

## SISTEMI ESPERTI (1)

---

- **Sistemi Basati Sulla Conoscenza (1980)**
- Un sistema **basato sulla conoscenza** è un sistema in grado di risolvere problemi in un **dominio limitato** ma con prestazioni **simili** a quelle di un **esperto** umano del dominio stesso.
- Generalmente esamina un largo numero di possibilità e costruisce dinamicamente una soluzione.
- *“La potenza di un programma intelligente nel risolvere un problema dipende primariamente dalla **quantità e qualità** di conoscenza che possiede su tale problema”. (Feigenbaum)*

1

## SISTEMI ESPERTI (2)

---

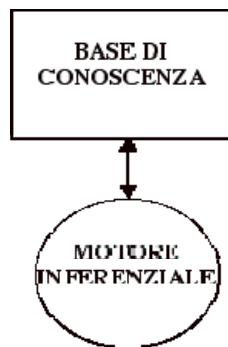
- Il programma non è un insieme di **istruzioni immutabili** che rappresentano la soluzione del problema, ma un ambiente in cui:
  - rappresentare;
  - utilizzare;
  - modificare;
  - una **base di conoscenza**.
- Caratterizzato dalle seguenti **proprietà**:
  - *Generalità.*
  - *Rappresentazione esplicita della conoscenza.*
  - *Meccanismi di ragionamento.*
  - *Capacità di spiegazione.*
  - *Capacità di operare in domini mal strutturati.*

2

## PRINCIPI ARCHITETTURALI

---

- Ogni sistema basato sulla conoscenza deve riuscire ad esprimere **due** tipi di conoscenza in modo **separato e modulare**:
  - Conoscenza sul dominio dell'applicazione (**COSA**);
  - Conoscenza su **COME** utilizzare la conoscenza sul dominio per risolvere problemi (**CONTROLLO**).
- **Problemi:**
  - Come esprimere la conoscenza sul problema?
  - Quale strategia di controllo utilizzare?



3

## SISTEMI DI PRODUZIONE

---

- Un sistema a regole di produzione (production system) è costituito da tre componenti fondamentali:
  - **Base di conoscenza a regole** (che prende spesso il nome di “memoria a lungo termine”) in cui sono contenute le regole di produzione;
  - **Memoria di lavoro** (memoria a breve termine) in cui sono contenuti i dati e in cui vengono mantenute le conclusioni raggiunte dal sistema;
  - **Motore inferenziale.**
- Ogni regola di produzione ha la seguente forma:
  - if <condizione> then <conclusione/azione>

4

## STRATEGIE DI CONTROLLO

---

- Strategia forward (in avanti) o controllo guidato dai dati;

```
while <obiettivo non raggiunto> do
begin
<MATCH: determina l'insieme delle regole applicabili (cioè
  le regole il cui antecedente è soddisfatto dai fatti
  contenuti nella memoria di lavoro)>;
<CONFLICT_RESOLUTION: seleziona la regola da applicare>;
<FIRE: esegui l'azione associata alla regola>
end.
```

