SISTEMI ESPERTI

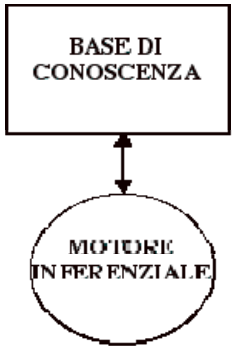
- **Sistemi Basati Sulla Conoscenza (1980)**
- Un sistema **basato sulla conoscenza** è un sistema in grado di risolvere problemi in un **dominio limitato** ma con prestazioni **simili** a quelle di un **esperto** umano del dominio stesso.
- Generalmente esamina un largo numero di possibilità e costruisce 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)*

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.*

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?



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/pattern/LHS> then <conclusione/azione/RHS>

  L'azione può modificare la memoria di lavoro o produrre effetti collaterali

STRATEGIE DI CONTROLLO

- Tipicamente 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.
```


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.
```

Matching efficiente: l'algoritmo RETE

- Sviluppato da Charles Forgy nel 1979 (implementato in OPS);
- Presente in quasi tutti i sistemi a regole ***forward*** di tipo commerciale
- Sistemi esperti reali: migliaia di regole, problemi di efficienza;
- Aumentare l'efficienza della parte di *pattern matching* evitando di fare il test sequenzialmente regola per regola;
- Crea un albero decisionale in cui ogni nodo corrisponde ad un *pattern* nella parte sinistra della regola (LHS);
- Ogni nodo ricorda i fatti che soddisfano la regola;
- LHS complete sono definite come strade dalla radice alla foglia;

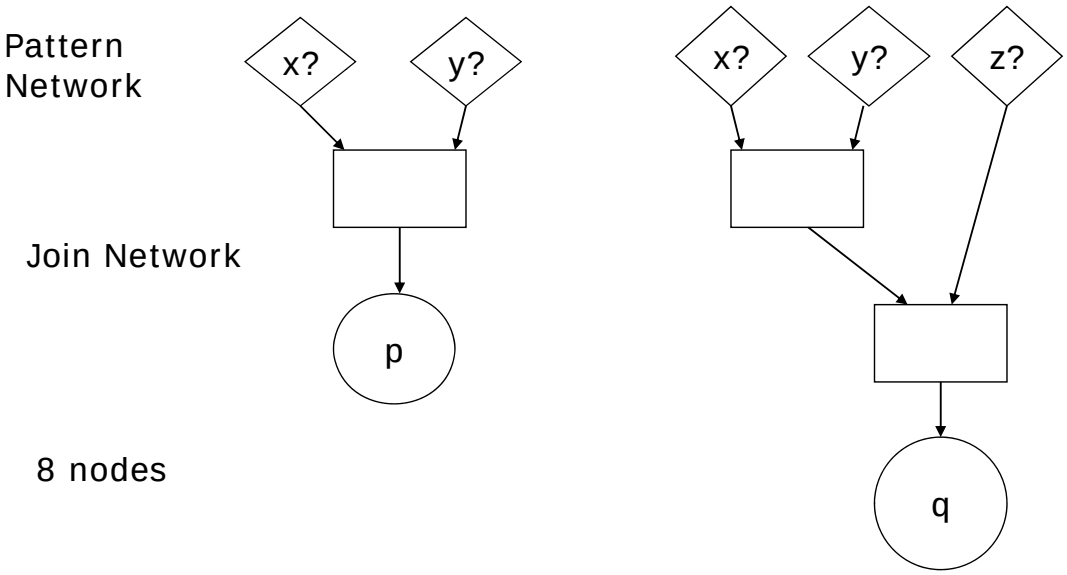
- Risparmia tempo, ma … consuma memoria

Rete Algorithm

- Unlike common implementation of a production system where each rule is checked against the known facts in the Knowledge base before executing (firing) that rule and resetting to the top of the rule stack after execution, the Rete-based system builds a network of nodes, where each node (except the *root*) corresponds to a *pattern* occurring in the left-hand-side of a rule (rooted Direct Acyclic Graph)
- The left-hand-side of a rule is a path from the *root* node to a leaf node. Each node has a memory of facts against the pattern. Traversing through the *root* and leaf node, as the combination of facts relates exactly to the pattern the corresponding rule is triggered
- The Rete algorithm uses more memory than other common binary algorithms used in the expert systems and Artificial Intelligence implementations but is more efficient and fast
- Some of the popular expert systems such as CLIPS, Jess, and Soar follow Rete algorithm (dated 2007)

