This course will discuss the key concepts and techniques behind the Knowledge-Based Systems that are the focus of such wide interest today. These systems are at the applied edge of research in Artificial Intelligence. To put them in perspective this course will take a short historical tour through the AI field and its related subtopics. This tour will focus on underlying themes, with examples drawn from representative systems.

The key factors that underly knowledge-based systems are knowledge acquisition, knowledge representation, and the application of large bodies of knowledge to the particular problem domain in which the knowledge-based system operates. Dr. Smith will discuss a number of formalisms for knowledge representation and inference that have been developed to aid in this process. Once again, this will be illustrated with examples drawn from existing systems.

The course will conclude with a discussion of the pragmatics of actually building a knowledge-based system. This will include: (1) suggestions for selecting problems that are amenable to the knowledge-based system approach, and (2) a description of the characteristics of software tools and high-level programming environments that are useful, and for most purposes necessary, for the construction of a practical knowledge-based system.

Reid G. Smith is the program leader for Expert Geology Systems at Schlumberger-Doll Research, Ridgefield, Connecticut, where he has been since 1982. His current research is on expert systems which explain failures and developjustifications for the information in their knowledge bases. He has been involved in the Dipmeter Advisor project and in the development of tools for expert system construction. He has also worked in knowledge-based systems for passive sonar interpretation for the Canadian Defense Research Establishment Atlantic. Dr. Smith received his B. Eng. and M. Eng. in electrical engineering at Carleton University before doing a Ph.D. at Stanford. He is the author of “A Framework for Distributed Problem Solving” (UMI Press).
PROSPECTUS

Artificial Intelligence
  Underlying Themes

Knowledge-Based Systems
  Underlying Themes
  Example: MYCIN
  Example: DIPMETER ADVISOR
  Assessment and Outlook

<break>

Representation and Reasoning
  Rules / Chaining
  Structured Objects / Inheritance
    Procedural Attachment

Pragmatics

Perspective
Artificial Intelligence

Goals:
To construct computer programs that perform at high levels of competence in cognitive tasks

Knowledge-Based Systems
To understand and develop computational models of human intelligence

Cognitive Science

As Experimental Computer Science: Side Effects
Time-sharing
Sophisticated Programming Environments
Exploratory Programming
Personal Machines
Local Area Network Processing
FOUR AREAS OF COMPUTING

<table>
<thead>
<tr>
<th>Type of Processing</th>
<th>NUMERIC</th>
<th>SYMBOLIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALGORITHMIC</td>
<td>traditional scientific calculations</td>
<td>data processing</td>
</tr>
<tr>
<td>HEURISTIC</td>
<td>computation-intensive application with heuristic control (manipulators)</td>
<td>Artificial Intelligence</td>
</tr>
</tbody>
</table>
WHAT IS A KNOWLEDGE-BASED SYSTEM?

Symbolic: It incorporates knowledge that is symbolic [as well as numeric].

Heuristic: It reasons with judgmental, imprecise, and qualitative knowledge as well as with formal knowledge of established theories.

Transparent: Its knowledge is simply and explicitly represented in terms familiar to specialists, and is separate from its inference procedures. It provides explanations of its line of reasoning and answers to queries about its knowledge.

Flexible: It is incrementally refinable and extensible. More details can be specified to refine its performance; more concepts and links among concepts can be specified to broaden its range of applicability.

It is an expert system if it provides expert-level solutions.

The power lies in task-specific knowledge.
Generality and Power

AI Paradigm Shift: circa 1970-72
SOME BASIC PRESCRIPTIONS

Don’t tell the program what to do, tell it what to know.

Keep the knowledge in the knowledge base.

Choose a representation that is as high-level transparent as possible.

The task is elucidating and debugging knowledge, not writing and debugging a program.

... Knowledge Programming
Basic Knowledge-Based System Architecture

Major Architectural Lesson
EXAMPLE: MYCIN

MAIN ISSUES

IN THE KNOWLEDGE LIES THE POWER

HOW TO DEAL WITH INCOMPLETE, INEXACT DATA

HOW TO DEAL WITH INEXACT INFERENCE

THE UTILITY AND POWER OF SIMPLE MECHANISMS

THE IMPORTANCE OF SYSTEMS THAT ACCOMMODATE THE USER
1) Patient's name: (first-last)
   ** C.R.