5

## STRATEGIE DI CONTROLLO

---

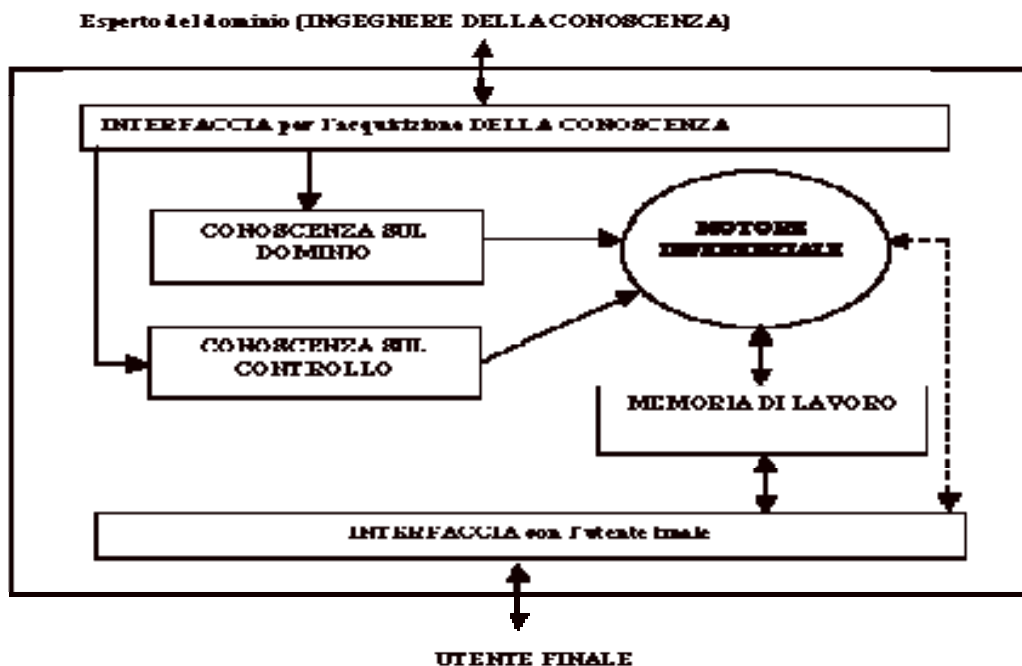
- Strategia "**backward**" (all'indietro) o controllo guidato dal goal G.

```
if <G è un fatto nella memoria di lavoro>
  then <G è dimostrato>
  else
begin
<MATCH: seleziona le regole il cui conseguente può essere
  unificato con G>
<CONFLICT_RESOLUTION: seleziona la regola da applicare>
<l'antecedente della regola selezionata diventa il nuovo
  obiettivo (congiunzione di obiettivi) da verificare>
end.
```

6

# SISTEMI BASATI SULLA CONOSCENZA ARCHITETTURA

- Rappresentazione della Conoscenza:



7

# SISTEMI BASATI SULLA CONOSCENZA ARCHITETTURA

- **Rappresentazione della Conoscenza:**
  - Regole;
  - Frames;
  - Proposizioni Logiche;
  - Vincoli;
  - Procedure;
  - Demoni;
  - Oggetti;
  - Fattori di Certezza, Variabili Fuzzy.....
- **Modalità di Inferenza:**
  - Ragionamento *forward*;
  - Ragionamento *backward*;
  - Risoluzione;
  - Propagazione di vincoli;
  - Strategie di ricerca euristiche;
  - Ragionamento Ipotetico ed Abduativo....

8

# INGEGNERIA DELLA CONOSCENZA AMBIENTI

---

- a) **Skeletal systems (SHELL)**
  - Ottenuti togliendo da Sistemi Esperti già costruiti la conoscenza propria del dominio e lasciando solo il motore inferenziale e le *facilities* di supporto.
  - EMYCIN (Empty MYCIN) deriva da MYCIN;
  - KAS deriva da PROSPECTOR;
  - EXPERT deriva da CASNET.
- b) **Sistemi general-purpose (TOOLS);**
  - KEE, ART, Knowledge-Craft, Nexpert, KAPPA, KAPPA PC, ...
  - non sono strettamente legati a una particolare classe di problemi: essi permettono una più ampia varietà di rappresentazione della conoscenza e strutture di controllo.
- c) **Linguaggi simbolici (Prolog, Lisp) e non (C, C++).**

9

---

## PASSI DI PROGETTAZIONE

---

- **IDENTIFICAZIONE** delle caratteristiche del problema.
- **CONCETTUALIZZAZIONE.**
- **FORMALIZZAZIONE:** progetto della struttura in cui organizzare la conoscenza.
- **IMPLEMENTAZIONE:** scrittura effettiva della conoscenza sul dominio.
- **VALIDAZIONE.**
  
- **SCELTA DI UN TOOL:**
  - non bisogna utilizzare un Tool più generale del necessario;
  - bisogna testare il Tool costruendo un piccolo prototipo del sistema;
  - bisogna scegliere un Tool che sia affidabile ed mantenuto da chi lo ha sviluppato;
  - quando il tempo di sviluppo è critico, è bene scegliere un Tool con *facilities* di spiegazioni/interazione incorporate;
  - bisogna considerare le caratteristiche del problema per determinare le caratteristiche che deve avere il Tool.

