Il Ragionamento

TIPI DI RAGIONAMENTO

- Logica non monotona: permette di cancellare e aggiungere enunciati alla base dati. In questa logica la credenza in un enunciato si può basare sulla mancata credenza in qualche altro enunciato (ragionamento by default).

- Ragionamento probabilistico: rende possibile rappresentare inferenze probabili ma incerte.

- Logica sfumata (fuzzy logic): per mette di rappresentare proprietà di oggetti continue (non binarie) o sfumate.

- Concetto di spazi di credenza: permette di rappresentare modelli di insiemi di credenze inserite l'uno nell'altro.

RAGIONAMENTO NON-MONOTONO

MOTIVAZIONI

Spesso la conoscenza sul mondo di cui si dispone è incompleta.

È tuttavia necessario sapere derivare conclusioni plausibili dalla conoscenza data, per poter prendere decisioni, fare piani, ecc.

Per derivare conclusioni plausibili è necessario fare delle assunzioni.

La scelta delle assunzioni non è cieca: gran parte della conoscenza sul mondo è data sotto forma di regole generali che specificano proprietà tipiche degli oggetti (che valgono per default). Ad esempio:

gli uccelli volano

che significa: gli uccelli *tipicamente* volano.

Ci sono però eccezioni: i pinguini, gli struzzi, ecc.

Il ragionamento non-monotono si occupa di come conclusioni plausibili (ma non infallibili) possono essere derivate da una base di conoscenza (ossia da un insieme di formule).

Siccome le conclusioni derivate sono un tentativo, esse possono dover essere ritrattate quando nuove informazioni vengono aggiunte alla base di conoscenza.

Esempio:

KB
- Gli uccelli tipicamente volano
- I pinguini non volano
- Tweety è un uccello

È plausibile concludere che Tweety vola poiché non c'è nessuna informazione contraria.

Se però viene aggiunta l'informazione

Tweety è un pinguino

si sa che Tweety non vola: non è più plausibile concludere che Tweety vola.

L'affermazione:

Tipicamente vale A

deve dunque essere letta:

In assenza di informazioni contrarie, si assume A.

Il problema sta nel definire in modo preciso cosa significa "in assenza di informazioni contrarie".

Il significato può essere: "non c'è nulla nella KB che sia inconsistente con l'assunzione A".

Altre interpretazioni sono possibili.

Interpretazioni diverse danno luogo a logiche non-monotone diverse.

ALTRE MOTIVAZIONI: CWA

Nella teoria dei Databases si adotta la convenzione che le informazioni negative non vengono rappresentate esplicitamente.

Se un fatto positivo non è esplicitamente presente nel DB, si assume che valga la sua negazione (Closed World Assumption, CWA).

Esempio.

Supponiamo che un sistema di prenotazioni aeree contenga fatti del tipo:

$$\text{collega}(\text{volo}, \text{città}_1, \text{città}_2)$$

Se il sistema non riesce a dimostrare

$$\text{collega}(\text{AZ113}, \text{Roma}, \text{Parigi})$$

allora il sistema può concludere

$$\neg\text{collega}(\text{AZ113}, \text{Roma}, \text{Parigi})$$

La CWA è un tipo di ragionamento non-monotono:

"Se DB $\nvdash \Phi$, allora DB $\vdash \neg\Phi$ mediante CWA".

INADEGUATEZZA DELLA LOGICA CLASSICA

Non va bene rappresentare una regola come "Gli uccelli tipicamente volano" nella logica classica, come segue:

$$\forall x(\text{uccello}(x) \wedge \neg\text{eccezione}(x) \rightarrow \text{vola}(x))$$

$$\forall x(\text{eccezione}(x) \leftrightarrow \text{pinguino}(x) \vee \text{struzzo}(x) \vee \text{senzaali}(x) \vee \ldots)$$

Questa rappresentazione richiede una lista completa di tutte le eccezioni: IMPOSSIBILE!!

Per poter derivare che un uccello particolare, Tweety, vola è necessario provare che non ci sono eccezioni, ossia

$$\neg \text{pinguino}(\text{Tweety})$$

$$\neg \text{struzzo}(\text{Tweety})$$

……

Al contrario, noi vorremmo concludere che Tweety vola perché *non si può dimostrare che è un'eccezione*, non perché *si può mostrare che non è un'eccezione*.

La logica classica è monotona:

se una formula q è derivabile da un'insieme di formule P, allora q è anche derivabile da un qualunque soprainsieme di P.

$$P \vdash q \quad \Rightarrow \quad P \cup \{\alpha\} \vdash q$$

Come si è visto, una regola del tipo

Se x è un uccello ed è inconsistente assumere che x vola, allora si assume che x vola

è non-monotona

se so che Tweety è un uccello, posso assumere Tweety vola

ma, se vengo a conoscenza del fatto che Tweety è un pinguino, la deduzione Tweety vola non è più valida.

IL FRAME PROBLEM

Problema di rappresentazione di mondi dinamici.

Come rappresentare gli aspetti del mondo che rimangono invariati rispetto a certi cambiamenti di stato.

Ad esempio, spostare un oggetto non ne cambia il colore.

In una rappresentazione in logica classica occorre descrivere esplicitamente tutto ciò che non cambia.

Occorre un numero molto elevato di frame axioms:

$$\forall x, c, s, l \, (color(x, e, s) \rightarrow color(x, c, result(move\_on, l, s)))$$

Un modo di generalizzare potrebbe essere:

$$\forall propr, e, s\ (holds(propr, s) \wedge \neg eccezione(propr, e, s) \rightarrow holds(propr, result(e, s)))$$

Anche in questo caso c'è bisogno di una forma di ragionamento che permetta di trattare le eccezioni.

LOGICHE NON-MONOTONE

"Non Monotonic Logic" Mc Dermott e Doyle, '80
"Default Logic" Reiter, '80
"Circoscrizioni" Mc Charty, '80
"Autoepistemic Logic" Moore, '84

Articolo di rassegna:

"Non Monotonic Reasoning" Reiter, in Annual Review in Computer Science, 1987, 2:147-186

DEFAULT LOGIC [Reiter 1980]

Permette regole di inferenza non standard per esprimere le proprietà che valgono per default.

Ad esempio

$$\frac{\text{uccello}(x) : \text{vola}(x)}{\text{vola}(x)}$$

che si può leggere:

"se x è un uccello e si può assumere in modo consistente che x voli, allora possiamo inferire che x vola".

In generale sono permesse default rules del tipo:

$$\frac{\alpha(x) : \beta(x)}{\gamma(x)}$$

dove:

$\alpha(x)$: prerequisito

$\beta(x)$: giustificazione

$\gamma(x)$: conseguente

che si legge:

"se $\alpha(x)$ vale e $\beta(x)$ può essere assunto in modo consistente, allora possiamo concludere $\gamma(x)$".

Date un insieme di formule (nella logica classica del I ordine), ossia una KB generalmente incompleta, le default rules permettono di completare (estendere) la KB $\Rightarrow$ ESTENSIONI.

