CS229b: A Survey of AI
Classnotes for Winter 84-85

by

Devika Subramanian

Department of Computer Science

Stanford University
Stanford, CA 94305

RNEGIE-MELLON UNIVERSITY
BURGH, PENNSYLVANIA 15213

first draft April 1985

CS229b: A Survey of AI

Classnotes for Winter 84-85

by

Devika Subramanian

lese are the compiled classnotes for the course CS229b offered in Winter 1985. This iirse was an intensive 10 week survey intended as preparation for the 1984-85 qualifying Etmination in Artificial Intelligence at Stanford University.

Contents

Preface

This is a compilation of the classnotes for the course CS229b offered in Winter 1985. This course was an intensive 10 week survey course in Artificial Intelligence intended as preparation for the 1984-85 qualifying examination in Artificial Intelligence at Stanford University. This document is best used as a companion volume to the annotated reading list (STAN-CS-85-1093) that was used for this class.

Most class sessions record the discussion that was held on the topic for that day. The questions for discussion were prepared by the author. The proceedings of the class have been edited and substantially extended by the author. The guest sessions have been edited by the guests themselves. We thank Prof. Buchanan, Dr. Grosz, Dr. Stickel, Prof. Nilsson and Prof. Rosenbloom for editing the transcripts of their discussions. We thank all the guest speakers who participated in the discussion forums. We also thank the members of the class (acknowledged overleaf) for contributing to the discussions in class and for helping in transcribing the minutes of the class session. We thank our faculty sponsor Prof. Buchanan for his guidance and encouragement. Finally, we express our thanks to Prof. Nilsson with whose help this document appears as a CS Department technical note.

Class Participants

Faculty Sponsor: Prof. Bruce Buchanan
Teaching Assistant: Devika Subramanian
Students:
Bill Erikson
Andy Golding
Ramsey Haddad
Keith Hall
Haym Hirsh
Mary Holstege
Kathleen Kells
Majid Khorram
Charlie Koo
Kim McCaU
Kathy Morris
Jeff Naughton
Guests:
Dr. Ronald J. Brachman
Prof. Bruce Buchanan
Prof. Michael R. Genesereth
Dr. Barbara Grosz
Dr. Jitendra Malik
Prof. Nils Nilsson
Prof. Paul Rosenbloom
Dr. Stan Rosenschein
Dr. Mark Stickel

Notetakers for CS229b

7 January	Introduction	Devika Subramanian
9 January	Search and Heuristics	Mary Holstege and Kim McCall
11 January	Search and Heuristics	Ramsey Haddad and Kathy Morris
14 January	Knowledge Representation	Andy Golding and Haym Hirsh
16 January	Guest session on Knowledge Representation	Devika Subramanian
18 January	Knowledge Representation	Mary Holstege and Kim McCall
21 January	Planning, Problem solving and AP	Ramsey Haddad
23 January	Planning, Problem solving and AP	Majid Khorram and Jeff Naughton
25 January	Guest session on Planning	Keith Hall
28 January	Deduction and Inference	Mary Holstege
30 January	Deduction and Inference	Haym Hirsh and Bill Erikson
1 February	Guest session on Deduction and Inference	Andy Golding and Ramsey Haddad
4 February	Expert System Principles	Kathleen Kells and Kim McCall
6 February	Expert System Principles	
8 February	Guest session on Expert System Principles	Devika Subramanian
11 February	Learning	Mary Holstege
13 February	Learning	Kathleen Kells and Charlie Koo
15 February	Learning	Andy Golding and Devika Subraman
20 February	Vision and Robotics	Ramsey Haddad
22 February	Vision and Robotics	Devika Subramanian
25 February	Natural Language	Mary Holstege
27 February	Natural Language	Kathleen Kells and Bill Erikson
1 March	Guest session on Natural Language	Devika Subramanian
4 March	AI applications	Bill Erikson and Charlie Koo
6 March	AI applications	Ramsey Haddad
8 March	Guest session on AI applications	Devika Subramanian
11 March	Guest session on Advanced topics	Devika Subramanian
13 March	Advanced topics	Bill Erikson
15 March	Advanced topics	Devika Subramanian

Chapter 1

CS229b: A Description

1.1 Goals of the Course

The primary aim is to prepare doctoral students in the Computer Science department for the qualifying examination in Artificial Intelligence. This course will attempt to present an analytical survey of the literature in Artificial Intelligence. It will be an extension of the traditional *qual study group* in that, discussion and question answering will be the main modes of dissemination of knowledge. It will be run by a Teaching Assistant who in conjunction with the Faculty Sponsor of the course will

- Prepare an annotated study list for the qualifying exam.
- Help define the order in which to read the material and also what to look for in the reading.
- Prepare *qual style* questions on the reading material.
- Lead critical discussion on the readings and encourage active participation in question answer sessions.
- Arrange for guest speakers who will provide a unifying and comprehensive picture of work in the different subareas of Artificial Intelligence.
- Provide a forum to help organize topics for future research.