2) Sex:
   ** MALE

3) Age:
   ** 52

4) Have you been able to obtain positive microbiological information about a possible infection of C.R.?
   ** YES

5) What is the infection?
   ** ENDARTERITIS
The first significant organism from this blood culture (CULTURE-1) will be referred to as:

--------ORGANISM-1--------

9) Enter the identity of ORGANISM-1:
   ** UNKNOWN
10) Is ORGANISM-1 a rod or coccus (etc.)?:
    ** ROD
11) The gram stain of ORGANISM-1:
    ** GRAMNEG
12) Have there been POSITIVE cultures yielding organisms about which you will NOT be seeking advice?
    ** YES
14) Is the patient's illness with ORGANISM-1 a hospital-acquired infection?

** WHY

[i.e., WHY is it important to determine whether or not the infection with ORGANISM-1 was acquired while the patient was hospitalized?]

[2.0] This will aid in determining the identity of ORGANISM-1. It has already been established that
[2.1] the morphology of ORGANISM-1 is rod
[2.2] the gram stain of ORGANISM-1 is gramneg
[2.3] the aerobicity of ORGANISM-1 is facul

Therefore, if
[2.4] the infection with ORGANISM-1 was acquired while the patient was hospitalized

then

there is weakly suggestive evidence (.2) that the identity of ORGANISM-1 is pseudomonas

[RULE050]
15) Enter the identity of ORGANISM-2:
** SALMONELLA (.4) E.COLI (.3)
INFECTION-1 is ENDARTERITIS with BACTEREMIA

<item 1> E.COLI [ORGANISM-1]
<item 2> SALMONELLA (species unknown) [ORGANISM-1]
<item 3> KLEBSIELLA-PNEUMONIAE [ORGANISM-1]
<item 4> PSEUDOMONAS-AERUGINOSA [ORGANISM-1]
<item 5> ENTEROBACTER [ORGANISM-1]
<item 6> PROTEUS-NON-MIRABILIS [ORGANISM-1]
[Rec 1 ] My preferred therapy recommendation is as follows:

In order to cover for Items <1 3 4 5 6>:
Give: GENTAMICIN
Dose: 128 mg (1.7mg/kg) q8h IV (or IM) for 10 days
Comments: Modify dose in renal failure

In order to cover for Item <2>:
Give: CHLORAMPHENICOL
Dose: 563 mg (7.5 mg/kg) q6h for 14 days
Comments: Monitor patient's white count

Do you wish to see the next choice therapy?
**  NO
Rule 27:
If 1) the gram stain of the organism is gram negative, and
   2) the morphology of the organism is rod, and
   3) the aerobicity of the organism is anaerobic,
Then: There is suggestive evidence (7) that the identity of the organism is Bacteroides
Rule 27
INEXACT INference Desiderata

Commutative

Asymptotic

Balancing of + and – Evidence

CF range: {-1 ... 0 ... 1}

\[
CF_n = \begin{cases} 
CF_p + CF(1 - CF_p) & CF_p, \quad CF > 0 \\
CF_p + CF(1 + CF_p) & CF_p, \quad CF < 0 \\
(CF_p + CF) / (1 - \text{min}\{|CF_p|,|CF|\}) & CF_pCF < 0 
\end{cases}
\]
** WHAT WAS THE IDENTITY OF ORGAISM-1 ?
= ORGANISM-1
The identity of ORGANISM-1 is e.coli (.51), salmonella (.45), klebsiella (.4), pseudomonas (.22), enterobacter (.21), proteus (.19), pseudomonas-aeruginosa (.08), serratia (.02)
** HOW DID YOU DECIDE THAT ORGANISM-1 WAS AN E.COLI?**

I used rule084 to conclude that the identity of ORGANISM-1 is e.coli. This gave a cumulative c.f. of (.51).

I used rule003 to conclude that the identity of ORGANISM-1 is e.coli. This gave a cumulative c.f. of (.43).
** WHY DIDN'T YOU THINK THAT ORGANISM-1 WAS NEISSERIA?

