Intelligent Systems: Reasoning and Recognition

James L. Crowley

MoSIG M1 Winter Semester 2021

Lecture 14 25 march 2021

Symbolic Reasoning, Expert Systems and MYCIN

Outline:

Expert Systems and the AI Revolution of the 1980s

An expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning through bodies of symbolically encoded knowledge, with reasoning generally driven by a collection of if–then rules.

Prior to 1980, Artificial Intelligence was considered a marginal or even superficial research area. Serious research could be found at very few Universities, with most results provided by MIT, CMU and Stanford.

After several initial attempts in the 1970s, the first widely successful expert system was the MYCIN antibiotic therapy advisor produced by members of the Heuristic Programming project at Stanford in 1979. The use of rule based reasoning and structured knowledge representation introduced in MYCIN was replicated in several commercial environments, and commercialized as a general AI tool by a start up named Teknowledge.

A number of successful systems in the early 1980s were demonstrated to provide enormous return on investment. For example, the PROSPECTOR system developed at SRI (Stanford Research Institute) Artificial Intelligence Center for the U.S. Geological Survey rapidly discovered an unknown deposit molybdenum in Washington State, worth more than 100 times the cost of development. The R1 system developed in the summer of 1978 at CMU enabled Digital equipment corporation to save over $25 Meg in 1980 by automatically producing optimal configuration of Vax computers. Expert systems were widely used to organize logistics for major companies, NASA and the US Military.

In the period 1980 to 1985, the success of expert systems drove a massive wave of media interest and investment in AI. AI became a reputable research topic taught as part of computer science curriculum at major universities.

Knowledge and Reasoning

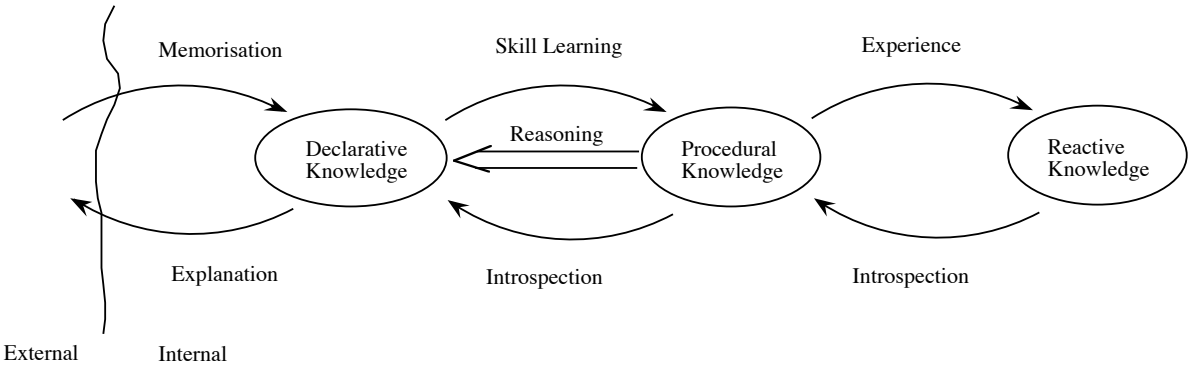
What is knowledge? - Competence
According to A. Newell [1980], knowledge is whatever enables the solution of problems.

Kinds of Knowledge
In the 1970s Cognitive Psychologists commonly identify several different categories of knowledge representation.

Declarative: A symbolic expression of competence.
Declarative knowledge is abstract
Declarative knowledge is used to communicate and to reason.
Declarative knowledge must be interpreted to be used.

Procedural: A series of steps to solve a problem

Reactive: stimulus - response. Perception-Action.



A hierarchy of knowledge representations.

Superficial knowledge provides reasoning without understanding. A common example of **superficial** reasoning is reasoning by symbol manipulation, without regard to the meaning of the symbols.

Deep knowledge requires the ability to predict and explain, and requires some form of model combined with experience.

Expert Systems are a technology that attempts to solve problems purely with Declarative and Procedural representations.

Knowledge Based system

An expert system is a computer system that emulates the decision-making ability of a human expert using symbolic encodings of human knowledge. Expert (knowledge based) systems were found to be useful for domains that are

Subjective,
Poorly formalized, and
Require manipulating large numbers of poorly related facts.

Examples include diagnosis, counseling, debugging, game playing, design in complex spaces and problem solving.

Expert system provided an alternative to algorithmic programming.
In the 1980s an expert system provided a first viable technology for Artificial Intelligence.

Expert System = Inference Engine + Domain Knowledge

An expert system typically combines a domain independent "inference engine" with a database of domain knowledge. The domain knowledge is encoded symbolically as rules and facts.