10

# APPLICAZIONI

---

- **Migliaia di Sistemi Esperti** nei settori più svariati.
  - **Interpretazione:** Si analizzano dati complessi e potenzialmente rumorosi per la determinazione del loro significato (Dendral, Hearsay-II).
  - **Diagnosi:** Si analizzano dati potenzialmente rumorosi per la determinazione di malattie o errori (Mycin, ...).
  - **Monitoring:** I dati si interpretano continuamente per la generazione di allarmi in situazioni critiche. Al sistema è richiesta una risposta in tempo reale soddisfacente (VM).
  - **Planning e Scheduling:** Si determina una sequenza intelligente di azioni per raggiungere un determinato obiettivo (Molgen).
  - **Previsione** (economica, politica ecc.) : Si desidera costruire un sistema in grado di prevedere il futuro in base a un appropriato modello del passato e del presente (Prospector).
  - **Progetto e configurazione:** Il Sistema Esperto deve essere in grado di progettare sistemi partendo da ben determinate specifiche (R1, XCON).

11

## OBIETTIVO DELLE CATEGORIE DI APPLICAZIONE

---

- **Interpretazione:** Inferire descrizioni di situazioni da dati rilevati da sensori.
- **Predizione:** Inferire conseguenze future da una data situazione.
- **Diagnosi:** Inferire malfunzionamenti da osservazioni.
- **Progetto:** Configurare oggetti rispettando vincoli.
- **Pianificazione:** Progettare sequenze di azioni.
- **Monitoring:** Confrontare osservazioni in tempo reale per identificare situazioni di allarme.
- **Debugging:** Prescrivere rimedi per malfunzionamenti.
- **Riparazione:** Eseguire un piano per ottenere il rimedio necessario
- **Insegnamento:** Diagnosi, Debugging e Riparazione del comportamento di uno studente.
- **Controllo:** Interpretare, Predire, Riparare, a Monitorare il comportamento di un sistema.
- Alcune applicazioni ne includono altre.

12

## CLASSIFICAZIONE: SISTEMI DI SINTESI

---

- Si possono classificare i S.E. in base alle operazioni svolte:
  - quelle che costruiscono un sistema (sintesi) e

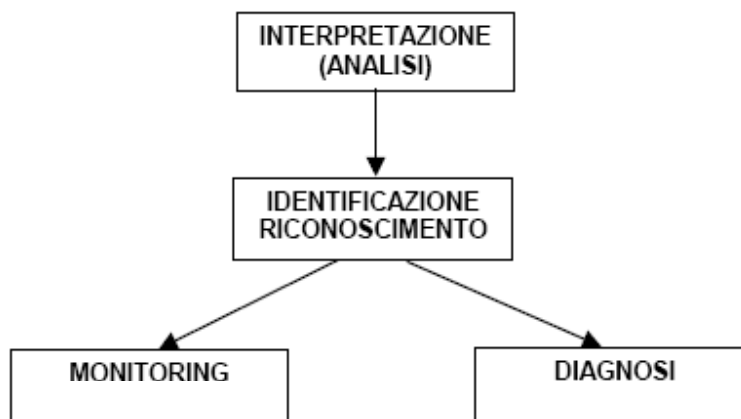


13

## CLASSIFICAZIONE: SISTEMI DI ANALISI

---

- quelle che interpretano un sistema (analisi).



14

## CLASSIFICAZIONE E DIAGNOSI

---

- W. J. Clancey, Heuristic Classification, Artificial Intelligence 27, North Holland, 1985 pp 289-350.
- Il più semplice tipo di classificazione consiste nell'identificare alcuni oggetti sconosciuti o fenomeni come appartenenti a una classe conosciuta di oggetti, eventi o processi.
- Tipicamente queste classi sono tipi organizzati gerarchicamente e il procedimento di identificazione corrisponde al **matching** delle osservazioni di entità sconosciute con caratteristiche note delle classi.

15

## CLASSIFICAZIONE E DIAGNOSI

---

- **Un esempio: Mycin**
  - Sviluppato da E.M. Shortliffe a partire dal 1972;
- **Obiettivi:**
  - decidere se il paziente ha un'infezione che deve essere curata;
  - determinare, se sì, quale è probabilmente l'organismo infettivo;
  - scegliere fra i farmaci adatti per combattere l'infezione quella più appropriata in rapporto alle condizioni del paziente.
  - Mycin risolve il problema di identificare un oggetto sconosciuto dalle culture di laboratorio, mediante un *matching* dei risultati di laboratorio con la gerarchia di batteri.

