SISTEMI BASATI SULLA CONOSCENZA

- 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.

- PRINCIPI ARCHITETTURALI
  - Ogni sistema basato sulla conoscenza deve riuscire ad esprimere due tipi di conoscenza:
  - Conoscenza sul dominio dell'applicazione (COSA);
  - Conoscenza su COME utilizzare tale conoscenza per risolvere problemi - CONTROLLO.

PROGRAMMA = CONOSCENZA + CONTROLLO

- Questi due tipi di conoscenza dovrebbero essere tenuti SEPARATI:
  - ALTA MODULARITÀ
  - PROGRAMMAZIONE DICHIARATIVA

- La conoscenza sul problema è espressa indipendentemente dal suo utilizzo (COSA non COME);

- Alta modularità e flessibilità;

PROGRAMMA = CONOSCENZA + CONTROLLO

- I moderni linguaggi e AMBIENTI per Intelligenza Artificiale tendono a riprodurre tale schema;
- Definire un ambiente (linguaggio) in tale schema significa definire come il programmatore può esprimere la conoscenza e quale tipo di controllo può utilizzare;

- Problematiche di **rappresentazione** della conoscenza;
- Problematiche di organizzazione del problema e **strategie di controllo**

VANTAGGI DELL'APPROCCIO DICHIARATIVO

- **ALTO LIVELLO:**
  - è più semplice impostare il problema (catena di IF..THEN);
- **GENERALITÀ:**
  - posso usare la stessa conoscenza in differenti modi;
- **FLESSIBILITÀ**
  - facile modifica, debugging ecc.
  - cambio del controllo

UNA NUOVA MACCHINA VIRTUALE

UNA NUOVA MACCHINA VIRTUALE

ESEMPIO

- **Semplicissimo problema di diagnostica:**
  - prescrivere una medicina in base ai risultati di un esame di laboratorio.

- **GOAL:** `prescribe (Drug)`
  - cioè "prescrivere una medicina adeguata per un determinato paziente".

UNA NUOVA MACCHINA VIRTUALE

- FATTI:

  `gram(neg).`

  `not(allergic(antb)).`

- REGOLE:
  - **`R1: gram (neg) → id (ecoli).`**

    Se il risultato dell'esame è *gram-negativo* allora l'identità è *enterium-coli*
  - **`R2: gram (pos) → id (strep).`**

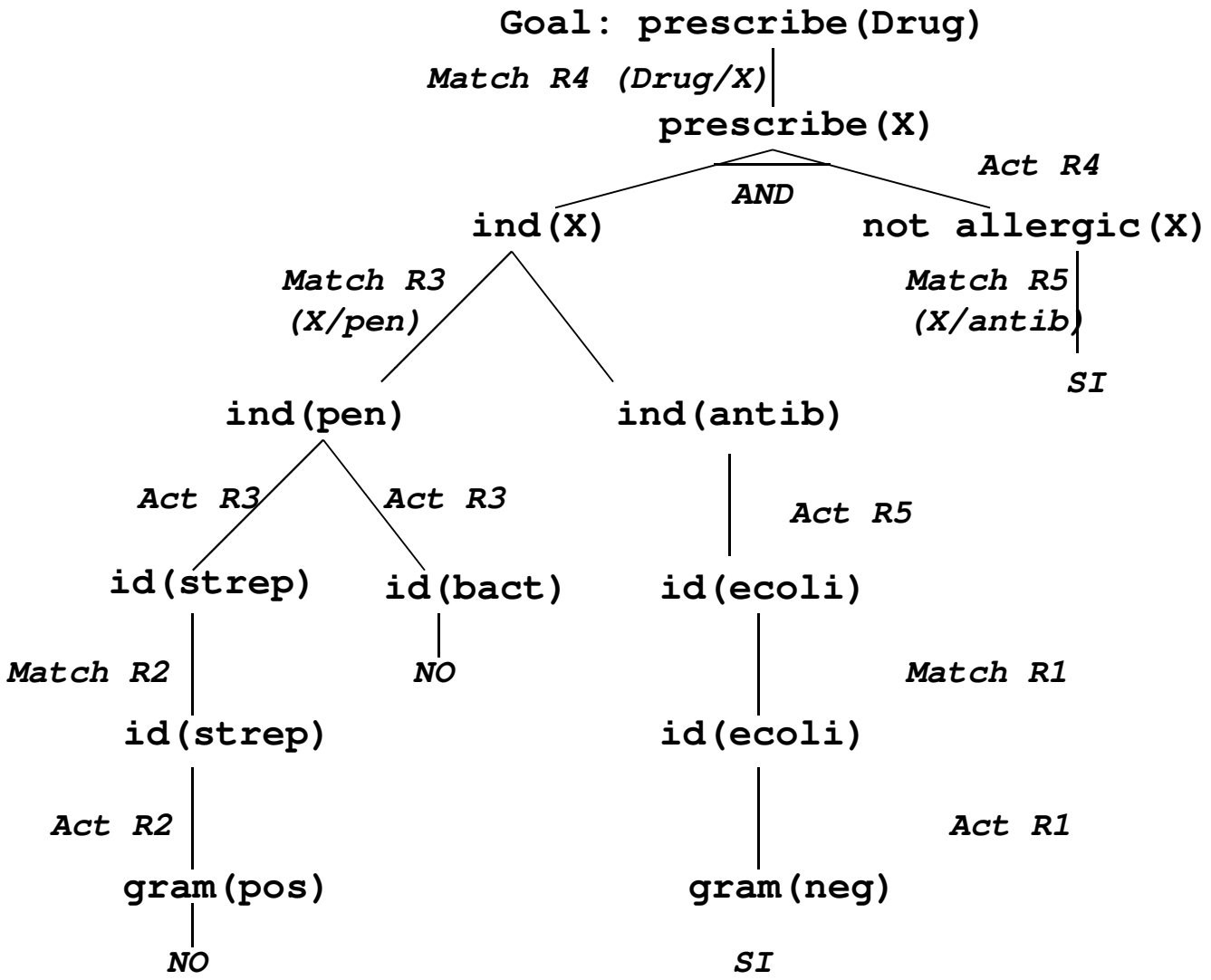
    Se il risultato dell'esame è *gram-positivo* allora l'identità è *streptococco*
  - **`R3: id(strep) OR id(bact) → ind(pen).`**