The following rules could have been used to determine that the identity of ORGANISM-1 is neisseria: 9 However, none of these succeeded in the context of ORGANISM-1. If you would like an explanation for why any of these rules failed, please enter their numbers:
** 9

Clause 2 of rule009 ["the morphology of the organism is coccus"] was already known to be false for ORGANISM-1, so the rule was never tried.
LESSONS: MYCIN

FLEXIBILITY [Refinement, Extension]

Inference Engine - Knowledge Base Separation

Simplicity

Modularity

USER ACCOMODATION

Consultant Role

Transparency

Interface
EXAMPLE:
DIPMETER ADVISOR

MAIN ISSUES

DEALING WITH SIGNALS

HOW TO COMBINE METHODS AND APPROACHES

INTEGRATION OF KBS INTO AN OVERALL SYSTEM

THE UTILITY AND POWER OF SIMPLE MECHANISMS

THE IMPORTANCE OF SYSTEMS THAT ACCOMMODATE THE USER

SOLVING THE PROBLEM, GOING COMMERCIAL
Gamma Ray

Dip Angle

0 50 100 0 10 20 30 40 50°

Uniform Structural Dip in Shale.

Downward Dip Increase Due to Drape.

Random dips in Reef Core Limestone.

Downward Decreasing Dips in Forereef Talus

Uniform Structural Dip in Pre-Reef Shale.
DIPMETER ADVISOR SYSTEM: SAMPLE RULE

IF

there exists a normal fault with class unknown, and
there exists a red pattern
with length < 50 ft.,
with bottom above the top of the fault,
with azimuth perpendicular to the fault strike

THEN

the fault is a late fault
with direction to downthrown block
equal to the azimuth of the red pattern

[Diagram of a normal fault with a red pattern and a downthrown block]
DIPMETER ADVISOR SYSTEM: PHASES

- **GENERAL:**
  - Initial Examination
  - Validity Analysis
  - Lithology Analysis

- **STRUCTURAL DIP DETERMINATION & REMOVAL:**
  - Green Patterns
  - Structural Dip Analysis & Removal

- **STRUCTURAL (TECTONIC) FEATURE ANALYSIS:**
  - *Preliminary Structural Analysis*
  - Structural Patterns
  - *Final Structural Analysis*

- **STRATIGRAPHIC FEATURE ANALYSIS:**
  - *Depositional Environment Analysis*
  - Stratigraphic Patterns
  - *Stratigraphic Analysis*
LESSONS: DIPMETER ADVISOR

SYSTEM INTEGRATION: EMBEDDING
   Much more than IE + KB

SMOOTH AUGMENTATION OF HUMAN ABILITIES
   The Intelligent Assistant
   Interactive Control of Inference

RULESETS AND INDEXING BY TASK
   Understanding a Ruleset as a Unit

SIGNAL PROCESSING / SYMBOLIC PROCESSING

SOLVING THE PROBLEM

TECHNOLOGY TRANSFER

Impact on the Way Computing Is Viewed and Practiced
Embedding a Knowledge-Based System: An Intelligent Assistant

A user gets a number of advantages from using the system – one of which is symbolic inference.

In watching the system operate, an observer might never realize that any intelligence is involved.
Detailed Knowledge-Based System Architecture

- Interaction Processor
  - Data Entry
  - Explanation
  - Graphics

- Knowledge Base Editor
  - Debugging Aids

- Knowledge Base
  - Domain Facts
  - Rules
  - Procedures

- Inference Engine

- Blackboard
  - Current Problem Data
WHY BUILD A KNOWLEDGE-BASED SYSTEM?

To Decrease Cost or Increase Quality of Goods and Services

Magnify Availability of Expertise

provide expertise to less experienced personnel
avoid delays when expertise is needed
provide expertise in locations where it is not available

Fuse Different Sources of Knowledge

Encode Corporate Knowledge

provide consistency and availability over time

Automate Some Routine Decision-Making or Bookkeeping Tasks

Keep Records of Decisions and Actions

provide a reliable database for later analysis
What's a potential application like?

Is there no known algorithmic solution, or is the algorithmic solution too costly?

Is the domain well-bounded, tractable, non-trivial?

Does the domain require little common sense reasoning?

Are there non-trivial, useful subproblems or "easy" versions of the problem?

Are there recognized experts in the domain?

Does the task have a high payoff?

Does the task normally take less than a few hours (days)?

Does the task have a combinatorial nature?

Can the expertise be incrementally acquired?

Are data and case studies readily available?