16



# CARATTERISTICA ESSENZIALE DELLA CLASSIFICAZIONE

---

- Il motore di inferenza seleziona partendo da un insieme di soluzioni pre-enumerate. Quindi **non costruisce** una nuova soluzione.
- Le evidenze (osservazioni) possono essere incerte per cui il sistema può dare come risposta una lista di possibili ipotesi numerate in ordine di plausibilità.
- Esistono anche regole di inferenza oltre al *matching*.
- In molti problemi le caratteristiche di una soluzione non sono date direttamente come dati, ma sono inferite mediante regole di inferenza che esprimono:
  - **Astrazione definizionale:** basata sulle caratteristiche necessarie di un concetto:
    - “Se è un mammifero allora allatta i figli”.
  - **Astrazione qualitativa:** coinvolge dati quantitativi e li confronta con valori normali:
    - “Se il paziente è adulto e il valore dei globuli bianchi è minore di 2500 allora il valore è basso”
  - **Generalizzazione in una gerarchia di sottotipi**
    - “Se il cliente è un giudice, allora è una persona educata”

17

---

## CLASSIFICAZIONE

---

- Classificare un oggetto significa riconoscerlo come appartenente ad una determinata classe.
- Una caratteristica essenziale della classificazione è che **seleziona da un insieme predefinito di soluzioni**.
- Le classi identificano delle regolarità e tutti i membri di una classe le condividono (condizioni necessarie).
- Di solito escludiamo il caso, denominato configurazione, in cui le soluzioni sono un insieme finito, ma molto ampio e quindi determinato dinamicamente come insieme potenza di elementi più semplici (componenti).

18

# CLASSIFICAZIONE SEMPLICE ED EURISTICA

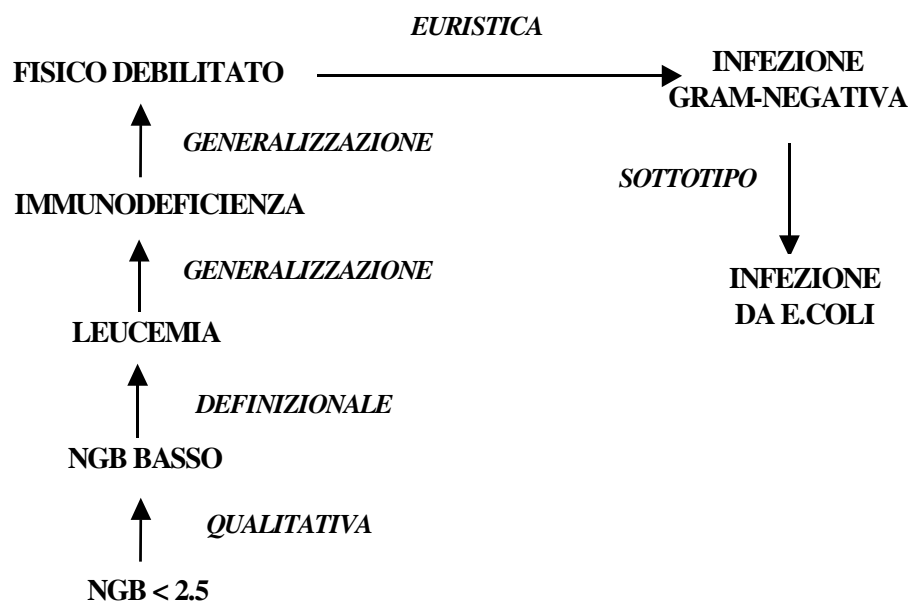
---

- Nella classificazione semplice i dati hanno un **match diretto** con le caratteristiche delle soluzioni (eventualmente dopo un passo di astrazione).
- Nella classificazione euristica le soluzioni possono anche essere trovate usando un **matching euristico** mediante un'associazione diretta e euristica con la gerarchia delle soluzioni.
- Normalmente l'associazione euristica è di tipo empirico.
- In pratica, nella classificazione euristica i dati del problema opportunamente astratti sono associati con classi di soluzioni di problemi.
- **Esempi:** Mycin, Grundy che seleziona i libri che una persona può preferire mediante una classificazione della personalità e poi del libro che può essere più adatto, Sophie che classifica un circuito elettronico in termini dei componenti che causano un malfunzionamento.