    Se l'identità è streptococco o bactero allora è bene indicare penicillina
  - **`R4: ind(X) AND not (allergic(X)) → prescribe(X)`**

    Se è bene indicare una certa medicina e il paziente non è allergico a tale medicina, allora si può prescrivere tale medicina al paziente
  - **`R5: id(ecoli) → ind(antb).`**

    Se l'identità è ***enterium-coli*** allora è bene indicare antibiotici

CONTROLLO BACKWARD
GRAFO AND/OR



CONSIDERAZIONI

- I fatti iniziali sono molto numerosi e dunque è bene domandarli selettivamente all'utente quando necessario.
- L'impostazione del problema preferibile è backward.
- askable: gram e allergic.

- Esempio di sessione di lavoro:
  - User: esecuzione
  - Sys: È la coltura Gram positiva ?
  - User: NO.
  - Sys: È la coltura Gram negativa?
  - User: SI.
  - Sys: Il paziente è allergico agli antibiotici ?
  - User: NO.
  - Sys: Allora prescriverei una cura a base di antibiotici.

SESSIONE DI SPIEGAZIONE

- Sys: È la coltura Gram positiva ?
- User: **WHY?** < perchè è importante determinare tale informazione?
- Sys: Tale informazione è di aiuto per determinare se (R1) l'identità dell'organismo è streptococco.
- User: **WHY?**
- Sys: Tale informazione è di aiuto perchè per tale organismo è indicata la penicilina (R3).
- Sys: ... Allora prescriverei una cura a base di antibiotici.
- User: **HOW?** <come puoi concludere tale responso?
- Sys: Ho utilizzato la regola R4: ho prescritto antibiotici poichè sono indicati per tale infezione (prem1) e il paziente non è allergico (prem2).
- User: **HOW Premessa 1 ?**

  <come puoi concludere che sono indicati gli antibiotici?
- Sys: Ho utilizzato la regola R5: sono indicati antibiotici poichè l'organismo che ha causato l'infezione è ecoli.

