Rule Based Systems
Rule-based Systems

Simplest method for building an Intelligent System

Example: Rule base

1. COLLAT ∧ PYMT ∧ REP ⇒ OK
2. APP ⇒ COLLAT
3. RATING ⇒ REP
4. INC ⇒ PYMT
5. BAL ∧ REP ⇒ OK

Description

COLLAT  satisfactory collateral
PYMT    payments undisputed
REP     good reputation
APP     high appraisal
RATING good rating
INC     positive income
BAL     excellent balance
OK      loan will be approved

Two modi of usage

Decision about loan: Facts are retrieved from database or user
Explanation of Decision: System answers the why question
Decision Rule Tree

Corresponding and/or-Tree:

```
OK
  /\       \ /
OK  /\  rule 1  \ /\  rule 2
OK  /\  COLLAT  \ /\  INC
Bal  \  REP   \  REP   \  REP
     \  REP   \  COLLAT  \  RATING
        \  RATING
```

rule 5
Example Queries

**User:** Why do you believe that the loan should be approved?

**System:** Because the collateral for the loan is satisfactory, and the applicant is able to make the loan payments, and the applicant has a good financial reputation.

**User:** Why did you ask me about whether or not the applicant’s income exceeds his/her expenses?

**System:** Because I was trying to establish whether or not the applicant is able to make the loan payments.

**User:** Why do you believe that the applicant has a good financial reputation?

**System:** Because the credit rating service said that the applicant has a good credit rating.
A rule in general is a *if-then*-construct consisting of a *condition* and an *action*.

If *condition* then *conclusion*

These two parts may be interpreted differently according to the context:

- **Inference rules**: If *premise* then *conclusion*
- **Hypotheses**: If *evidence* then *hypothesis*
- **Productions**: If *condition* then *action*

Rules are often referred to as *productions* or *production rules*.
A rule represents a unit of knowledge.

A set of rules together with an execution/evaluation strategy comprises a program to find solutions to specific problem classes (Expert System).

The Prolog (logical programming language) is perfect for building: rule-based system

Rule-based systems are historically the first types of AI systems and were for a long time considered prototypical AI Systems.

Nowadays, AI system are often developed by use data-driven systems (Big Data, Deep Learning) for decision and learning.
**Forward chaining**

Expansion of knowledge base: as soon as new facts are inserted the system also calculates the conclusions/consequences.

Data-driven behavior

Premises-oriented reasoning: the chaining is determined by the left parts of the rules.

**Backward chaining**

Answering queries

Demand-driven behavior

Conclusion-oriented reasoning: the chaining is determined by the right parts of the rules.
Components of a Rules-based System

**Data base**
- Set of structured data objects
- Current state of modeled part of world

**Rule base**
- Set of rules
- Application of a rule will alter the data base

**Rule interpreter**
- Inference machine
- Controls the program flow of the system
Main scheme forward chaining

- Select and apply rules from the set of rules with valid antecedences. This will lead to a modified data base and the possibility to apply further rules.

Run this cycle as long as possible.

The process terminates, if

- there is no rule left with valid antecedence
- a solution criterion is satisfied
- a stop criterion is satisfied (e.g. maximum number of steps)

Following tasks have to be solved:

- Identify those rules with a valid condition
  ⇒ Instantiation or Matching

- Select rules to be executed
  ⇒ need for conflict resolution (e.g. via partial or total orderings on the rules)
Simple Example: Intelligent Assistant Systems

Suppose you are trying to determine if a patient has inhalational anthrax. You observe the following symptoms:

- The patient has a cough
- The patient has a fever
- The patient has difficulty breathing
Simple Example: Intelligent Assistant Systems

You would like to determine how likely the patient is infected with inhalational anthrax given that the patient has a cough, a fever, and difficulty breathing.

We are not 100% certain that the patient has anthrax because of these symptoms. We are dealing with uncertainty!
Now suppose you order an x-ray and observe that the patient has a wide mediastinum.

Your belief that the patient is infected with inhalational anthrax is now much higher.
Qualities of Knowledge

In this example and in most cases our knowledge about the present world is:

- **incomplete/missing** (knowledge is not comprehensive)
  - e.g. “I don’t know the bus departure times for public holidays because I only take the bus on working days.”

- **vague/fuzzy/imprecise** (knowledge is not exact)
  - e.g. “The bus departs roughly every full hour.”

- **uncertain** (knowledge is unreliable)
  - e.g. “The bus departs probably at 12 o’clock.”

In most real world applications we have to decide nonetheless in the presence of incompleteness, imprecision uncertainty!