Expert Systems are constructed by hand coding symbolic expressions of the expertise provided by a "domain expert". Such systems are constructed by iterative refinement through the collaboration of a programmer and a domain expert. The programmer "imitates" the system behavior, evoking corrections and advice by the expert. The programmer then encodes these as fact, rules and data structures. The result is a system that proposes solutions to problems using superficial reasoning. Such systems may appear to understand but are in fact superficial. They tend to reason without regard to meaning. The system can make non-sense statements if applied outside their domain.

The technologies for "Expert Systems" are based on symbol manipulation, without regard for the meaning of the symbols. Thus Expert Systems are based on symbolic encoding of declarative and procedural knowledge from experts. The reasoning is syntactic. Thus it is shallow. The system cannot judge when it has been asked to solve a problem outside its domain.

The MYCIN Expert System

In the 1970's, at Stanford University, Edward Feigenbaum directed a research group named the "Heuristic Programming Project". Their central thesis was

> Intelligence = Large quantity of domain knowledge and a little bit of reasoning.

This led to an investigation into Knowledge Representation Techniques.

From 1970 to 1973 the sought to build a system that could interpret data from Mass Spectrograms.

A Mass Spectrograph is a device that uses electrical or magnetic fields to determine the masses of atoms or molecules in a sample. A beam of ions is passed through the electrical or magnetic field. The field deflects the ions at different angles depending on their masses, thereby breaking the beam into separate, identifiable bands.

The result of their project was a system named DENDRAL. DENDRAL was an un-maintainable "hack". However, by 1973 the group had learned to express declarative knowledge as "rules". It was decided to start over, building a "rule based" system for "anti-biotic Therapy".

Penicillin was discovered in 1929 and came into widespread use as an antibiotic in the 1940's. During the 1950's and 1960's a variety of new antibiotics were discovered. Each had unique properties and uses. By the 1970s, most medical doctors required consultation with a specialized expert to prescribe antibiotics.

The Stanford University Medical School was a world famous center for research in antibiotics. The Medical School asked the Computer Science School for help. Feigenbaum proposed to construct an "Artificial Expert" antibiotic therapy advisor.

The MYCIN system as created in the doctoral research of Edward Shortliffe, under that direction of Bruce Buchanan in the HPP team at Stanford.

The system developed from 1973 to 1978. It evolved into the first true "Expert System". As such it became the model for a new class of systems. It also revealed the importance of reasoning with uncertainty and the knowledge acquisition problem.

MYCIN: An Antibiotics Therapy Advisor.

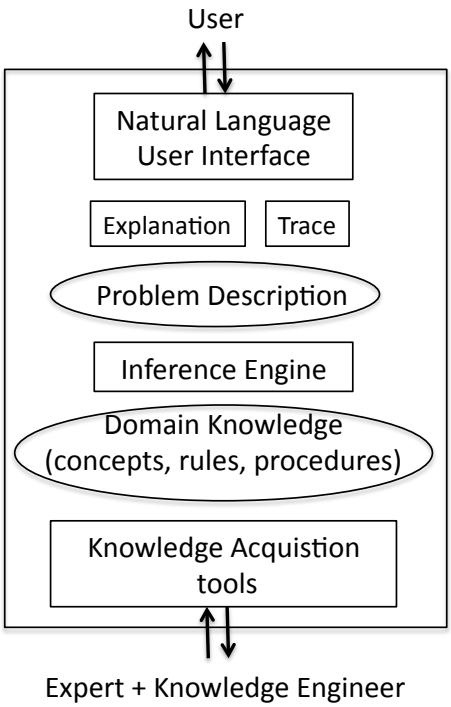
By 1975, a large variety of Antibiotics were available. Each antibiotic was effective against a specific set of microbes, and triggered a specific set of side effects.

Patients were often allergic to certain families of antibiotics. MYCIN was designed to be used by ordinary doctors who lacked the specialized training required to develop anti-biotic therapies. The system was required to be:

- Easy to use
- Reliable
- Able to manipulate large numbers of unrelated facts.
- Able to use inexact and incomplete facts
- Able to explain its advice.

The MYCIN architecture

The heart of the MYCIN system was a simple inference engine that used backward chaining rule to develop a tree that described a user's problem and offered advice for a solution. The tree (problem description) was constructed using rules and concepts about the domain provided by a domain expert working with a programmer (knowledge engineer). During the process, the system interacted with the user using pseudo natural language. The dialog was driven by interpreting the problem description and related rules with natural language. At any time the user could ask why and obtain a detailed, pre-coded explanation of the systems concepts. The user could as how, and receive an interpretation of the system reasoning in natural language.