DEFAULT THEORIES

Una default theory è una coppia <D, W> dove
D è un insieme di default rules e
W è un insieme di formule del I ordine

Esempio:

$$D = \left\{ \frac{uccello(x) : vola(x)}{vola(x)} \right\}$$

$$W = \{uccello(Tweety, \forall x(pinguino(x) \rightarrow \neg vola(x)\}$$

Siccome uccello(Tweety), è vero e posso assumere consistentemente vola(Tweety); vola(Tweety) è vero in una estensione (l'unica) della teoria.

Se aggiungo in W pinguino(Tweety), da W posso derivare ¬vola(Tweety). Questo "blocca" l'applicazione della default rule: non è consistente assumere vola(Tweety).

Esempio (Nixon Example):

$$D = \left\{ \frac{Rep(x) : \neg Pac(x)}{\neg Pac(x)}, \frac{Quak(x) : Pac(x)}{Pac(x)} \right\}$$

I repubblicani sono tipicamente non-pacifisti, mentre i quaccheri sono tipicamente pacifisti.

W = {Rep(Nixon), Quak(Nixon)}

Nixon è quacchero e repubblicano.

Cosa si può concludere sul pacifismo o non di Nixon?

Entrambe le default rules hanno il prerequisito derivabile da W:

a) se applico la I e concludo ¬Pac(Nixon), non posso più applicare la II (Pac(Nixon) non può essere assunto consistentemente)

b) se applico la II e concludo Pac(Nixon) la I regola non è più applicabile.

$\Rightarrow$ ci sono 2 ESTENSIONI:
una in cui è vero ¬Pac(Nixon)
l'altra in cui è vero Pac(Nixon).

TMS

Il ragionamento non monotono (per default) è basato sull'assunto:

- se X non è noto, allora concludere Y

o, più precisamente

- se X non può essere provato, allora concludere Y.

Poiché la logica dei predicati è indecidibile, si pone più correttamente:

- se X non può essere provato entro un determinato tempo, concludere Y.

(questi sistemi non possono essere caratterizzati in modo formale!)

Nel sistema non monotono gli enunciati possono essere cancellati o aggiunti alla base dati.

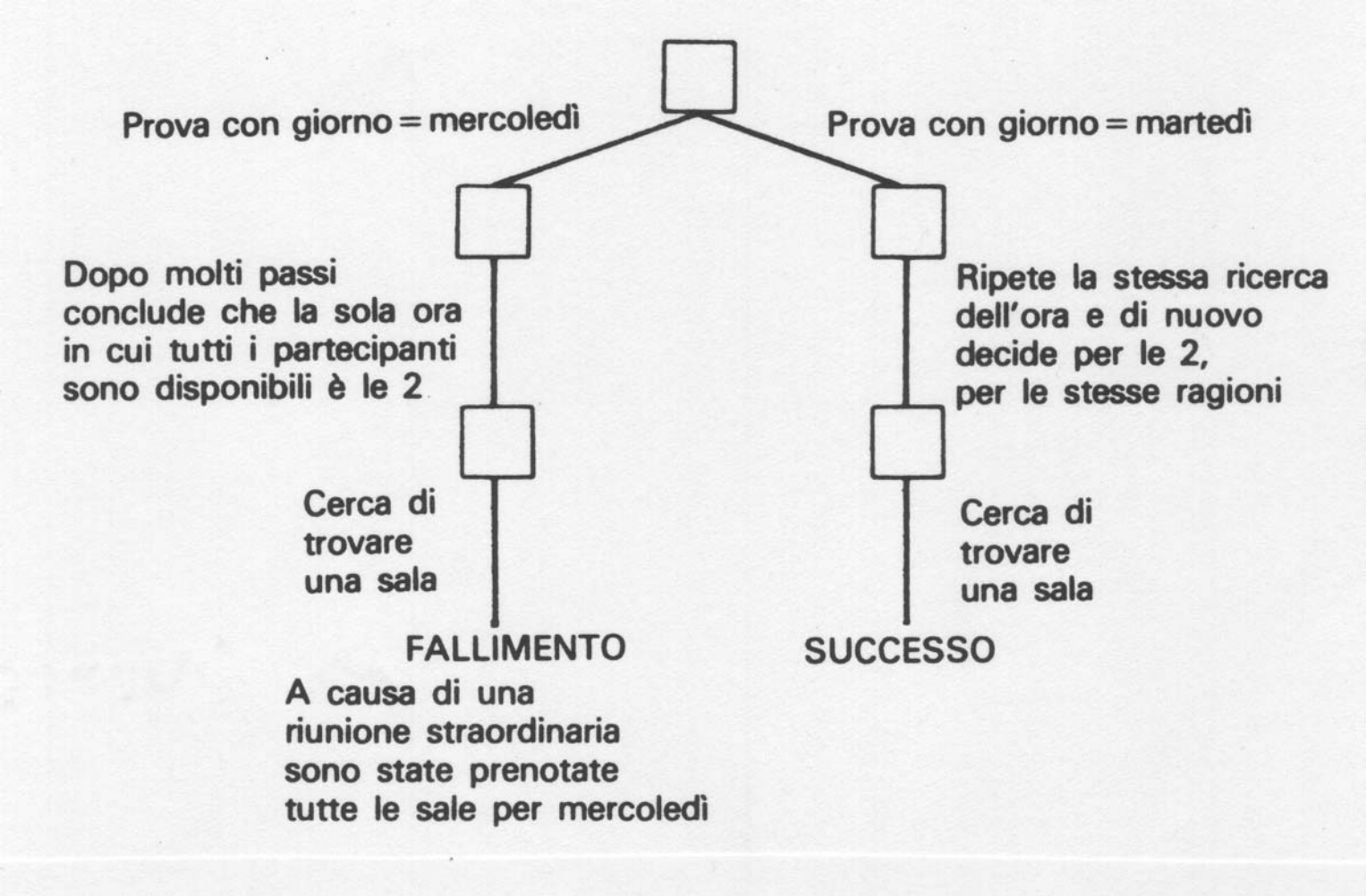
Se si cancella un enunciato, può succedere di dover cancellare tutti gli enunciati la cui prova è dipesa dal primo.

Anche se la base di conoscenza è completa, durante la soluzione di un problema si può arrivare ad un sistema non monotono.

Esempio: trovare un periodo di tempo in cui tre persone occupate possono partecipare ad una riunione.

Si parte col provare se va bene un particolare giorno, per es. mercoledì, e si controllano le incompatibilità. Se si rileva una incompatibilità, si elimina l'assunzione iniziale, più tutte quelle inferite da questa.

Si ottiene un albero di ricerca orientato con ritorno all'indietro:



Si osservi che si è dovuta abbandonare anche l'assunzione sull'ora, le 2, che invece aveva una validità intrinseca (qui veniva dopo la scelta del mercoledì!).

L'eliminazione degli enunciati dovrebbe essere legata solo alla responsabilità di incoerenza, non all'ordine temporale di generazione.

Occorre un processo di ritorno all'indietro guidato dalla dipendenza.

Sistema di ragionamento non monotono TMS (Truth Maintenance System).

Ogni enunciato viene chiamato nodo, e si trova in uno dei due stati:

IN è creduto vero

OUT non è creduto vero (non vi sono ragioni per crederlo vero o nessuna ragione da cui dipende è attualmente valida).

