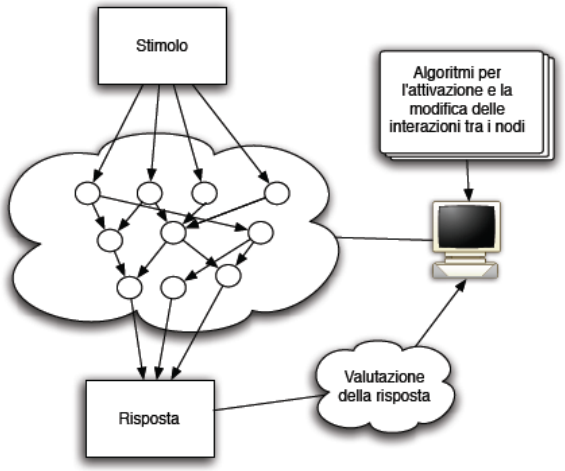


Figura 3.18: *Il computer quale modello del cervello*

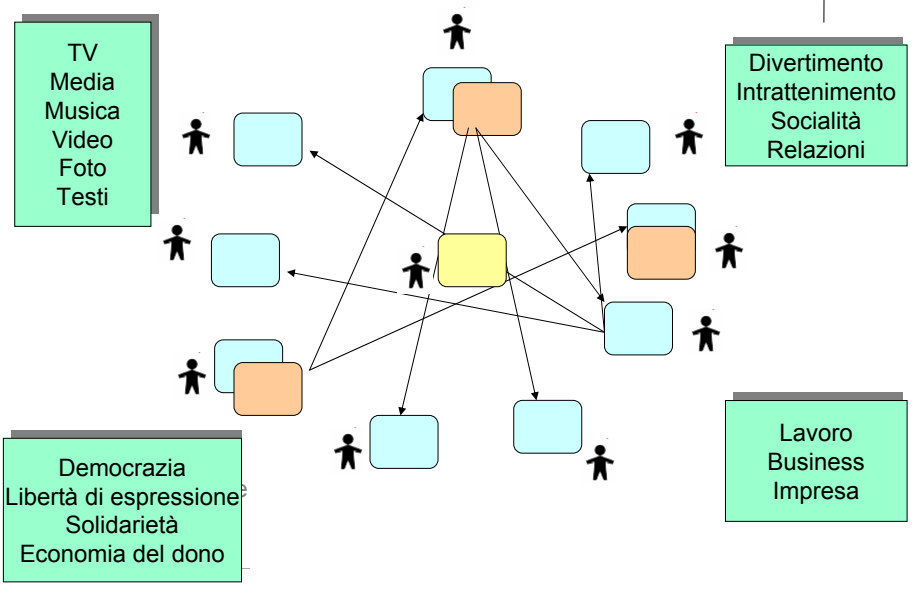
Il computer come modello del cervello: reti neurali

- In questo caso il computer non si limita ad emulare un modello di ragionamento predefinito dal programmatore ma è dotato dal programmatore stesso di elementi di comportamento tali da emulare il cervello umano nel prendere decisioni
- Le reti neurali simulano le sinapsi del cervello e quindi, per ambiti ristretti, il computer può, sollecitato dagli eventi esterni, decidere in autonomia secondo schemi non predefiniti
- Il computer può anche correggere i propri errori o trarre conclusioni diverse sulla base delle esperienze accumulate
- Utilizzi: ambito finanziario, riconoscimento di figure, giochi, robot per lavori pericolosi

Il computer come modello del cervello: agenti intelligenti

- In questo caso il computer non si limita ad emulare il cervello umano nel prendere decisioni ma si cerca di insegnargli anche capacità comportamentali: autonomia, reattività, pro-attività, flessibilità, intelligenza, mobilità, predurabilità nel tempo (Sartor, Gli agenti software: nuovi soggetti del ciberdiritto?)
- Gli agenti intelligenti possono anche aggregarsi per portare a termine un compito formando sistemi multi-agenti
- In questo modo gli agenti simulano il lavoro di gruppo ossia una cooperazione o coreografia
- Utilizzi: negoziazione, comparazione in rete di prezzi, raccolta informazioni, risoluzione di dispute on-line

La rete di computer come modello della società



Il computer come modello della società

- La rete di computer consente di creare luoghi virtuali dove la società prendere forma in modo diverso creando nuove opportunità, istituzioni, regole, comportamenti
- La cooperazione, la condivisione di materiali, la possibilità di costruire contenuti nella rete come soggetti attivi e non più passivi fa della rete un luogo sociale simile ad una agorà distribuita
- Il risultato è dato da una *intelligenza collettiva* [John Searle] la cui qualità è proporzionale al quadrato dei nodi della rete [legge di Metcalfe]
- Marketing virale, economia del dono, rinsaldo dei legami sociali, valore del sé, palcoscenico su cui azionare dei ruoli, condivisione delle esperienze fra pari, veicolazione della conoscenza, capitalizzare le relazionali sono tutti aspetti del successo dei social networking

Casi di successo dell'uso di social networking

- Standard Chartered Bank, una banca di Londra – hanno installato un Facebook privato interno dedicato ai dipendenti
- Indesit ha installato due social network per il team tecnico e per il team di ricerca
- Fiat ha affiancato nel suo sito un avatar, Chiara, che guida nella composizione dell'ordine
- Volkswagen e AlfaRomeo hanno un sito che simula un social network fra i clienti
- LinkedIn riconfigura il modo di trovare lavoro

Conclusioni

"The question is not whether intelligent machines can have emotions, but whether machines can be intelligent without any emotions"

Marvin Minsky, The Society of Mind 1985, (traduzione italiana, La società della mente, Adelphi, 1989)

Domande

- Cosa è un programma
- Come si trasforma un algoritmo in un programma
- Cosa è un linguaggio ad alto livello, il linguaggio assembler e il linguaggio macchina
- Cosa è il codice sorgente e il codice oggetto
- Tipi di linguaggi di programmazione
- Cosa è il compilatore e l'interprete: caratteristiche e comportamenti
- Cosa è il calcolatore come esecutore, come ragionatore, come modello della mente umana, come insieme di agenti intelligenti, come rete sociale