So handling uncertainty is a major part of all "intelligent systems"!
Topic of the course: How to handle uncertainty (subjective belief).

One possibility to express this partial belief is by using (subjective) probability theory. Probabilities are used to formally describe the “uncertainty” of an agent.

The first „intelligent“ Systems (expert system 1975) used a different method for the quantification of partial belief, the so called certainty factors.
Certainty Factors
First Expert systems: Mycin (1970)

Objective: Development of a system that supports physicians in diagnosing bacterial infections and suggesting antibiotics.

Features: Uncertain knowledge was represented and processed via uncertainty factors.

Knowledge: 500 (uncertain) decision rules as static knowledge base.

Case-specific knowledge:
- static: patients’ data
- dynamic: intermediate results (facts)

Strengths:
- diagnosis-oriented interrogation
- hypotheses generation
- finding notification
- therapy recommendation
- explanation of inference path
A Mycin Rule

RULE035

PREMISE: \( \text{($AND \ (\text{SAME CNTXT GRAM GRAMNEG}) \ (\text{SAME CNTXT MORPH ROD}) \ (\text{SAME CNTXT AIR ANAEROBIC}))} \)

ACTION: \( \text{(CONCL.CNTXT IDENTITY BACTEROIDES TALLY .6)} \)

If 1) the \textit{gram stain} of the organism is \textit{gramneg}, and
2) the \textit{morphology} of the organism is \textit{rod}, and
3) the \textit{aerobicity} of the organism is \textit{anaerobic}

then there is suggestive evidence (0.6) that the \textit{identity} of the organism is \textit{bacteroides}
Probabilistic modelling of Mycin: Handling of a 500-dim probability space is needed. Therefore, the developer in Stanford used the concept of certainty factors.

Uncertainty factor: $CF \in [-1, 1] \approx$ degree of belief

Rules:

$$CF(A \rightarrow B) \begin{cases} 
1 & , B \text{ is certainly true given } A \\
> 0 & , A \text{ supports } B \\
0 & , A \text{ has no influence on } B \\
< 0 & , A \text{ provides evidence against } B \\
-1 & , B \text{ is certainly false given } A 
\end{cases}$$
Example

\[ A \rightarrow B [0.80] \]
\[ C \rightarrow D [0.50] \]
\[ B \land D \rightarrow E [0.90] \]
\[ E \lor F \rightarrow G [0.25] \]
\[ H \rightarrow G [0.30] \]

\[ A [1.00] \]
\[ C [0.50] \]
\[ F [0.80] \]
\[ H [0.90] \]
**Propagation Rules**

**Conjunction:** \( CF(A \land B) = \min\{CF(A), CF(B)\} \)

**Disjunction:** \( CF(A \lor B) = \max\{CF(A), CF(B)\} \)

**Serial Combination:** \( CF(B, \{A\}) = CF(A \rightarrow B) \cdot \max\{0, CF(A)\} \)

**Parallel Combination:** for \( n > 1 \):
\[
CF(B, \{A_1, \ldots, A_n\}) = \\
f(CF(B, \{A_1, \ldots, A_{n-1}\}), CF(B, \{A_n\}))
\]

with

\[
f(x, y) = \begin{cases} 
  x + y - xy & \text{if } x, y > 0 \\
  x + y + xy & \text{if } x, y < 0 \\
  \frac{x + y}{1 - \min\{|x|, |y|\}} & \text{otherwise}
\end{cases}
\]
Example (cont.)

\[ f(0.3 \cdot 0.9, 0.25 \cdot 0.8) = 0.27 + 0.2 - 0.27 \cdot 0.2 = 0.416 \]
Was Mycin a failure?

It worked in the Mycin case because the rules had tree-like structure.

It can be shown that the rule combination scheme is inconsistent in general.

Example: \( \text{CF}(A) = 0.9, \text{CF}(D) = ? \)

\[ \text{CF}(D) = 0.9 + 0.9 - 0.9 \cdot 0.9 = 0.99 \]

\[ \text{CF}(D) = 0.9 \]

Certainty factor is increased just because (the same) evidence is transferred over different (parallel) paths!
Was Mycin a failure?

The certainty factor approach should not be used: It is a simple extension fo rule based systems, but in most cases inconsistent.

In the case of MYCIN it worked quite well (with probabilistic methods one can show, why!). In other applications the result was a desaster...

Mycin was never used for its intended purpose, doctors physicians were distrustful and not willing to accept Mycin’s recommendations (nor other recommendations by computers).

However,

- Mycin was a milestone for the development of expert systems.
- It gave rise to impulses for expert system development in general.

Nowadays most intelligent systems use probabilistic methods for handling uncertainty.