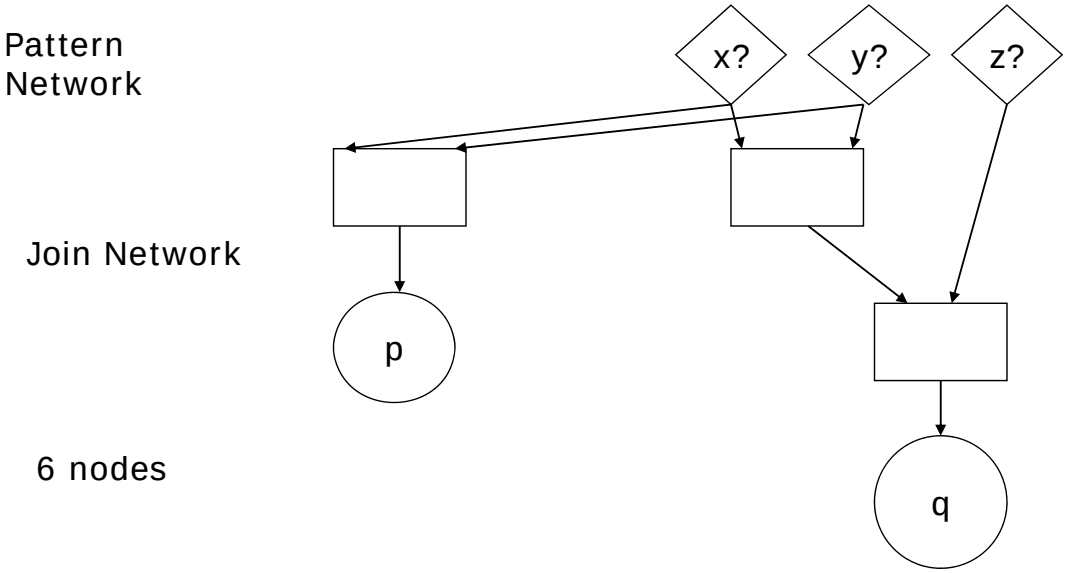
Esempio1:

Rules: IF x & y THEN p
IF x & y & z THEN q



Rete esempio (3)

Rules: IF x & y THEN p
IF x & y & z THEN q



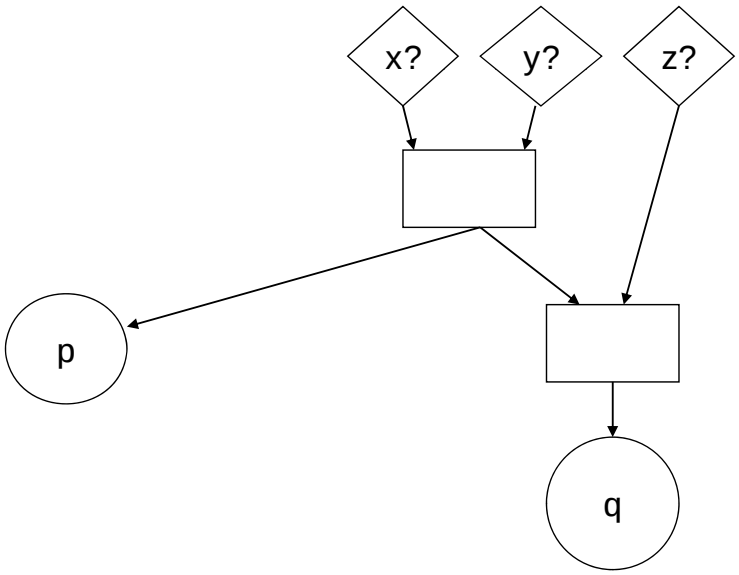
Rules: IF x & y THEN p
IF x & y & z THEN q

Rete esempio (3)