Ad ogni nodo è associata una lista di giustificazioni.

Nodi IN: hanno almeno una giustificazione valida.

Nodi OUT: non hanno alcuna giustificazione valida.

NB: il sistema è dinamico, ma non si hanno "cancellazioni" reali, ma solo siglature (OUT): inferenze già effettuate e poi non ritenute valide possono ridiventare valide in tempi successivi.

Per poter modellare situazioni differenti, le giustificazioni possono essere costituite da:

Liste di supporto, nella forma

(SL (nodi_in) (nodi_out))

Prove condizionali, nella forma

(CP conseguente (ipotesi_in) (ipotesi_out))

Liste di supporto

Le giustificazioni basate su SL sono valide se tutti i nodi elencati nella lista (nodi_in) sono IN e tutti quelli in (nodi_out) sono OUT (attualmente!).

Esempio:

1. È inverno (SL () ())
2. Fa freddo (SL (1) ())

La 1) non dipende da da altri nodi: è una *premessa*.

La 2) dipende dal nodo 1): se quest'ultimo diventa out, non si può più ritenere che fa freddo (in quanto non possiede più un supporto ben fondato).

Se la lista (nodi_out) non è vuota, TMS tratta il ragionamento per default.

Esempio:

1. È inverno (SL () ())
2. Fa freddo (SL (1) (3))
3. Fa caldo (attualmente non è ritenuto valido)

In pratica: se è inverno, e non c'è evidenza che faccia caldo, allora assumi che fa freddo.

Nomenclatura: se la lista (nodi_out) in SL non è vuoto, il nodo è detto *assunzione*.

Nota: questa struttura impone di mantenere anche i nodi OUT: se, ad esempio, si togliesse il nodo 3), si dovrebbe eliminare anche il nodo 2)!

Prove condizionali

Permettono di rappresentare argomentazioni ipotetiche, valide se il nodo conseguente è IN ogni volta che sono IN i nodi in (ipotesi_in) e sono OUT i nodi in (ipotesi_out).

TMS le trasforma in giustificazioni SL (di più facile manipolazione).

NB: TMS non crea le giustificazioni, ma serve solo a mantenere una base di credenze coerente. È quindi un supporto per un sistema per la soluzione di problemi (ragionamento).

Esempio di uso.

Problema della scelta della data di un incontro.

Si parte con l'ipotizzare mercoledì come giorno valido per il meeting.

1. Giorno (M) = mercoledì (SL () (2))
2. Giorno (M) ≠ mercoledì (non c'è niente che supporti questa convinzione, quindi è OUT).

(Nota: il nodo 1) è attualmente IN, perché 2) è OUT).

Una serie di altre verifiche (nodi 57, 103, 45) portano a concludere che un'ora valida per l'incontro è le due pomeridiane.

La base di conoscenza diventa:

1. Giorno (M) = mercoledì (SL () (2))
2. Giorno (M) ≠ mercoledì
3. Ora (M) = 14.00 (SL (57, 103, 45) ())

Il sistema cerca una sala per la riunione, ma non la trova. Lo dichiara a TMS creando un nodo

4. Contraddizione (SL (1, 3) ())

Parte la procedura di ritorno all'indietro guidata dalla dipendenza. A partire dalla lista SL di 4), cerca l'insieme minimo di assunzioni che, se eliminate, farebbero sparire la contraddizione (il procedimento è basato sui rimandi, ed è quindi ricorsivo).

Nel nostro caso, è sufficiente eliminare il nodo 1): si memorizza il risultato dell'operazione di ritorno all'indietro aggiungendo un nodo che conglobi l'insieme di condizioni contraddittorie:

5. Nogood N1 (CP 4 (1, 3) ())

A questo punto si rende OUT il nodo 1), rendendo IN uno dei nodi della sua lista nodi_out (qui c'è solo il nodo 2: ce n'è sempre almeno uno, perché è un'assunzione!).

Si attiva il nodo 2) ponendogli accanto la giustificazione. La base di conoscenza diventa:

1. Giorno (M) = mercoledì (SL () (2))
2. Giorno (M) ≠ mercoledì (SL (5) ())
3. Ora (M) = 14.00 (SL (57, 103, 45) ())
4. Contraddizione (SL (1, 3) ())
5. Nogood N1 (CP 4 (1, 3) ())

Nota: si osservi che, poiché 5) è IN, 2) è diventata IN, da questo 1) è OUT, e quindi anche 4) è OUT, risolvendo la contraddizione.

Il tutto senza toccare il nodo 3), che rimane IN (la scelta dell'ora rimane valida).

Si procede quindi a scegliere un altro giorno.

Si osservi che il nodo 5) è controllato da una giustificazione CP: la derivazione logica resta valida, indipendentemente dal fatto che i nodi implicati siano attualmente IN o OUT (in pratica, correla logicamente lo stato del nodo conseguente, il 4), con lo stato di altri nodi posti in due liste, ipotesi_in e ipotesi_out).

Se si fosse posto

5. Nogood N1 (SL (4) ())

L'effetto sarebbe stato differente, facendo salva la dipendenza di 5) da 4) e quindi anche da 1) e 3): se infatti il nodo 4) diventa OUT, anche 5) diventa OUT e questo renderebbe OUT il nodo 2) e di conseguenza il nodo 1) ridiventerebbe IN. Si avrebbe una sorta di *ragionamento circolare* da cui non si saprebbe come uscire.

La prova condizionale evita questo fatto.

Spiegazione:

1. Giorno (M) = mercoledì (SL () (2))
2. Giorno (M) ≠ mercoledì (SL (5) ()) IN
3. Ora (M) = 14.00 (SL (57, 103, 45) ())
4. Contraddizione (SL (1, 3) ())
5. Nogood N1 (SL (4) ())

Spiegazione:

1. Giorno (M) = mercoledì (SL () (2)) OUT
2. Giorno (M) ≠ mercoledì (SL (5) ()) IN
3. Ora (M) = 14.00 (SL (57, 103, 45) ())
4. Contraddizione (SL (1, 3) ())
5. Nogood N1 (SL (4) ())

Spiegazione:

1. Giorno (M) = mercoledì (SL () (2)) OUT
2. Giorno (M) ≠ mercoledì (SL (5) ()) IN
3. Ora (M) = 14.00 (SL (57, 103, 45) ())
4. Contraddizione (SL (1, 3) ()) OUT
5. Nogood N1 (SL (4) ())

Spiegazione:

1. Giorno (M) = mercoledì (SL () (2)) OUT
2. Giorno (M) ≠ mercoledì (SL (5) ()) IN
3. Ora (M) = 14.00 (SL (57, 103, 45) ())
4. Contraddizione (SL (1, 3) ()) OUT
5. Nogood N1 (SL (4) ()) OUT

Spiegazione:

1. Giorno (M) = mercoledì (SL () (2)) OUT
2. Giorno (M) ≠ mercoledì (SL (5) ()) ~~IN~~ OUT
3. Ora (M) = 14.00 (SL (57, 103, 45) ())
4. Contraddizione (SL (1, 3) ()) OUT
5. Nogood N1 (SL (4) ()) OUT

RAGIONAMENTO PROBABILISTICO

Motivazioni:

- Il mondo rilevante è realmente casuale: per esempio, il movimento degli elettroni di un atomo o la distribuzione della popolazione che si ammalerà durante una epidemia

- Dato un numero sufficiente di informazioni il mondo rilevante non è casuale, ma il nostro programma non sempre avrà accesso a tutti i dati necessari; per esempio la probabilità di successo che ha un farmaco nel combattere la malattia di un determinato paziente.