Reasoning with Backward Chaining Rules

MYCIN was included approximately 600 rules, manipulating a large base of structured facts. The rules provided procedural knowledge to

1) Request or infer the required information
2) Apply specialized knowledge to determine a therapy
3) Provide advice to doctors in the form of proposed therapy.
4) Respond to questions about its reasoning.

Facts in MYCIN were represented by 4 items: (C, P, V, CF)

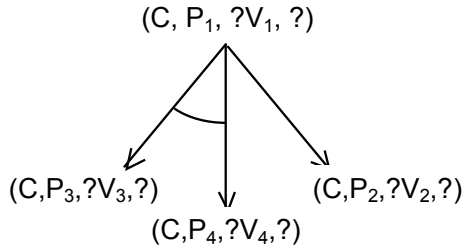
C = Context (a group of facts related to an entity
P = Parameter (facts)
V = Value (Boolean, Numeric, Symbol, List)
CF = Confidence Factor $CF:[-1,1]$

Backward Chaining with Rules generate Goals of the form (C, P, ?V, ?)
For reasoning. Goals are interpreted with an AND-OR Tree.

For example,

IF Goal = (C, P_1, ?V_1, ?) Then Ask (C, P_2, ?V_2, ?)
IF Goal = (C, P_1, ?V_1, ?) Then Ask (C, P_3, ?V_3, ?) AND (C, P_4, ?V_4, ?)

Graphical Representation:



Goals are then expanded recursively with additional rules.
Some rules can recursively open a new context. This context must be completed before the previous context.

Domain Concepts (Facts)

All facts in MYCIN are represented by a "quadruple":

(Context, Parameters, Value, CF)

C = Context (a group of facts related to an entity
P = Parameter (facts) about a context
V = Value (Boolean, Numeric, Symbol, List)
CF = Confidence Factor $CF : [-1,1]$

Contexts are used to structure reasoning and provide control by enabling rules.
For the MYCIN antibiotic therapy advisor, 10 contexts were required.

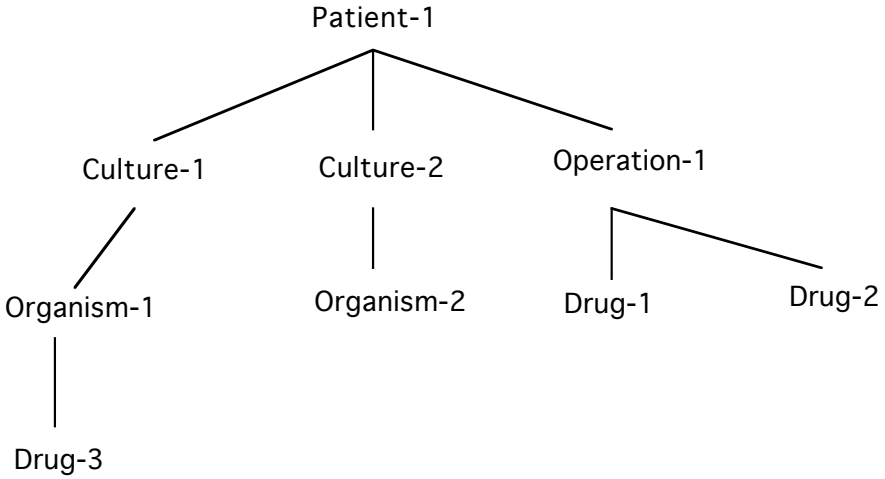
PERSON: Data about the patient, including age, sex and weight
OPERS: Past Medical procedures
CURCULS: Medical cultures taken from the patient
CURDRUGS: Current drug therapies for the patient
CURORGS: Known microbial infections in the patient
OPDRUGS: Drugs used during recent medical procedures
PRIORCULS: Past medical cultures
PRIORDRUGS: Past medical therapies
PRIORORGS: Past infections.

Contexts are organized in a tree.

MYCIN Context Tree was structured to respond to 4 questions.
1) What symptoms does the patient show
2) What microbes infect the patient
3) What antibiotics are effective against the microbes
4) What is the most appropriate antibiotic.

The context tree was called the "Dynamic Tree", and was composed of instances of each context.

The context tree served to focus reasoning. The following is a tree that describes a patient who has had an operation, and has been diagnosed with two simultaneous infections.



Parameters: the attributes of facts

Each context was composed of a number of parameters. Each parameter was described by a data structure.