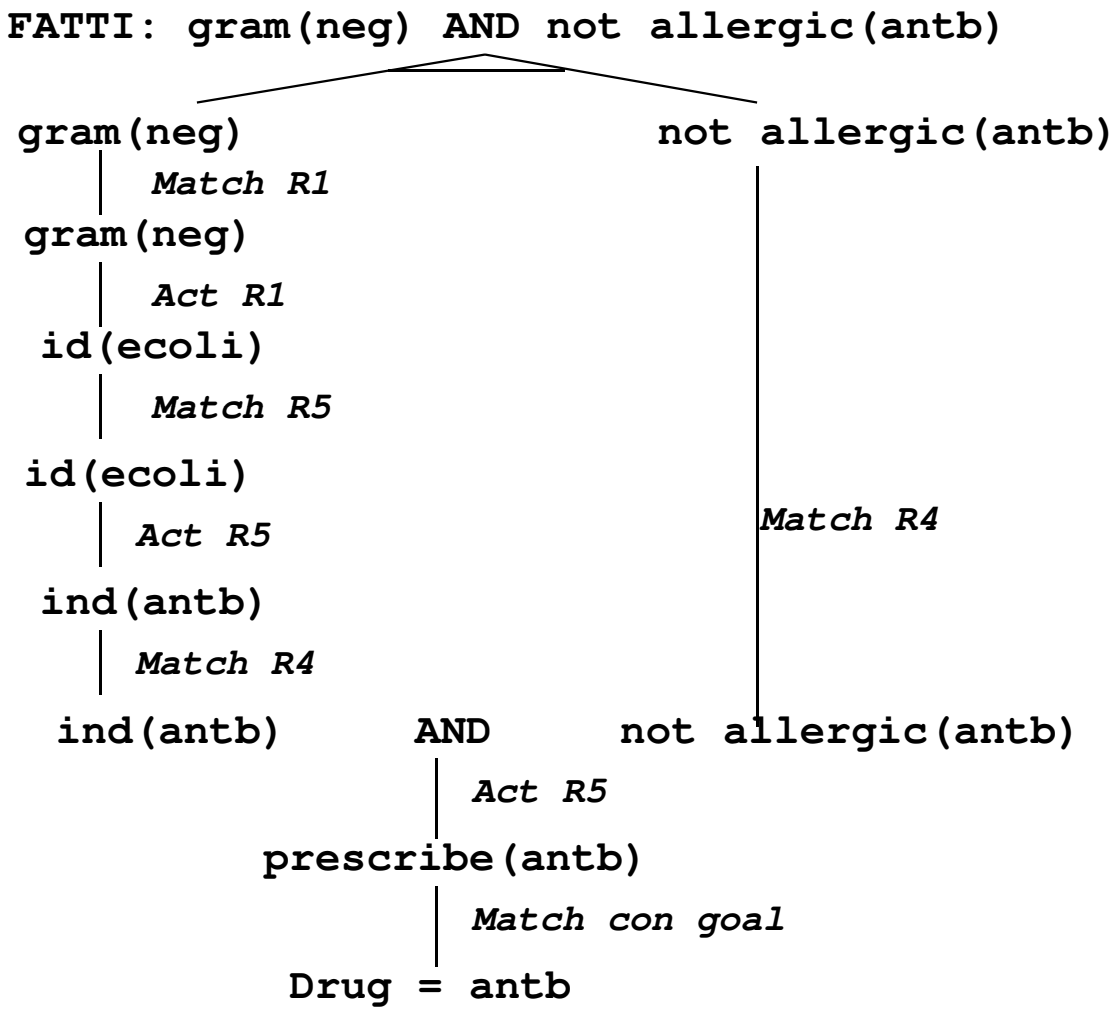
STRUTTURA ALGORITMICA: Parte 1a

```
program prescribe(input,output);
var Answer: string[2]; presc: boolean;
begin
presc:=false;
write('e la coltura Gram Positiva?');
read(Answer);
if Answer='Si' then
begin
write('il paziente è allergico alla penicillina?');
read(Answer);
    if Answer='No' then
    begin
    write('allora prescriverei una cura a base di
      penicillina');
    presc:=true;
    end;
end;
```

STRUTTURA ALGORITMICA: Parte 2a

```
if not presc then
begin
write('coltura Gram Negativa?');
read(Answer);
    if Answer='Si' then
    begin
    write('il paziente è allergico
    agli antibiotici?');
    read(Answer);
        if Answer='No' then
        begin
        write('allora prescriverei una cura a base di
          antibiotici');
        presc:=true;
        end;
    end;
end;
if not presc then
write('prescrizione impossibile'); end.
```

CONTROLLO FORWARD
GRAFO AND/OR



Schema architetturale: SISTEMA DI PRODUZIONI

- Insieme di Operatori (regole);
- Uno o più database (Memorie di lavoro);
- Strategia di Controllo.
- MODULARITÀ - FLESSIBILITÀ
- Operatori:
  - **`IF <pattern> THEN <body>`**
  - non si chiamano per nome, ma si attivano in base al pattern-matching