Notes

The primary objective of this course is to help students preparing for the Artificial Intelligence qualifying exam, by going through the readings collectively and stimulating discussions on various issues in AI. At the end of this course we hope to be able to present a unified account of the work in AI which is deeper than the one obtained through CS223 and more coherent than the treatment in CS224. The lack of an intermediate level course in AI has been bemoaned by several students and faculty members and this course is an attempt to fill this gap. We however emphasize that the qual is the primary motivation and use this to guide the depth and breadth of our discussion oriented course.

1.2 Administrative details

1.2.1 Faculty Sponsor

The faculty sponsor for this course is Prof Bruce Buchanan.

1.2.2 Class composition policy

As a matter of policy, this course is open to doctoral students intending to take the AI qual. Other interested students who wish to attend this course should obtain the permission of the TA or the instructor. This policy is to ensure some uniformity in the objectives of those who take this course and we hope that this will lead to more productive discussions.

1.2.3 Prerequisites

We will assume the following prerequisite : a familiarity with Artificial Intelligence as defined by the syllabus for the AI section of the Comprehensive Examination or CS223 or equivalent.

1.2.4 Grading

Is Pass/Fail preferably. Students who need a letter grade will have to undertake a substantial project - like updating some parts of the AI handbook. Please see the TA or the instructor about this.

1.3 Course Format

1.3.1 Discussion format

This course will be discussion oriented. As the previous version of this course (offered Winter 83-84) testified, discussion is invaluable for learning the material in sufficient depth and promoting ease and versatility of understanding. We continue the tradition in this version of the course too.

1.3.2 Number of sessions per week

The course meets three times a week (MWF) for 1 1/2 to 2 hours each. The topic for discussion during the week is made available in advance (at the beginning of the term). The Monday session will be devoted to an internal discussion of the topic monitored by the TA. The Wednesday session will be a question answering session conducted in rounds by the TA or the faculty sponsor. The Friday session will be a discussion session with the guest speaker.

This arrangement is chosen over the one adopted last year - two 2 1/2 hour sessions per week. These sessions were too long to be productive and there was no time for question answer sessions, which are vital from the point of view of taking the oral exam.

1.3.3 Reading list

The reading list for the course will be prepared by the TA with the help of the faculty sponsor. This will be an annotated and graded set of required and recommended reading for the qualifying examination. All reading for a given week will be made available, at least the week before the topic will be discussed. Copies of the reading material will be put on reserve in the Math/CS library.

1.4 Course Details

1.4.1 Organization of reading material

Here we are primarily limited by time constraints. This is a 10 week course and we have chunked work in AI into 10 topics as follows:

- Search and Heuristics

- Knowledge Representation
- Planning, Problem Solving, Automatic programming
- Deduction and Inference
- Expert system principles
- Natural language
- Learning
- Vision, Robotics, Speech understanding
- Expert system applications in Science,Medicine and Education
- Advanced Topics

We will attempt to cover the following during the course of the discussions :

- Directions for AI, Critiques of AI's ambitions, History of AI
- Architectures for AI and AI programming languages (under Knowledge Rep.)
- Cognitive models (under planning and problem solving)
- Game playing (under search)
- User models and explanations (expert systems and AI applications)

Notes

The first four topics in the core list form a 'theoretical' core for AI which set the stage for understanding issues in the subsequent 6 topics. We discuss issues like belief modelling and formalizing multi-agent plans, reflective architectures, speech act theory, epistemological issues in AI, non-monotonic reasoning and truth maintenance under advanced (and current) topics in AI. We have clumped Vision, Robotics and Speech understanding together because they are all signal to symbol transformation problems. We discuss specific expert systems under Expert system applications after we have covered the principles on which they are based. The auxiliary topics are important for a well rounded understanding of AI and we will address them during the discussions on the 10 core topics as indicated.

1.4.2 Question Answer sessions and question preparation

As stressed before, question answer sessions are very important for the qual. This gives the students the opportunity to become comfortable with the format of the qual. They will be conducted once a week on the topic designated for the week. Some questions will be handed to all students at the beginning of the term. We will use this list as a starting point for the question answer session, students can hand in questions before class, or else in class.

1.4.3 Guest speakers

For the topics chosen above, we will invite the following guest speakers.

Knowledge Representation	:	M. Genesereth and R. Brachman
Planning and Problem Solving, AP	:	Stan Rosenschein
Deduction and Inference	:	Mark Stickel
Expert system principles	:	Bruce Buchanan
Natural language	:	Barbara Grosz
Learning	:	Paul Rosenbloom
Vision, Robotics, SU	:	Tom Binford
Expert system applications	:	Bruce Buchanan
Advanced topics	:	Nils Nilsson

1.4.4 Demos of existing programs

The main idea behind this is to let the students have a feel for the typical characteristics of an AI program. It will also give them an opportunity to get some hands-on experience on programs that they read about in texts.