Are domain experts readily available?
WHAT MAKES FOR A GOOD APPLICATION?

there are recognized experts
the experts are provably better than amateurs
there is general agreement about the knowledge
the commitment of an expert can be obtained
the task has a high payoff
the task takes an expert a few minutes to a few hours
the knowledge is primarily symbolic
the task has a combinatorial nature
the skill is (routinely) taught to neophytes
data and test cases are available
incremental progress is possible
the task requires no common sense
WHAT IS THE STATE OF THE ART?

Expert-level performance on narrow problems

Sufficient knowledge to solve important problems

Understandable, but limited explanation of line of reasoning

Natural human interface, both graphical and text, but with stylized language and limited vocabulary

Flexible knowledge bases

Requirement for an experienced "knowledge engineer"

Limited to one expert as the "knowledge czar"
IN THE KNOWLEDGE LIES THE POWER

BUT HOW DO WE GET THE KNOWLEDGE?

KNOWLEDGE BASE CONSTRUCTION

MACHINE LEARNING
Knowledge Acquisition

KE: Conceptualization
    Construction
    Refinement

KE: Using the framework
   vs
   Designing the framework
REPRESENTATION
AND REASONING

How to conceptualize and encode the knowledge

How to use the knowledge
DESIDERATA FOR GOOD REPRESENTATIONS

Good representations make the important things explicit.

They expose natural constraints, facilitating some class of computations.

They are complete. We can say all that needs to be said.

They are concise. We can say things efficiently.

They are transparent to us. We can understand what has been said.

They allow for incremental refinement and extension of the knowledge.

They facilitate computation. We can store and retrieve information quickly.

They suppress detail. We can keep rarely used information out of sight, but we can still get to it when necessary.

They are computable by an existing procedure.

_Theoretical Equivalence Is Different From Practical Equivalence_
<table>
<thead>
<tr>
<th>Representation</th>
<th>Typical Inferencing Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>Chaining</td>
</tr>
<tr>
<td>Structured Objects (Frames)</td>
<td>Inheritance</td>
</tr>
<tr>
<td></td>
<td>Procedural Attachment</td>
</tr>
</tbody>
</table>

**THEME:** Explicitly capture an increasing degree of structure in the domain knowledge.
RULES

Thinking in Rules
(Forward) Chaining
A Rule Interpreter
Backward Chaining (Subgoaling)
Adding New Rules

Conclusion: Rules are fine if you either cannot identify more specific knowledge structure in the domain, or if you do not need to in order to solve the problem at hand.
THINKING IN RULES

Situation / Action

if temp > 300°C then turn off boiler.

Premise / Conclusion

if stain is grampos then organism is strep.

Antecedent / Consequent

if x is a dog then x is an animal.
FORWARD CHAINING

if stain is grampos then organism is strep.

if stain is gramneg then organism is e.coli.

if organism is strep or bacteroides then penicillin is indicated.

if a drug is indicated and don't know whether allergic to the drug then ask whether allergic to the drug.

if a drug is indicated and not allergic to the drug then prescribe the drug.
A Rule Interpreter

MATCH \rightarrow RULE BASE \rightarrow EXECUTE

FACT BASE

\{ fact \\
rule \}

new rule

new fact
R1: A rule-based program that configures VAX and PDP-11 computer systems. [3300 rules; used for 20,000 orders (Jan 84)]

If the most current activity 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 the number of devices that each massbus should support is known, and there is a massbus that has been assigned at least one disk drive and that should support additional disk drives, and the type of cable needed to connect the disk drive to the previous device on the massbus is known then assign the disk drive to the massbus
Q: What about prescribing penicillin?

if stain is grampos then organism is strep.

if stain is gramneg then organism is e.coli.

if organism is strep or bacteroides then penicillin is indicated.

if a drug is indicated and don't know whether allergic to the drug then ask whether allergic to the drug.

if a drug is indicated and not allergic to the drug then prescribe the drug.

A: Prescribe penicillin if the stain is grampos and patient is not allergic to penicillin.
ADDING NEW RULES

Myth: "You just add more rules."