Pattern Network

Join Network

5 nodes

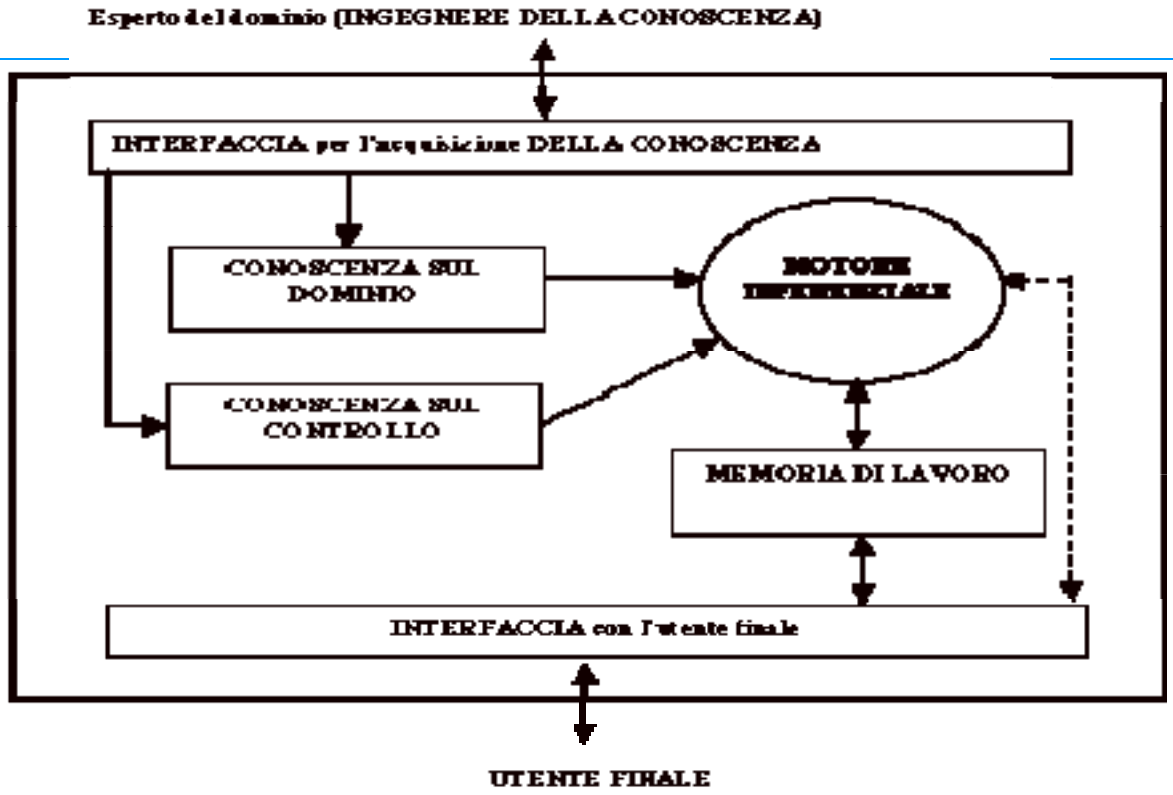


Conflict Resolution

- Fase di decisione fra regole applicabili (pattern matching) e quelle che si applicherà effettivamente (fire)
- Tutte le regole che hanno le condizioni soddisfatte sono inserite nell' AGENDA (conflict set)

- Conflict resolution seleziona quale regola applicare
- I Tools per Sistemi Esperti offrono diverse strategie per la conflict resolution
  - La più specifica;
  - La meno specifica;
  - La più prioritaria;
  - La più recente (l'ultima diventata attiva, recente modifica dell'item nella memoria di lavoro..
  - Etc.
  - Inoltre nessuna regola già scattata dovrebbe essere selezionata nuovamente con gli stessi *item* della memoria di lavoro (*loop avoiding*)

ARCHITETTURA - KB SYSTEM



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 Abduttivo....

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.

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-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++, Java, etc).

Tools per sistemi a regole:

Esempi commerciali:

- ***JRules***
  - JRules, prodotto dalla ILOG www.ilog.com, (commerciale) è uno strumento per costruire sistemi esperti che combina tecniche di inferenza basate sulle regole di produzione e programmazione ad oggetti

- ***Jess***
  - Java Expert System Shell http://herzberg.ca.sandia.gov/jess/ è nato nel 1995 come clone Java di CLIPS (un altro tool) per poi differenziarsi grazie a numerose nuove funzionalità. Utilizza l'algoritmo RETE per il forward chaining, ma è in grado di lavorare anche in backward-chaining in modo efficiente;
  - La sintassi delle regole è macchinosa (LISP).

Tools per sistemi a regole (Open Source)