- Il mondo sembra casuale perché non lo abbiamo descritto al livello giusto. Esempio: per rappresentare i caratteri si usano matrici di pixel, ma sembra più opportuno utilizzare *strokes*.

Considerazione metodologica:

nei primi due casi l'approccio probabilistico è corretto; nel terzo caso è meglio salire di livello nelle primitive utilizzate (per es. archi e linee) per avere una descrizione "più adeguata".

Si utilizza la teoria della probabilità per correlare eventi che dipendono da altri eventi (a cui è associata una evidenza statistica). Esempi:

- Probabilità congiunta:
  Se A e B sono indipendenti:
  $\text{Prob}(A \wedge B) = \text{prob}(A) * \text{prob}(B)$

- Valutazione probabilistica:

  $$\text{Punteggio} = \sum_{i=1}^{N} prob(i) * \text{valore(i)}$$

  dove N è il numero dei possibili eventi

Esempio:

Valutare la mossa di un avversario, tenendo conto del punteggio di una mossa e delle probabilità che quella mossa venga effettuata.

- Teorema di Bayes: permette di calcolare la probabilità di un certo evento a partire da un insieme di osservazioni effettuate.

Siano:

$P(H_i|E)$ = la probabilità che sia vera l'ipotesi H_i data l'evidenza E

$P(E|H_i)$ = la probabilità di osservare l'evidenza E dato che sia vera l'ipotesi i

$P(H_i)$ = la probabilità *a priori* che sia vera l'ipotesi i in assenza di alcuna evidenza specifica

k = il numero di ipotesi possibili

Allora il teorema dice che

$$P(H_i \mid E) = \frac{P(E \mid H_i) * P(H_i)}{\sum_{n=1}^{k} P(E \mid H_n) * P(H_n)}$$

Esempio 1:

- è nota la probabilità a priori di trovare un materiale in un certo luogo
- è nota la probabilità che, se c'è un materiale in un certo luogo, si osservino certe proprietà fisiche

Il teorema di Bayes permette di calcolare la probabilità che in un luogo siano presenti vari minerali, a partire dall'evidenza raccolta (PROSPECTOR).

Esempio 2:

I caratteri siano descritti da un vettore di pixel (in realtà una matrice).

Input: vettore del carattere da riconoscere.

Inoltre:

- è nota la probabilità a priori della comparsa di ogni lettera
- è nota la probabilità che ogni lettera compaia come un certo valore del vettore.

Applicando Bayes:

$$P(carattere \mid vettore) = \frac{P(vettore \mid carattere) * P(carattere)}{\sum_k P(vettore \mid carattere_k) * P(carattere_k)}$$

Si effettua il calcolo per tutti i caratteri del set e si sceglie come vincitore il carattere che fornisce la probabilità più grande.

PROBLEMI DETERMINISTICI TRATTATI IN MODO PROBABILISTICO

Problemi complessi quando manca informazione sufficiente.

Esempio: la prossima mossa nel gioco degli scacchi può essere valutata col metodo dei cammini (teoricamente); nella pratica non si cerca il cammino ottimo fino alla meta, ma si ricorre all'euristica per valutare obiettivi intermedi (vantaggio sul numero di pezzi, mobilità, controllo del centro, ecc.).

Problema: come combinare queste evidenze?

Soluzione possibile:

calcolo di una funzione lineare, pesando ogni contributo.

Problema connesso: valutare i pesi.

Soluzione (e anche problema conseguente):
usare una procedura di addestramento, in modo da aggiornare in modo adattativo i pesi.

Ulteriori problemi:

- quali pesi influenzano la valutazione, e con che modalità?
- Come isolare l'influenza di ciascun fattore e poi come combinare le influenze? (problema rilevante nella diagnostica medica)

Può essere opportuno passare ad una rappresentazione basata su regole.

Un sistema a regole su basi probabilistiche può essere usato in sistemi complessi in cui si è interessati a diverse conclusioni e come strumento di base in sistemi di decisione, anche semplici, basati su tecniche di ricerca tradizionali.

Esempio:

Gioco degli scacchi: si vuole valutare la bontà di una posizione.

Si adottano regole di valutazione del tipo:

- Se è presente la configurazione A allora vi è qualche evidenza che la nostra regina sarà attaccata.
- Se è presente la configurazione B allora vi è qualche evidenza che la nostra regina sarà attaccata.
- Se la nostra regina è sotto attacco allora vi è qualche evidenza che perderemo
- Se l'altro giocatore ha un vantaggio materiale allora vi è qualche evidenza che perderemo.

Altro esempio di sistema basato su regole (di tipo medico-diagnostico):

- Se vi è evidenza di un organismo con proprietà X_1 e Y_1 allora vi è qualche evidenza che vi sia un'infezione causata dall'organismo Z.
- Se vi è evidenza di un organismo con proprietà X_2 e Y_2 allora vi è qualche evidenza che vi sia un'infezione causata dall'organismo Z.
- Se vi è un'infezione causata dall'organismo Z allora vi è qualche evidenza che il farmaco Q curerà il paziente.
- Se il paziente sta già prendendo il farmaco R allora vi è qualche evidenza che il farmaco Q non avrà alcun effetto.

Vantaggi: non occorre memorizzare esplicitamente tutte le combinazioni di sintomi per i quali il farmaco Q risulta efficace.

Occorre ancora associare ad ogni regola una probabilità. Esempio per la I regola:

- Se vi è evidenza di un organismo con proprietà X_1 e Y_1 allora con probabilità 0.7 vi è un'infezione causata dall'organismo Z.

Si usa quindi la teoria della probabilità per combinare le regole da usare nel processo di ragionamento.

Questo sistema di regole può essere accresciuto con l'inserimento di nuove regole o modificato: è dinamico, oltre che strutturato.

Conclusione:

- Un sistema a più alto livello conduce la ricerca nell'albero di gioco con l'algoritmo minimax
- A più basso livello, quando si arriva ad un nodo terminale, parte il sistema di valutazione basato su regole, che restituisce al sistema principale la sua risposta (valutazione).

Vantaggi:

- Si possono trattare con le stesse modalità e in modo intergrato situazioni in cui la conoscenza è incompleta e/o vi sono situazioni intrinsecamente probabilistiche (es.:diagnostica medica, riconoscimento di forme, ecc.)

- Si modella meglio il mondo reale, in cui non ci si esprime con fatti come:
  - Tutti i negozi sono chiusi la domenica
  - Nessuno vive più di 150 anni
  - Le macchine dei vigili del fuoco sono sempre rosse

  ma piuttosto:
  - La maggiorparte dei negozi sono chiusi di domenica
  - Quasi nessuno vive più di 100 anni
  - Le macchine dei vigili del fuoco in genere sono rosse.