Reality:

- No Chaining
- Infinite Chaining
- Introducing Contradictions
- Modifying Existing Rules
- Rulesets and Structure

Conclusions:

- It's still programming
- KBS design requires careful choice of abstraction levels in the task domain and rules to move between levels
INTRODUCING CONTRADICTIONS

New Rule:

N) \text{if stain is gramneg and shape is rod then organism is pseudomonas.}

Existing Rules:

1) \text{if organism is pseudomonas then organism is not e.coli.}

2) \text{if stain is gramneg then organism is e.coli or bacteroides.}

3) \text{if shape is rod then organism is not bacteroides.}

Contradiction:

\begin{align*}
\text{gram(neg)} \\
\text{shape(rod)} \\
\hline
N \text{id(pseudomonas)} \\
2 \text{id(e.coli) or id(bacteroides)} \\
1 \text{not id(e.coli)} \\
3 \text{not id(bacteroides)}
\end{align*}
STRUCTURE OBJECTS

(Frames)

Thinking in Objects
Inheritance
Messages and Procedural Attachment
Event Handlers
Things That Can Go Wrong

**Conclusion:** Objects are good for capturing static structural knowledge. They are also an excellent organizational paradigm for programming in general.
THINKING IN OBJECTS

An object is an encapsulation of data and procedures relevant to a concept.

From Simulation
In building complex software systems, there is a distinct advantage to constructing a computational world that is an image of the physical world in which the systems operate.

From AI & Cognitive Psychology
A frame is a data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed.

Marvin Minsky (1974)

Capturing Natural Abstractions
Knowledge Acquisition
Knowledge Base Maintenance
<table>
<thead>
<tr>
<th>GrowthFault1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strike</td>
</tr>
<tr>
<td>TimeOfFault</td>
</tr>
<tr>
<td>Slip</td>
</tr>
</tbody>
</table>

Slots
PROPERTY INHERITANCE

Taxonomic Hierarchies
  Class & Instance

Factoring
  Avoiding Redundancy
  Conceptual Clarity
  Ease of Extension/Refinement
MESSAGES & PROCEDURAL ATTACHMENT

Uniform Invocation Method

Polymorphism: Increasing Transparency

Modularity: Avoiding Assumptions
Loops Gauges
EVENT HANDLERS

procedure UpdateMeterDisplay (Setting)
  Send(ClearDisplay)
  Send(SetDisplay Setting)
end
HangingWallBlock {DownthrownBlock}:
UpperDistortionRegion:
BrecciaRegion {CrushedZone}:
FaultPlane:
LowerDistortionRegion:
FootWallBlock {UpthrownBlock}:
Strike:
FaultAngle {Hade}:
Direction to DownthrownBlock:
Slip:
Throw:
TimeOfFaulting:
Draw:
Instantiate: InstantiateFault
Detect: (RuleNFR1 RuleNFR3 RuleNFR4 RuleNFR5 RuleNFR7)
Specialize: (RuleNFR6 RuleNFR9 RuleNFR10 RuleNFR11 RuleNFR12)
THINGS THAT CAN GO WRONG

Smashing the semantics

\textit{there are many kinds of hierarchies}

Not using the paradigm uniformly

Efficiency: What is an object?
HIERARCHY OF GEOLOGIC AGE

- EON
  - AZOIC
  - CRYPTOZOIC
  - PHANEROZOIC

- ERA
  - PRECAMBRIAN
  - PALEOZOIC
  - MESOZOIC
  - CENOZOIC

- GEOLOGIC-AGE
  - PERIOD
    - CAMBRIAN
    - ORDOVICIAN
    - SILURIAN
    - DEVONIAN
    - CARBONIFEROUS
    - PERMIAN
    - TRIASSIC
    - JURASSIC
    - CRETACEOUS
    - TERTIARY
    - QUATERNARY
      - MISSISSIPPIAN
      - PENNSYLVANIAN
  - EPOCH
    - PALEOCENE
    - EOCENE
    - OLIGOCENE
    - MIocene
    - PLIOCENE
    - PLEISTOCENE
    - HOLOCENE

- TOTAL-HISTORY
“PARTS” OF GEOLOGIC AGE

TOTAL-HISTORY

CRYPTOZOIC

AZOIC

PALEOZOIC

PHANEROZOIC

MESOZOIC

CRETACEOUS

JURASSIC

TRIASSIC

PERMIAN

CARBONIFEROUS

DEVONIAN

SILURIAN

ORDOVICIAN

CAMBRIAN

PALEOCENE

EOCENE

OLIGOCENE

MIOCENE

PLIOCENE

HOLOCENE

QUATERNARY

PLEISTOCENE

CENOZOIC
Question: Which shall I use ... 

Rules? 
or Structured Objects? 
or Procedures?

[ANALOGY: Cobol or Fortran or Snobol?]