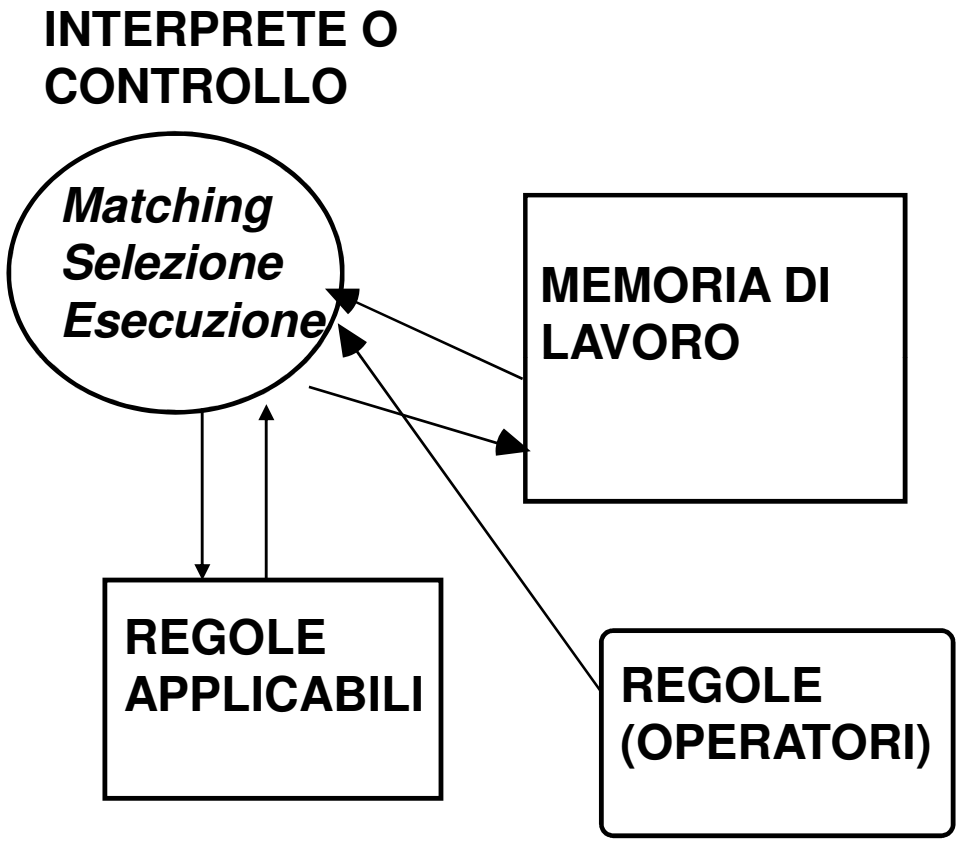
Production-rule systems:

- Definizione:

> Programmi che realizzano metodi di ricerca per problemi rappresentati come spazio degli stati.

- Consistono di:
  - Un insieme di regole,
  - una 'working memory', che contiene gli stati correnti raggiunti
  - una strategia di controllo per selezionare le regole da applicare agli stati della 'working memory' (matching, verifica di precondizioni e test sullo stato goal se raggiunto).

ARCHITETTURA GENERALE:



DUE MODALITÀ DI "RAGIONAMENTO"

- FORWARD O DATA-DRIVEN:
  - La memoria di lavoro nella sua configurazione iniziale contiene la conoscenza iniziale sul problema, cioè i fatti noti.
  - Le regole di produzione applicabili sono quelle il cui antecedente può fare *matching* con la memoria di lavoro (F-rules).
  - Ogni volta che una regola viene selezionata ed eseguita nuovi fatti dimostrati vengono inseriti nella memoria di lavoro.
  - Il procedimento termina con successo quando nella memoria di lavoro viene inserito anche il goal da dimostrare (condizione di terminazione).

DUE MODALITÀ DI "RAGIONAMENTO"

- BACKWARD O GOAL-DRIVEN:
  - La memoria di lavoro iniziale contiene il goal (o i goal) del problema.
  - Le regole di produzione applicabili sono quelle il cui conseguente può fare *matching* con la memoria di lavoro (B-rules).
  - Ogni volta che una regola viene selezionata ed eseguita, nuovi subgoals da dimostrare vengono inseriti nella memoria di lavoro.
  - Il procedimento termina con successo quando nella memoria di lavoro vengono inseriti fatti noti (CONDIZIONE DI TERMINAZIONE).

QUANDO APPLICARE BACKWARD E QUANDO FORWARD ?

- Esistono più Stati Iniziali o più Goals?
- Quale è il numero medio di rami generati da un singolo nodo?
- Quale è la modalità di ragionamento più naturale? (spiegazione all'utente)

- BIDIREZIONALE O MISTO:
  - È la combinazione dei metodi descritti precedentemente;
  - La memoria di lavoro viene suddivisa in due parti l`una contenente i fatti e l'altra i goals o subgoals;
  - Si applicano simultaneamente F-rules e B-rules alle due parti di memoria di lavoro e si termina il procedimento con successo quando la parte di memoria di lavoro ricavata mediante backward chaining è uguale o un sottoinsieme di quella ricavata mediante forward chianing (CONDIZIONE DI TERMINAZIONE).