19

# CLASSIFICAZIONE SEMPLICE ED EURISTICA

---

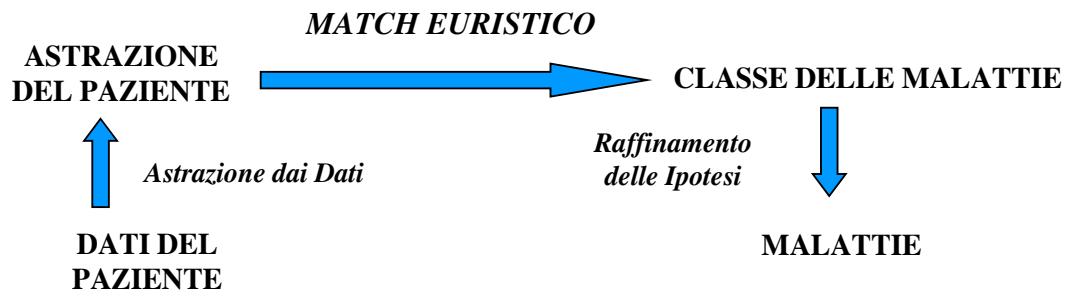


20

# CLASSIFICAZIONE SEMPLICE ED EURISTICA

---

- In pratica, nella classificazione euristica i dati del problema opportunamente astratti sono associati con classi di soluzioni di problemi.



21

---

## FATTORI DI CERTEZZA

---

- Compromesso rispetto a un sistema bayesiano puro.
- Introdotti per la prima volta nel Sistema Esperto Mycin.
- Regole non certe (l'implicazione corretta sarebbe invertita) mediate da un **fattore di certezza**.
- Un fattore di certezza è un numero intero "ad hoc" che varia fra +1 e -1.
- Permette l'inserimento di fatti apparentemente contraddittori, ambedue plausibili con differenti valori di certezza.
- **Fatti:**  
(**<attributo> <entità> <valore> <fattore di certezza>**).
- Esempi di fatti espressi in Mycin sono (sintassi del Lisp):  
(SITE CULTURE-1 BLOOD 1.0)  
(IDENT ORGANISM-2 KLEBSIELLA .25)  
(IDENT ORGANISM-2 E.COLI 0.73)  
(SENSITIVS ORGANISM-1 PENICILLIN -1.0)

22

## REGOLE

---

PREMISE <premessa> ACTION <azione>.

PREMISE

(\$AND

(SAME CNTXT INFECT PRIMARY-BACTEREMIA)

(MEMBF CNTXT SITE STERILESITES)

(SAME CNTXT PORTAL G1))

ACTION

(CNTXT IDENT BACTEROIDES 0.7).

- Significato della regola

IF

(1) the infection is primery-bacteremia,

(2) the site of the culture is one of sterilesites, and

(3)the suspected portal of entry of the organism is the  
gastro-intestinal tract,

THEN there is a suggestive evidence

(0.7) that the identity of the organism is bacteroides.

23

## OSSERVAZIONI

---

- I fattori di certezza iniziali sono forniti dagli esperti.
- Ogni CF in una regola di Mycin rappresenta il contributo della regola al fattore di confidenza di un'ipotesi.
- Rappresenta in un certo senso una probabilità condizionale  $pr(H/E)$ .
- In un sistema Bayesiano puro però si deve assumere che la sola evidenza rilevante per  $H$  sia  $E$ , altrimenti dobbiamo tenere conto delle probabilità congiunte.
- Dunque Mycin assume che tutte le regole siano indipendenti ed è colui che scrive le regole che deve garantire ciò.