- Associando le valutazioni probabilistiche a delle regole (e quindi non usando semplicemente valutazioni probabilistiche) il sistema mantiene capacità esplicative (può continuare a dare ragione delle scelte effettuate.

Problemi ancora da approfondire:

- Come dovrebbero essere interpretate le probabilità? Come si possono combinare insieme?
- Come possono venire trattati in modo adeguato eventi separati che non sono l'uno indipendente dall'altro, in modo da non contare più di una volta quella che è sostanzialmente la stessa evidenza?

- Quanto sforzo si dovrebbe impiegare per propagare attraverso il sistema i cambiamenti di probabilità? Si noti che se A → B con probabilità P_1 e B → C con probabilità P_2, allora se cambia la nostra fiducia in A, cambiando così la nostra fiducia in B, dobbiamo cambiare anche la nostra fiducia in C, che era basata su B. La questione di come evitare di dedicare tutto il tempo del sistema a propagare piccoli cambiamenti è simile al problema di propagare cambiamenti attraverso una base di conoscenza dello stesso tipo di TMS.

Le soluzioni sono molteplici.

Verrà considerata una soluzione collaudata come quella di MYCIN.

MYCIN:

- sistema basato su regole
- usa il ragionamento inesatto
- è destinato alla diagnostica medica (tratta le infezioni batteriche ed è capace di prescrivere terapie)
- interagisce con TEIRESIAS, il quale realizza il sistema esplicativo rispetto alle domande + permette di ampliare la base di conoscenza.

MYCIN opera in un dominio di conoscenza incerta.

È basato su regole come:

Se: (1) dalla provetta l'organismo risulta gram-positivo, e

(2) la morfologia dell'organismo è cocco, e

(3) la conformazione di crescita dell'organismo è a blocchi, allora vi è una rilevante evidenza (0.7) che l'identità dell'organismo è stafilococco.

Implementate in LISP, quindi con strutture a liste:

PREMESSA: ($AND
(SAME CNTXT GRAM GRAMPOS)
(SAME CNTXT MORPH COCCUS)
(SAME CNTXT CONFORM CLUMPS))

AZIONE: (CONCLUDE CNTXT IDENT
STAPHILOCOCCUS TALLY 0.7)

- MYCIN usa il ragionamento all'indietro: parte dalla meta (evidenziare gli organismi causa di patologie) e va verso i dati clinici disponibili (riduce il fattore di ramificazione).

- Usa molte regole. Come combinare le stime di certezza rilevate dall'uso di ciascuna per fornire un grado di certezza per le conclusioni?

Qui l'applicazione del teorema di Bayes non è opportuno per i seguenti motivi (n.b. queste critiche hanno validità generale!):

- Spesso è difficile raccogliere tutte le probabilità condizionali e congiunte a priori richieste. Per farlo è necessario accumulare un gran numero di dati. Inoltre è molto costoso. Ma, cosa ancora peggiore, i dati sarebbero obsoleti già nel momento in cui vengono raccolti.
  I microorganismi possono cambiare la loro reazione ai farmaci in un tempo minore di quello necessario per raccogliere abbastanza dati sulle loro reazioni precedenti.

- È molto difficile modificare la base dati di un sistema bayesiano a causa dei gran numero di interazioni fra le sue varie componenti. Per esempio, le probabilità di tutti i possibili risultati devono dare come somma 1. Supponiamo che sia così. Ma che cosa succede ora se vogliamo aggiungere alla base dati conoscenza su una nuova malattia? Molte delle entrate esistenti dovranno essere modificate per mantenere costante la somma delle probabilità.
  Questa è una debolezza seria in un sistema costruito per lavorare in un dominio di compito troppo complesso per essere descritto completamente, anche se rimanesse immutato, cosa che spesso non succede.

- Valutare la formula di Bayes in un dominio complesso richiede una grande quantità di calcolo poiché devono essere prese in considerazione tante probabilità, molte delle quali possono contribuire piuttosto poco alla precisione della risposta. Ma la precisione della risposta finale sarà limitata in ogni caso dalla precisione delle probabilità usate per calcolarla e, come abbiamo già discusso, tale precisione può non essere molto alta. Quindi un calcolo così dettagliato può essere uno sforzo sprecato.

- Perché la formula di Bayes dia una stima accurata della probabilità di un determinato risultato, tutti i risultati possibili devono essere disgiunti, vale a dire non deve poter succedere mai che due di essi si verifichino contemporaneamente. Spesso non è così. Per esempio, nel dominio di MYCIN, facilmente un paziente potrebbe avere due o anche parecchie infezioni diverse contemporaneamente.

- La precisione della formula di Bayes dipende anche dalla disponibilità di un insieme completo di ipotesi. In altre parole, deve succedere sempre che una delle ipotesi note sia vera. Questo spesso può non essere vero a meno di introdurre una finta ipotesi "nessuna delle precedenti“.
  Un paziente potrebbe avere una malattia che nessuno ha mai diagnosticato prima.

In alternativa a Bayes, MYCIN usa due indici:

MB misura di credenza
MD misura di non credenza

definiti così:

- Data un'ipotesi h e un'evidenza e, MB è la diminuzione proporzionale della non credenza in h.

In formule:

$$MB[h, e] = \begin{cases} 1 & \text{se } P(h) = 1 \\ \dfrac{\max[P(h \mid e), P(h)] - P(h)}{\max[1,0] - P(h)} & \text{altrimenti} \end{cases}$$

- analogamente, MD è la diminuzione della credenza in h fornita dall'evidenza e. In formule:

$$MD[h, e] = \begin{cases} 1 & \text{se } P(h) = 0 \\ \dfrac{\min[P(h \mid e), P(h)] - P(h)}{\min[1,0] - P(h)} & \text{altrimenti} \end{cases}$$

In pratica, MB misura quanto l'evidenza avvalora l'ipotesi, e vale 0 se non l'avvalora.

MD misura quanto l'evidenza avvalora la negazione dell'ipotesi ed è 0 se l'evidenza rende certa l'ipotesi.

Ovvero ancora, se un'evidenza aumenta la probabilità di h, allora MB[h,e] > 0 e MD[h,e] = 0; se la diminuisce, allora MD[h, e] > 0 e MB[h,e] = 0.

Da questi indici si ricava

CF fattore di certezza

definito come

$$CF[h,e] = MB[h,e] - MD[h,e]$$

- se CF > 0, il sistema desume che l'ipotesi è vera
- se CF < 0, il sistema desume che l'ipotesi è falsa.

Se per un fatto concorrono due evidenze s_1 e s_2, esse vengono combinate così:

$$MB[h,s_1\&s_2] = \begin{cases} 0 & \text{se } MD[h,s_1\&s_2] = 1 \\ MB[h,s_1] + MB[h,s_2]*(1- MB[h,s_1]) & \text{altrimenti} \end{cases}$$

$$MD[h, s_1 \& s_2] = \begin{cases} 0 & \text{se } MB[h,s_1\&s_2] = 1 \\ MD[h, s_1] + MD[h, s_2]*(1- MD[h, s_1]) & \text{altrimenti} \end{cases}$$

Cioè:

Se vi è certezza nella non credenza in h, MB vale 0,

altrimenti l'MB della prima osservazione viene incrementato con una frazione dell'MB dovuta alla seconda osservazione, pesata con quanto manca alla certezza (= 1) alla prima osservazione.