Answer:

Think about the knowledge structures first. Representations are just a tool. Any tool can be used badly.

State of the art systems orchestrate a variety of representations.
PRAGMATICS

Methodology

Tools/Computing Environments

Personnel Training

Costs

Pitfalls

Excessive aspirations

Inadequate Resources

Poor Problem

Technology Transfer and Sociology
Methodology

Identify the problem.

Identify concepts and levels of abstraction in the problem (e.g., parameter, segment, syllable, word, word-sequence, phrase).

What Kind of Knowledge Is Involved?

Identify knowledge sources for traversing links between levels.

How Is The Knowledge Used?

Select or devise an appropriate representation.

How Should The Knowledge Be Represented?

Expose constraints or regularities.

Create particular procedures (for using the constraints, given the representation).

Verify via experiments.
HOW DO SOME SYSTEMS GET BUILT SO QUICKLY?

The tip of the iceberg

RULES

Graphics

Consultation

Explanation

Rule Acquisition

Knowledge Base Facilities

Inference Engine
EXPLORATORY PROGRAMMING

Conscious intertwining of design and implementation for construction of large, complex software systems under uncertain specifications

Uncertain Specifications Arise Because...

we don't understand the problem \textit{a priori} (AI ...)

the design space is too large to explore without extensive experimentation (interactive graphics ...)

How To Deal With This:

Minimize and defer constraints placed on the programmer. Place more of the burden for managing complexity on the programming environment.
Sources of Power

*freedom of expression*

variety of data types and abstractions

*late binding: deferring commitments*

automatic storage management
dynamic typing of variables
dynamic binding of procedures

*procedures as data*

program manipulation subsystems
interpreters for special purpose languages

*integrated tools*

interpreter and incremental compiler
data inspector
language-sensitive editor
optimizing compiler
static and dynamic analysers

*personal workstation*

interactive graphics
LAN connections (heterogeneity)
Feasibility Demonstration

"this has been implemented"
chewing gum and paper clips

Prototype

experimental users
user interface
change of venue?

Commercial Product

solving enough of the problem to be interesting and useful
integration into an overall system
accuracy of knowledge base
potential of framework for development
efficiency, modularity
user interface, documentation
COSTS OF BUILDING KNOWLEDGE-BASED SYSTEMS

ASSESSMENT
a few days or weeks

PROTOTYPING
1-2 man-years knowledge engineer
0.5 man-years domain specialist

DEVELOPMENT
2-5 man-years knowledge engineer
half-time from domain specialist

FIELDING
software engineering
SOME HARD PROBLEMS

Inexact Reasoning
Knowledge Engineering
Learning by Induction
Default Reasoning
Common-Sense Knowledge
Strategies
Qualitative Reasoning
Huge Knowledge Bases
Self-Awareness
User Modeling
## THE CURRENT STATE OF SOME HARD PROBLEMS

<table>
<thead>
<tr>
<th></th>
<th>PRACTICE</th>
<th>THEORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inexact Reasoning</td>
<td>CF Model</td>
<td>Almost OK</td>
</tr>
<tr>
<td>Knowledge Engineering</td>
<td>An Art</td>
<td>Unexplored</td>
</tr>
<tr>
<td>Learning by Induction</td>
<td>Hand-Crafted</td>
<td>Over-Developed</td>
</tr>
<tr>
<td>Default Reasoning</td>
<td>Inheritance</td>
<td>Emerging</td>
</tr>
<tr>
<td>Common-Sense Knowledge</td>
<td>Add Items To KB</td>
<td>Puzzling</td>
</tr>
<tr>
<td>Strategies</td>
<td>Meta-Level Knowledge</td>
<td>Not Well Explored</td>
</tr>
<tr>
<td></td>
<td>PRACTICE</td>
<td>THEORY</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Qualitative Reasoning</td>
<td>Omit Details</td>
<td>Not Well Explored</td>
</tr>
<tr>
<td>Huge Knowledge Bases</td>
<td>Internist AI &amp; DB</td>
<td>Not Well Explored</td>
</tr>
<tr>
<td>Self-Awareness</td>
<td>Meta-Level Knowledge</td>
<td>Not Well Explored</td>
</tr>
<tr>
<td>User Modeling</td>
<td>Overlay Models</td>
<td>Not Well Explored</td>
</tr>
</tbody>
</table>
**Scenario:** We don't know how to solve this problem ... maybe we should build an expert system ...