- ***CLIPS***
  - CLIPS è un tool di pubblico dominio scritto in linguaggio C ed è stato usato come base di partenza per lo sviluppo di molti altri tools di sistemi esperti, tra cui Jess e Drools ed è supportato anche da Protégé (editor di ontologie). http://clipsrules.sourceforge.net/
- ***Algernon***
  - è implementato in Java e si interfaccia con Protégé. Algernon esegue sia il forward sia il backward chaining su basi di conoscenza frame-based.
  - La base di conoscenza può essere gestita con Protégé che supporta i linguaggi Schema, RDF, OWL ed altri. http://algernon-j.sourceforge.net/
- ***JEOPS***
  - JEOPS (Java Embedded Object Production System) è un tool open-source pensato per il Java Developer e permette essenzialmente di implementare la parte di regole a partire da un file di configurazione. Non si tratta di una libreria da utilizzare per le proprie applicazion,i ma di un vero e proprio precompilatore che genera i sorgenti java necessari a realizzare la logica richiesta con l'algoritmo RETE (forward-chaining). http://sourceforge.net/projects/jeops/
- ***MANDARAX***
  - è una libreria open-source di classi java per regole di deduzione. Fornisce una infrastruttura per definire, gestire e interrogare una base di regole.
  - implementa un algoritmo in backward chaining, simile a Prolog.
  - partecipa attivamente allo sviluppo di RuleML e permette di usare tale formato per salvare e caricare la knowledge-base in modo nativo. http://www.mandarax.org

DROOLS

- Drools (http://www.jboss.org/drools) è una suite modulare per lo sviluppo e la gestione di sistemi avanzati basati su regole.
- E' open source e basato su Java; è in continuo sviluppo, anche grazie ad una numerosa comunità di contributors, ed è attualmente alla versione 5.x

- Il suo rule engine è basato sull'algoritmo RETE (forward-chaining), mentre le regole sono scritte usando un linguaggio proprietario.
- Al pari di sistemi commerciali della stessa classe (es. ILOG) fornisce supporto per eventi (KB con vincoli temporali) e workflow (flussi di esecuzione).
- Lo vedremo in laboratorio.
- Sono disponibili approfondimenti per tesine/laboratorio, tesi (progetto con Osp. S.Anna)

APPLICAZIONI

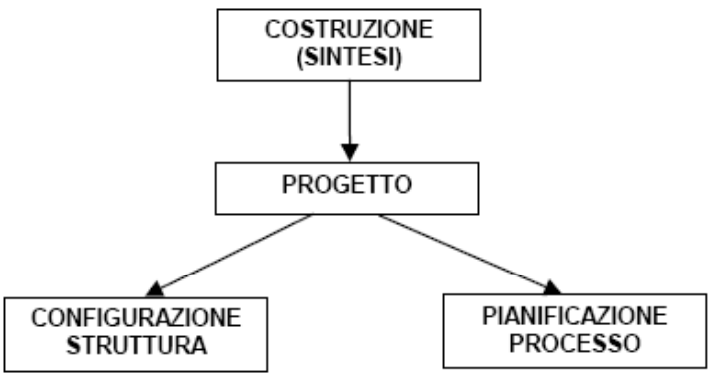
- **Migliaia** di Sistemi Esperti nei settori più svariati.
  - **Interpretazione**: Si analizzano dati complessi e potenzialmente rumorosi per la determinazione del loro significato (Dendral, Hearsay-II).
  - **Diagnosi**: Si analizzano dati potenzialmente rumorosi per la determinazione di malattie 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).

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.

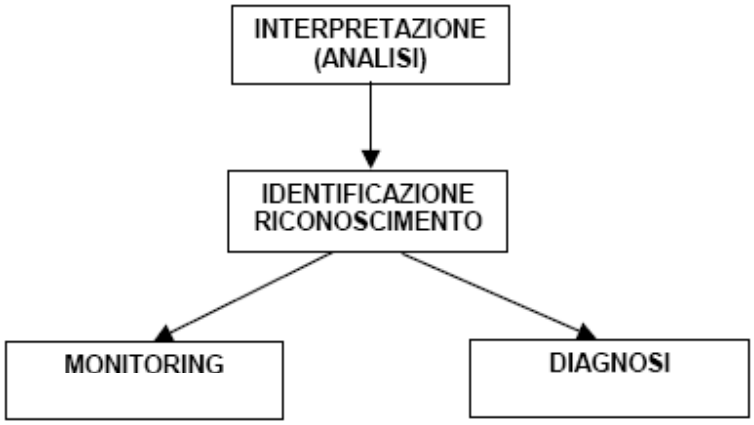
CLASSIFICAZIONE: SISTEMI DI SINTESI

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



CLASSIFICAZIONE: SISTEMI DI ANALISI

- quelle che interpretano un sistema (analisi).



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.

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 le medicine adatte 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.