24

# R1

---

- Sviluppato all'Università Carnegie-Mellon da John Mc Dermott per conto della Digital a partire dal 1978.
- COMPITO PRINCIPALE: configurare il calcolatore VAX-11/780 automaticamente.
- In base all'ordine del cliente, R1 è in grado di:
  - assicurare che l'ordine sia completo;
  - determinare le relazioni spaziali fra le componenti.
- Un tipico sistema ha più di 100 componenti con varie possibilità di interazione.
  
- Esperti: technical editors
- Nato con un nucleo di 250 regole, nella versione finale ne possiede circa 2800.
- Dal 1980, è un prodotto funzionante Digital.
- R1 è implementato in OPS-5.

25

---

## TIPI DI CONOSCENZA IN R1

---

- a) **Informazione sui componenti:**
  - in memoria di massa sono raccolte informazioni su circa 400 componenti della Digital che il Sistema va a recuperare quando necessario.

```
(RK711-EA
!class bundle
!type disk drive
!supported yes
!component-list 1 070-12292-25
    1 RK07-EA*
    1 RK611)
```
- b) **Conoscenza sui vincoli:**
  - è la conoscenza di come associare determinati componenti per formare configurazioni parziali corrette e di come associare fra di loro le configurazioni parziali

26

## TIPI DI CONOSCENZA IN R1

---

- b) **Conoscenza sui vincoli:**

- Distributed-mb-devices-3**

- IF: current active context is distributing massbus devices

- AND there is a single port disk drive

- that has not been assigned to a massbus

- AND there are no unassigned dual port disk drives

- AND .....

- THEN:assign the disk drive to the massbus.

- c) **MEMORIA DI LAVORO:**

- tiene traccia della conoscenza che viene accumulata dinamicamente durante il processo di configurazione.

27

## CONTROLLO IN R1

---

- Forward
- Strategia Irrevocabile
- Tasks non interagenti → sistema di produzione decomponibile
- Nessuna strategia di controllo esplicita
- Ambiente povero
- Regole per il cambio di contesto
- **Check-voltage-and-frequency-1**

- IF: the MOST CURRENT ACTIVE CONTEXT is checking voltage and frequency

- AND there is a component that requires one voltage or frequency

- AND there is another component that requires a different voltage or frequency

- THEN:ENTER THE CONTEXT of fixing voltage or frequency mismatches.

28

# S.E. SVILUPPATI DAL GRUPPO DI IA

## Univ. Ferrara e Univ. Bologna

---

- Sistemi **utilizzabili** (almeno allo stato prototipale) nelle Aree:  
Progetto, Monitoring, Diagnosi, Scheduling.
- **ADES** (ATP Design Expert System) per il **progetto** dei sistemi per il controllo delle stazioni ferroviarie (SASIB);
- **SMA** (Station Master Assistant) per il **monitoring** e la **pre-diagnosi** degli enti della stazione al fine di determinare la fattibilità degli itinerari (SASIB);
- **TSA** (Train Scheduling Assistant) per regolare il traffico dei treni all'interno di una stazione di grosse dimensioni (SASIB).
- **FUN** (Function Point Measurement) per il calcolo dei Function Point per un sistema software.
- Scheduling delle lavorazioni giornaliere (Officine Ortopediche Rizzoli, linguaggio di CLP).
- Identificazione di difetti in semilavorati meccanici (BERCO S.p.A, approccio mediante apprendimento automatico di regole).
- Diagnosi, verifica degli esami medico-clinici, interpretazione dei dati (DIANOEMA SpA ora Noemalife , S.Orsola-Malpighi Bologna).

29

---

## PROSPETTIVE

---

- Verso gruppi di agenti intelligenti cooperanti (ingegneria del software).
- Cooperazione con un essere umano
- interfacce grafiche.
- Integrazione con vari sistemi informatici (interfacce grafiche, database ecc..), tecnologie tradizionali e non (reti neurali, sistemi fuzzy)
- sistemi ibridi.
- Da costose workstations special-purpose (LISP-Machine) verso sistemi Unix o PC con l'utilizzo di Tools scritti in linguaggi più tradizionali (C, C++).
- Sistemi basati sulla conoscenza in aree più "semplici": costruzione di libri e manuali non più passivi ma **attivi**.
- In fondo, i linguaggi Object-Oriented sono nati nel 1967 e i Database Relazionali negli anni 70.....

30