Esempio:

un'osservazione iniziale conferma una credenza in h con MB = 0.3

Quindi:

$MD[h,s_1] = 0$

$CF[h,s_1] = 0.3$

Una seconda osservazione conferma la credenza in h con $MB[h, s_2] = 0.2$

Combinando le due osservazioni si ha:

$MB[h, s_1 \& s_2] = 0.3 + 0.2 * 7 = 0.44$

$MD[h, s_1 \& s_2] = 0$

$CF[h, s_1 \& s_2] = 0.44$ ← cresce perché si hanno due conferme

Occorre anche poter combinare due ipotesi, o per congiunzione o per disgiunzione.

Le formule usate sono:

$MB[h_1 \& h_2, e] = \min(MB[h_1, e], MB[h_2, e])$

$MB[h_1 | h_2, e] = \max(MB[h_1, e], MB[h_2, e])$

In certe situazioni, l'evidenza non è un valore assoluto, ma ha un suo grado di fiducia (per esempio, è ricavata da una misura di laboratorio, da un qualche esperimento, ecc.).

Se MB'[h, s] è la misura di quanto si è sicuri della validità di *s*, e se *e* sono le osservazioni che portano a credere ad *s* (ad esempio, i test di laboratorio), allora:

$$MB[h, s] = MB'[h, s] * \max(0, CF[s, e])$$

Osservazioni:

1. Struttura di una regola:
   - ipotesi
   - collezione di elementi di evidenza richiesti per l'applicazione
   - fattore di certezza

Poiché data una h e una e una sola tra MB ed MD è $\neq 0$, per ogni regola si dà solo la CF (fornita dall'esperto medico).

Infatti:

se $P(h|e) > P(h)$ allora

$$MB[h, e] > 0 \text{ e } MD[h, e] = 0$$

se $P(h|e) < P(h)$ allora

$$MB[h, e] = 0 \text{ e } MD[h, e] > 0$$

se $P(h|e) = P(h)$ allora

$$MB[h, e] = MD[h, e] = 0$$

2. Vengono utilizzate regole complesse (con più elementi di evidenza) piuttosto che regole semplici da combinare insieme.
   L'ipotesi sottostante le regole di combinazione è l'indipendenza: ogni regola ingloba elementi non indipendenti!

   Infatti se un sintomo predicesse una malattia con CF = 0.7 e un secondo sintomo predicesse la stessa malattia con CF = 0.7, sarebbe errato affermare che la concomitanza dei sintomi predice la malattia con CF = 0.91 (combinazione dei due CF), se per caso succede che i due sintomi si presentano sempre insieme!

3. Sorgono dei problemi deduttivi se, rispetto ad un fatto, si inserisce una regola che descrive una relazione causale e un'altra che descrive una relazione causale inversa.

   Esempio:

   regola 1) Se la notte scorsa lo spruzzatore era in funzione (S)
   allora vi è una rilevante evidenza (0.9) che questa mattina il prato sarà bagnato (W).

   regola 2) Se questa mattina il prato è bagnato (W)
   allora vi è una rilevante evidenza (0.8) che la notte scorsa ha piovuto (R).

La concatenazione delle regole potrebbe portare a dedurre che:

ha piovuto (R) perché crediamo che lo spruzzatore era in funzione (S).

Soluzioni per questo problema:

- usare un solo tipo di regole
- partizionare i due tipi in modo che non interferiscano.

Questioni irrisolte:

1. Come trasformare i fattori di certezza espressi in termini umani in termini numerici. Per esempio, che cosa significa "è molto probabile che"?

2. Come normalizzare le diverse scale utilizzate dalle diverse persone, in modo particolare se la soluzione alla prima domanda consiste nel far sì che sia la gente a fornire direttamente numeri.

3. In che misura propagare i cambiamenti nei CF sulla base di nuova evidenza. Se il $CF[h_1, e]$ cambia molto leggermente e h_1 è parte dell'evidenza rilevante per un'altra ipotesi, h_2 , si dovrebbe cambiare anche il $CF[h_2, e]$? Se cambiamenti molto piccoli sono propagati sempre il più estesamente possibile, il sistema può utilizzare per questa operazione tutto il tempo a disposizione con un impatto molto scarso sul risultato finale. D'altra parte, molti piccoli cambiamenti possono, sommandosi, portare ad un cambiamento significativo che non dovrebbe essere ignorato.

4. Come fornire feedback alla base dati per migliorare la precisione dei CF delle regole. Questo problema in MYCIN è stato parzialmente risolto dalla capacità del sistema TEIRESIAS di spiegare il processo di ragionamento ad un medico e poi di accettare asserzioni da parte del medico su come le regole dovrebbero essere riformulate.

Organizzazione del Sistema

Per un buon sistema basato sulla conoscenza occorre integrare e far interagire diversi metodi di rappresentazione della conoscenza e differenti algoritmi e strategie.

L'approccio di scrivere tutte le regole e lasciare che sia il sistema ad usarle senza una supervisione si è rivelato poco efficace perché:

- l'aggiunta di nuove regole rivelate necessarie può portare ad interferenze con le regole precedenti;
- è inefficiente che ad ogni passo si provino tutte le regole
- alcune metodologie piuttosto che altre possono risultare efficaci in certi momenti o per affrontare particolari aspetti del problema.

La critica ad un approccio "piatto" porta come conseguenza ad indirizzarsi verso un approccio "a moduli" (Dijkstra: vale per la programmazione, vale per i grandi sistemi).

Come far interagire i moduli? Un buon approccio è quello proposto da HEARSAY-II, sistema per la comprensione del discorso, detto "*a lavagna*".

È costituito da un insieme di moduli detti *knowledge source* (KS, fonti di conoscenza), specializzati in differenti compiti e differenti domini, e da una struttura dati condivisa, detta appunto lavagna.