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"

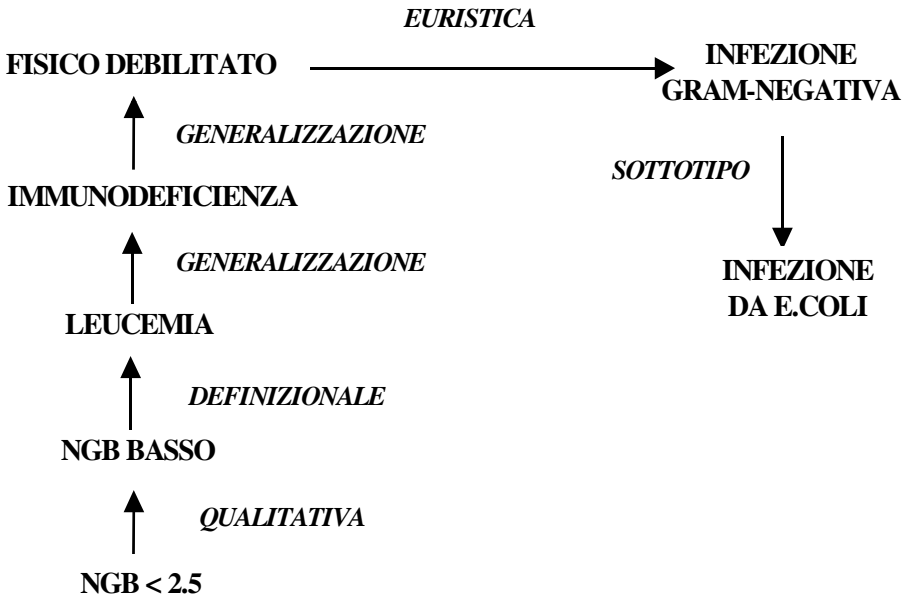
CLASSIFICAZIONE

- 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).

CLASSIFICAZIONE SEMPLICE ED EURISTICA

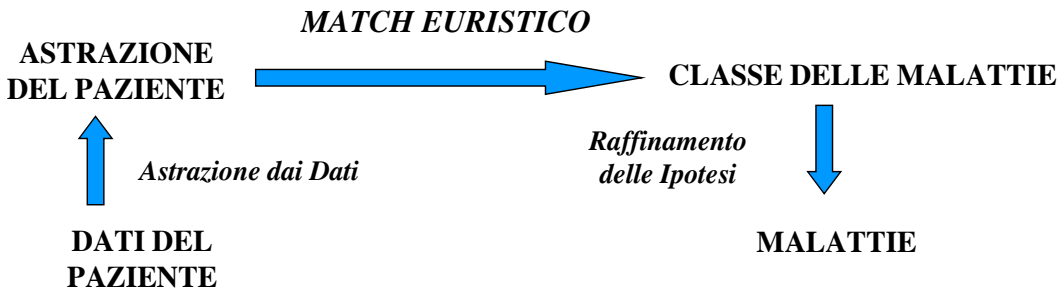
- Nella classificazione semplice i dati hanno un ***match diretto*** con le caratteristiche delle soluzioni (eventualmente dopo un passo di astrazione).
- Nella classificazione euristica le soluzioni possono anche essere trovate usando un ***matching euristico*** mediante un'associazione diretta e euristica 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.

CLASSIFICAZIONE SEMPLICE ED EURISTICA



CLASSIFICAZIONE SEMPLICE ED EURISTICA

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



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)
```


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.
```

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.

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 ora ne possiede circa 2800.
- Dal 1980 è un prodotto funzionante Digital.
- R1 è implementato in OPS-5.

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

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.

CONTROLLO IN R1

- Forward
- Strategia Irrevocabile
- Tasks non interagenti → sistema di produzione devomponibile
- 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.

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 Mesaurement) per il calcolo dei Function Point per un sistema software.
- Identificazione di difetti in semilavorati meccanici (BERCO S.p.A, approccio mediante apprendimento automatico di regole).
- Sistema Esperto per scelta colore (COROB S.P.A.) Progetto UE

Sistemi Esperti in campo medico

- Diagnosi, verifica degli esami medico-clinici, interpretazione dei dati (Noemalife SpA, S.Orsola-Malpighi Bologna). In particolare:
  - DNSEV (Expert System for clinical result Validation), per migliorare la qualità del processo di validazione eseguito dai laboratori di analisi biochimica.

    ESMIS (Expert System for Microbiological Infection Surveillance), per migliorare la qualità del processo di validazione eseguito dai laboratori di analisi microbiologica e per monitorare gli eventi infettivi all'interno di un ospedale.

    DNTAO (Expert System for supporting the Oral Anticoagulation Treatment) per il supporto ai medici per le prescrizioni e visite per la Terapia Anticoagulante Orale.
- Definizione di linee guida in campo medico (SPRING) e formalizzazione di protocolli di screening