**Maxim:** It's very difficult to build knowledge-based systems without benefit of knowledge, or expert systems without benefit of expertise.

**Scenario:** Fred isn't doing anything right now, and he wrote a computer program once. Let's have him spend a couple of months and write an expert system. That will enable us to find out whether there's anything more than hype.

**Maxim:** Acquire or train people.
Physical Symbol Systems

A broad class of systems capable of having and manipulating symbols, yet realizable in the physical universe.

*Allen Newell and Herbert Simon*

Symbol: A physical pattern.

Symbol Structure: A number of instances (or tokens) of symbols related in some physical way (such as one token being next to another).

Designation: A relation between a symbol and the entities (e.g., operators, symbol structures) it symbolizes.

Interpretation: The act of accepting an input that designates a process and then performing that process.

Physical Symbol System components: input, output, memory, control (that performs interpretation), and a set of operators capable of assigning symbols (creating designations), and copying (creating new symbols and symbol structures), reading (obtaining the symbols that comprise a symbol structure) and writing symbol structures. A Physical Symbol System is a *Universal Machine* that produces through time an evolving collection of symbol structures.
The Physical Symbol System Hypothesis

The necessary and sufficient condition for a physical system to exhibit general intelligent action is that it be a physical symbol system.

Allen Newell and Herbert Simon

Necessary means that any physical system that exhibits general intelligence will be an instance of a physical symbol system.

Sufficient means that any physical symbol system can be organized further to exhibit general intelligent action.

General intelligent action means the same scope of action seen in human action; that in real situations behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some physical limits.

This hypothesis sets the terms on which we search for a scientific theory of mind. What we seek are the further specifications of physical symbol systems that constitute the human mind or that constitute systems of powerful and efficient intelligence.
The Problem Space Hypothesis

The fundamental organizational unit of all human goal-oriented symbolic activity is the problem space.

*Allen Newell*

*Problem Space:* states; operators

*Problem:* initial states; goal states; path constraints

The only way to solve a problem in a problem space is to search in the space. (Problem-specific knowledge may allow very efficient search.)
PRAGMATICS

● Why Build a Knowledge-Based System?
● What Makes for a Good Application?
● What Is the State of the Art?

● Methodology
● Tools and Computing Environments
● The Development Team
● Costs / War Stories/ Hard Problems

● Pitfalls
  Excessive Aspirations
  Inadequate Resources
  Poor Problem Selection
  Technology Transfer & Sociology...
THE DEVELOPMENT TEAM

- DOMAIN EXPERTISE
- KBS TOOL DESIGN
- KNOWLEDGE ENGINEERING
- PROGRAMMING SUPPORT
- EXPERIMENTATION
- ENGINEERING
Incremental Development

Performance

- Tool Development
- Knowledge Expansion

Time
OBSERVATIONS ON THE TRADITIONAL WISDOM

- THROWAWAY CODE vs PROGRESSIVE RELEASE
- INCREMENTAL DEVELOPMENT including specialists
- DIFFICULTY OF CLEAR TASK DEFINITION
  - Evolving Performance Changes Definition
  - Contingent Definition
- EARLY CODING
- HUMILITY ABOUT PERSONAL DOMAIN EXPERTISE
- FLEXIBLE INTERFACES
- DOMAIN SPECIALISTS: SINGLE OR MULTIPLE
  - Difficulty of Understanding Multiple Views
  - Missing an Important Alternate View
  - Observing Interactions - Changes in Strategy
- SEDUCTION BY A SIMPLE FORMALISM
- STIMULATION OF SPECIALISTS
TECHNOLOGY TRANSFER

- The Forward Pass
- Prototypes Not Enough
- OnSite Engineering & Field Involvement
- Domain Expertise and Commitment
- Solving a Real Problem
- Flexibility
- Non-Impact Technology (e.g., WorkStations)
- Value-Added Systems
  smooth extension of human capabilities
THEMES

I. The key to success is Domain Knowledge, not clever programming.

**Corollary:** If you have a good handle on the domain knowledge, the right problem organization will follow.
II. Levels of Understanding

EMPIRICAL ASSOCIATIONS

UNDERLYING STRUCTURE AND FUNCTION

Maxim: What you can do depends on what level of understanding you have.
For Further Information