Quando una KS viene attivata, esamina il contenuto attuale della lavagna e

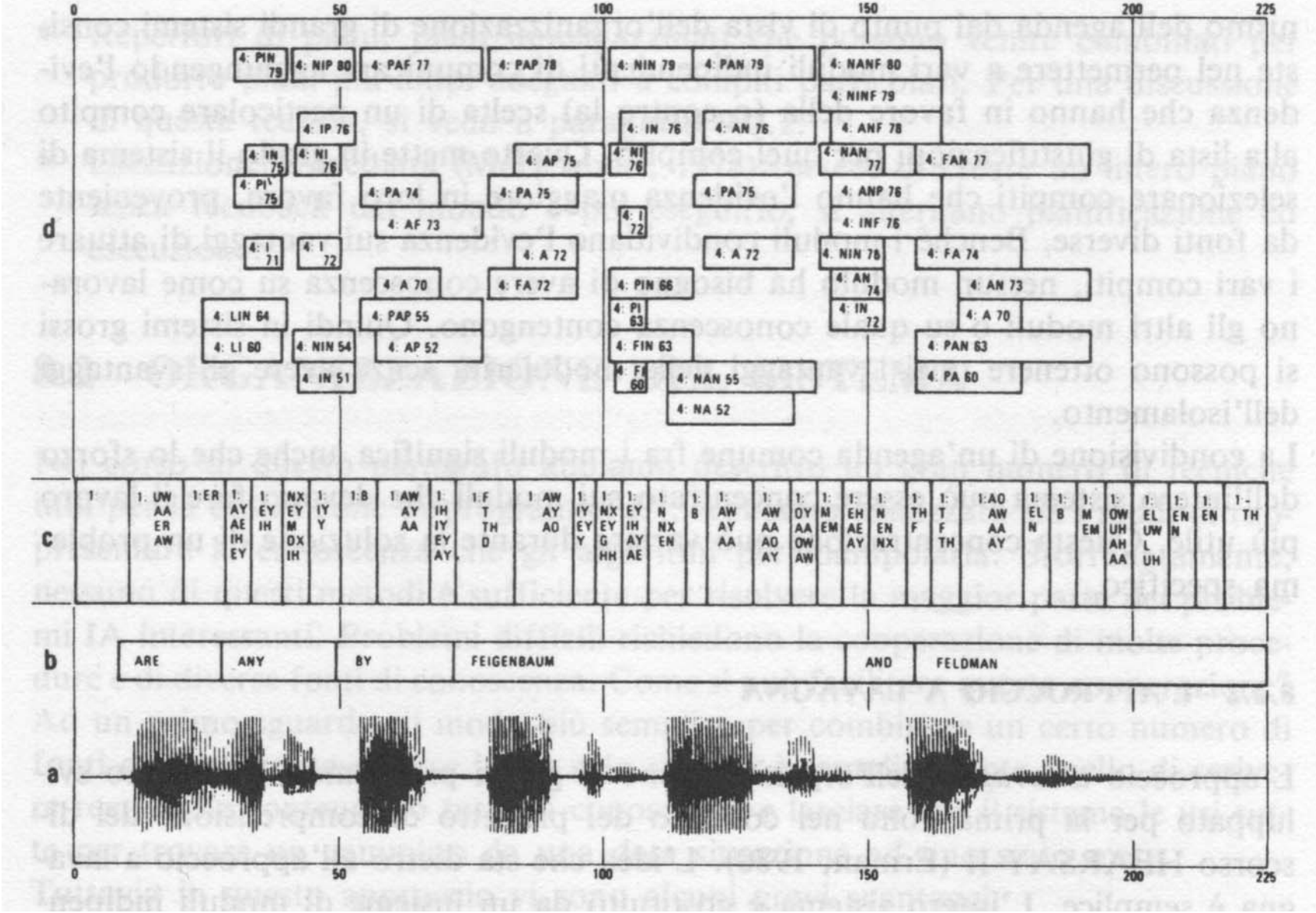
- crea una nuova ipotesi, ponendola sulla lavagna
- modifica una ipotesi già esistente.

La meta è di creare un'unica ipotesi che rappresenti la soluzione del problema (interpretare una frase).

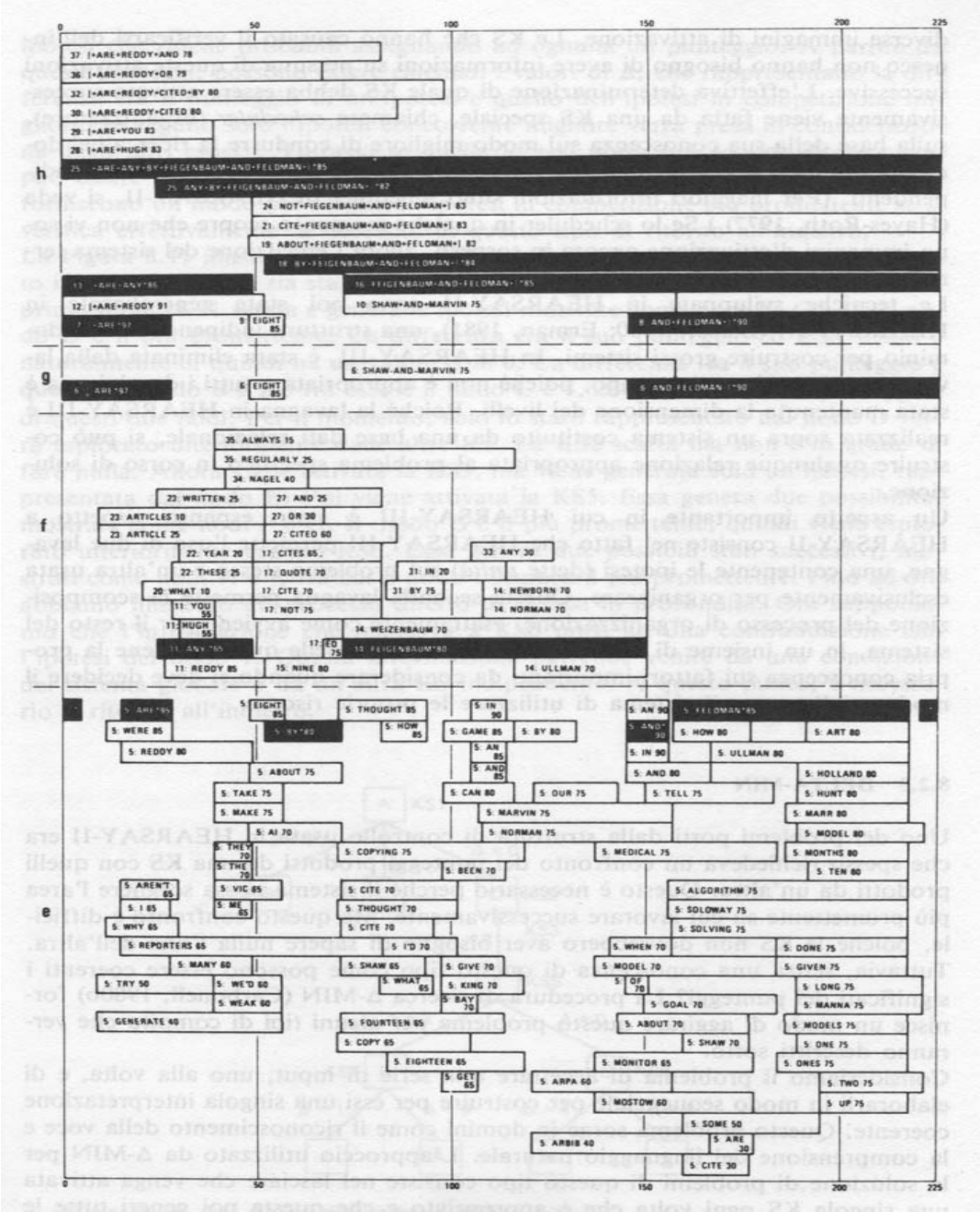
Si crea così una struttura "a livelli".