Descriptors (facets) of Parameters included:

Expect : {Y/N, NUMB, ONE_OF, ANY_OF}
PROMPT : A sentence to ask for the value of the parameter.
LABDATA (Y/N) : Whether the parameter should be requested from the doctor or inferred automatically by the system.

(LABDATA was later renamed "ASK-FIRST")

LOOKAHEAD: A list of rules for inferring the value of a Parameter.
TRANS: An English language explanation of the parameter and the meaning of its values. Extensive pre-coding of English sentences allowed to system to appear capable of intelligent dialog. The MYCIN system could almost pass the Turing Test!

PARAMETER Categories:

SingleValued: Parameter could take a single value. If multiple values are provided the most likely must be determined.

MultiValued: The parameter could have multiple values. A fact was created for each value.

Binary: A Boolean value; A single value that can be Yes or No

Reasoning between alternative single valued parameters required some for me evidential reasoning. For this, the MYCIN team invented a "Confidence Factor".

The MYCIN Confidence Factor

ALL facts in MYCIN are labeled with a confidence Factor $CF \in [-1, 1]$
ALL rules in MYCIN are labeled with a FORCE: $CF \in [-1, 1]$.

As facts are established the conclusions are computed along with a confidence factor.

$$(C, P_2, V_2, CF_2) \xrightarrow{CF_R} (C, P_1, V_1, CF_1)$$

$$CF_1 = CF_R \cdot CF_2$$

$$(C, P_3, V_3, CF_3) \text{ AND } (C, P_4, V_4, CF_4) \xrightarrow{CF_R} (C, P_1, V_1, CF_1)$$

$$CF_1 = CF_R \cdot \min\{CF_3, CF_4\}$$

Mycin rules can be disjunctive (use OR)

$$(C, P_2, V_2, CF_2) \text{ OR } ((C, P_3, V_3, CF_3) \text{ AND } (C, P_4, V_4, CF_4)) \xrightarrow{CF_R} (C, P_1, V_1, CF_1)$$

$$CF_1 = CF_R \cdot \max\{CF_2, \min\{CF_3, CF_4\}\}$$

Independent Rules and the Combine Function

NOTE that disjunction is different from mutually independent Rules.

With two separate Rules, R1 and R2.

$$(C, P_2, V_2, CF_2) \xrightarrow{CF_{R1}} (C, P_1, V_1, CF_1)$$
$$(C, P_3, V_3, CF_3) \text{ AND } (C, P_4, V_4, CF_4) \xrightarrow{CF_{R2}} (C, P_1, V_1, CF_1)$$

$$CF_1 = \text{Combine}(CF_{R1} \cdot CF_2, CF_{R2} \cdot \min\{CF_3, CF_4\})$$

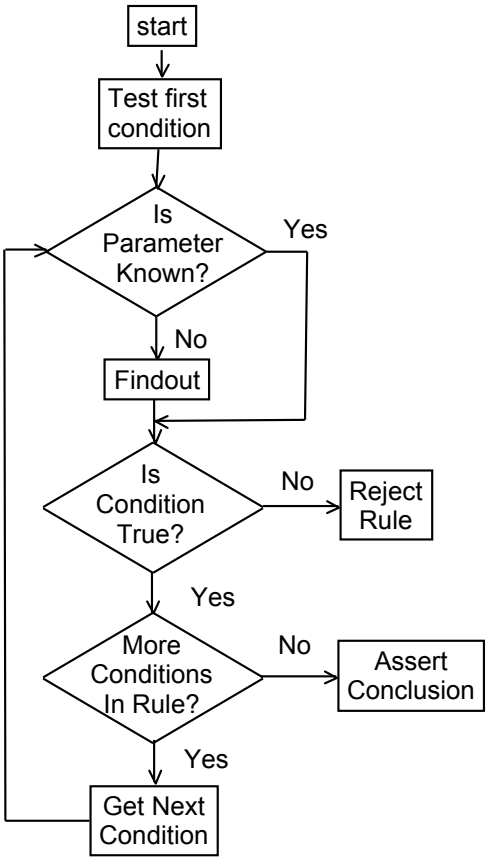
The confidence from independent rules is interpreted with the function Combine

$$Combine(CF_1, CF_2) = \begin{cases} CF_1 + CF_2 - CF_1 \cdot CF_2 & \text{if } CF_1 \geq 0 \text{ AND } CF_2 \geq 0 \\ -Combine(-CF_1, -CF_2) & \text{if } CF_1 < 0 \text{ AND } CF_2 < 0 \\ \dfrac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)} & \text{if } CF_1 \cdot CF_2 < 0 \end{cases}$$