Esempio:

I livelli bassi di HEARSAY-II:


0
50
100
150
200
225
d
4: PIN 79
4: NIP 80
4: PAF 77
4: PAP 78
4: NIN 79
4: PAN 79
4: NANF 80
4: NINF 79
4: IP 76
4: IN 76
4: AN 76
4: ANF 78
4: IN 75
4: NI 76
4: AP 75
4: NI 76
4: NAN 77
4: FAN 77
4: PI 75
4: PA 74
4: PA 75
4: PA 75
4: ANP 76
4: AF 73
4: I 72
4: INF 76
4: I 71
4: I 72
4: A 72
4: A 72
4: NIN 78
4: FA 74
4: A 70
4: FA 72
4: PIN 66
4: AN 74
4: AN 73
4: LIN 64
4: PAP 55
4: PI 63
4: IN 72
4: A 70
4: LI 60
4: NIN 54
4: AP 52
4: FIN 63
4: PAN 63
4: IN 51
4: NAN 55
4: PA 60
4: NA 52
c
b
ARE
ANY
BY
FEIGENBAUM
AND
FELDMAN
a

La seconda parte di una lavagna HEARSAY-II:



I livelli sono:

a) La forma d'onda della frase
   ARE ANY FEIGENBAUM AND FELDMAN?
b) Le parole corrette mostrate solo come indicazione
c) I segmenti di suono
d) Le classi di sillabe
e) Le parole come vengono create da una KS sulle parole
f) Le parole come vengono create da una seconda KS sulle parole
g) Sequenze di parole
h) Espressioni linguistiche

Quando viene attivata una KS? Si adotta il paradigma dei "demoni": ad ogni KS è associato un insieme di trigger che specificano le condizioni di attivazione; una procedura controlla continuamente le condizioni che diventano vere e fa scattare i trigger.

Quando un trigger scatta, crea "una immagine di attivazione" (una descrizione della KS da attivare e l'evento stesso che ha fatto scattare il trigger: questo è utile per i processi di focalizzazione dell'attenzione).

Uno stesso evento può triggerare più KS: un modulo particolare, detto "scheduler", determina l'ordine con cui devono intervenire i KS attivati.

Lo scheduler fonda la sua attività sui punteggi forniti da ciascuna KS, utilizzando un algoritmo detto DELTA-MIN.

Se non ci sono immagini d'attivazione in sospeso, lo scheduler termina fornendo l'ipotesi risultante.

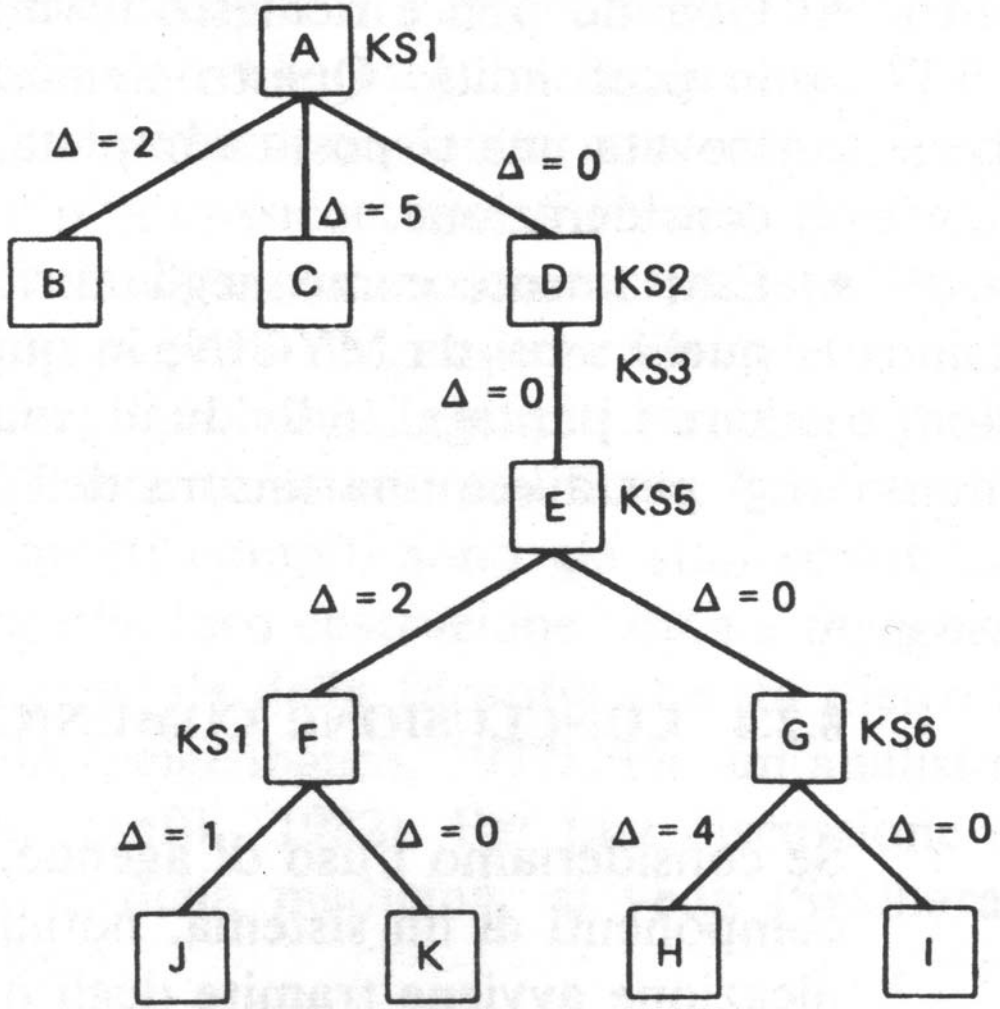
L'algoritmo delta-min

Come confrontare punteggi generati da KS differenti?

Ogni KS fornisce più ipotesi, assegnado ad ognuna un punteggio. All'ipotesi migliore si associa un valore $\Delta = 0$, e si calcolano i delta delle altre ipotesi per differenza.

Nella ricerca, si utilizza subito l'ipotesi migliore ($\Delta = 0$), mentre le altre ipotesi vengono prese in considerazione quando si verifica un ritorno all'indietro.

Si abbia ad esempio la seguente sequenza di attivazioni:



Nota: KS2 non produce ipotesi

All'inizio si percorre il grafo lungo A→D→E→G→I (si seguono le ipotesi migliori).

Al nodo I il sistema globale (o una KS specifica) rileva una contraddizione. Dove tornare indietro?

Se si sceglie la seconda ipotesi dopo G, si ha un $\Delta = 4$.

Ma se si torna al nodo E, la seconda ipotesi ha un $\Delta = 2$. Conviene tornare qui ed esplorare il nodo F.

Se procedendo anche il nodo K fallisce, si rivalutano i punteggi differenziali (cioè l'incertezza delle ipotesi: questa è la somma dell'incertezza locale e di tutte le ipotesi che le precedono).

Nel nostro esempio, la valutazione di J è 3, mentre B è valutata 2: si sceglie B.

Vantaggi del metodo:

- non vengono confrontati i punteggi di KS diverse, ma solo i delta (c'è meno dipendenza)
- se non è richiesto un ritorno all'indietro, si ha una ricerca in profondità
- come in MYCIN, si combinano punteggi per una valutazione di incertezza complessiva.