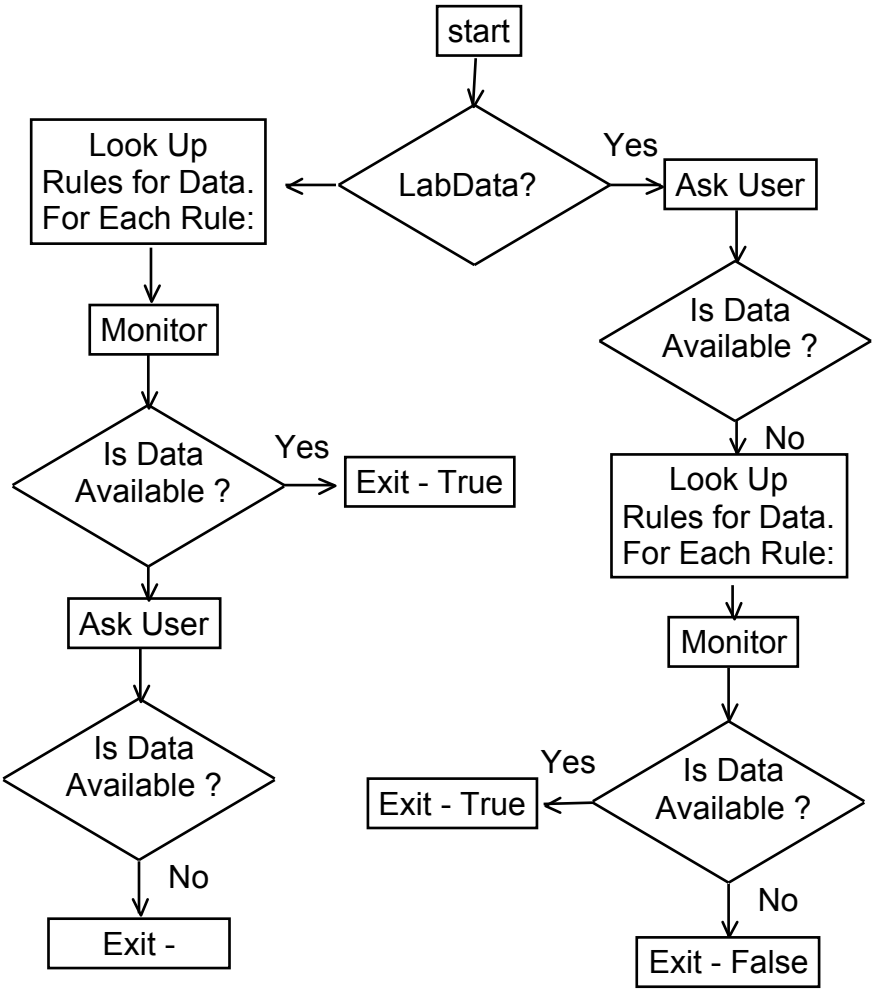
Co-routines: Findout and Monitor

Mycin inference engine interprets the domain knowledge using two recursive Coroutines: Findout and Monitor

MONITOR :



FINDOUT:



At any instant the user may ask

WHY? The system provide an interpretation of the trace of reasoning
HOW: The system provides the source for a fact.

Coupled with the extensive use of preprogrammed sentences, this made the system appear to be intelligent. However the reasoning was shallow.

The Mycin system demonstrated the importance of reasoning with uncertain facts. This triggered many groups to investigate and propose alternatives to reasoning with uncertainty.

Why did expert systems fail?

While Expert Systems technologies provided useful solutions for many applications, there was a fundamental problem: Hand-crafting a knowledge base is generally a very expensive and difficult process.

The Media Hype in AI promised a fundamental revolution in computing and the emergence of a practical technology for intelligent systems. Interest gradually waned as most projects failed to fulfill the widely inflated expectations, and generated widespread disillusionment.

At the same time, useful techniques from expert systems were adopted as commonly software engineering techniques and became standard in software engineering. For example many of the early concepts in Object Oriented Programming were developed for expert systems and then assimilated in software engineering.

Other techniques fell out of favor. In particular, the process of encoding expert knowledge was found to be prohibitively expensive and generally unreliable. Efforts to provide automatic methods for acquiring symbolic knowledge for problem solving in the area of Symbolic Machine Learning were notoriously unreliable. Efforts at machine learning using data were generally severely limited by a lack of available computer readable data.

None-the-less, concepts and methods developed in this period for knowledge representation provided a new foundation for cognitive science and for ergonomics. These have lead to a much deeper understanding of human intelligence, and now provides a sort of roadmap for developing machine intelligence.