1.4.5 Publication summaries

A list of the most important publications that an AI researcher should know about has been compiled. This is an annotated list with short characterizations of the material to be found in each publication. A partial list will be handed to the students at the beginning of the term. The class will work as a team in producing a complete list.

1.4.6 Automatic Qual Question Generator

At the end of the term, students will turn in a design of a system that will generate qual questions in AL This will help in the understanding of what the important issues in AI are, as well as present an opportunity for the application of ideas and methods covered in this course.

Chapter 2

Search and Heuristics I

2.1 Review of Prof. Buchanan's suggestion from last session

The things to concentrate on for each topic are

- Literature survey - main ideas, chronology, names of people and programs
- Techniques and tools
- Concepts and definitions
- Outstanding problems

2.2 Questions on Search

1. Why is **search considered a part of** AI?
    - Lots of AI programs rely heavily on search,
    - It has been analyzed fairly deeply and well understood formally, thus adds class to an otherwise fuzzy subject !
    - It provides a framework onto which we hang heuristics, which are the real essence ofAL
    - It is a natural way of casting the problem of problem solving (the state space formalism) and this also a natural way to think about reasoning in general. Some

of us contended that this view left out much of natural language understanding and vision.

- Newell and Simon's PSS hypothesis emphasizes the primacy of search in AI. This is the notion that all human thought is problem solving which is heuristic search. [Aside : Claim : natural language does not fit into this paradigm. Counterclaim : Not quite so, more like – the work that gets done in natural language is that which can be made to fit the problem solving paradigm. Whether this is adequate for handling natural language is a separate issue].
- It has met with considerable success and is an effective way of casting many problems (cf. Generalization as search : Mitchell)

2. **What do we really need to know about search?**
   - the basic ideas in search algorithms
   - search strategies used by some important programs
3. **Why has interest in search died out more recently? Is there anything left to do in search?**
   - There must be, since there is a section for search included in the IJCAI proceedings.
   - In 1983 the AI journal ran a special issue on search and heuristics. A short summary of the articles there is to be found in this week's reading list. The current focus seems to be shifting from how to search to finding the right space to search (formulation of the problem in an appropriate space).
   - In Nilsson's view, there is no more work to be done here.
   - Analyzing the knowledge/search tradeoff. Judea Pearl has made an interesting start in this direction (cf the article in the recommended reading in this week's reading list).
   - No good stochastic search techniques yet.

Details of some search algorithms

4. **Winston says that best first search will not necessarily lead to the best solution (the shortest path to the goal). Why not?**

Winston's version of the best first search seeks to minimize the cost from the current position in the search to the goal, but ignores the cost already expended in reaching each of the nodes already expanded. (This makes it useful for solving problems that are solved completely on the fly, such as symbolic integration, but not for planning where there is a difference between the cost of discovering a solution and the cost of employing that solution.)

5. **What is best first search?**
Always expand the most promising node found so far, stopping when the most promising node is a goal node. But what does promising mean? Nilsson gives two different definitions.

   - smallest estimate of remaining cost
   - smallest sum of cost so far and the estimate of the remaining cost

   Pages 76-84 of Nilsson's text contains a definitive and valuable discussion of admissibility and monotonicity.

6. **Claim in Winston : Resolution is mainly backwards reasoning**
Since in general, the branching factor is better that way, most systems are set up to go backwards (cf MYCIN), but which is better really depends on the shape of the search space. Claim : It is very natural for people to reason forwards. Claim : We can view proofs by contradiction as a way of changing the goal so that one can use forward instead of backward reasoning. Claim : Proof by contradiction "increases goal space" because there are more paths to a contradiction than to a particular goal.

7. **Branch and bound**
The queue is sorted by costs and cutoffs used. A* is a heuristic version of branch and bound with admissibility condition on the heuristic function. Remark : The material in Chapter 4 of Winston is enough to know for the qual.

8. **Static evaluation**
Static evaluation is done on positions but people seem to evaluate operators instead and this seems better. That is, static evaluation function on states in a state space comes nowhere near the way people evaluate possible actions. In playing chess, we do not imagine a bunch of possible moves and ask which of states they lead to is

best, but we have goals we want to accomplish and we select moves we think likely to contribute to achieving those goals.

The above really seems to be a top-down/bottom-up distinction in that people use a ranked goal set to determine which moves to examine while most game programs do not use goal-directed move generation. Some attempts have been made to do this in PARADISE (Wilkins). PARADISE was able to look down 18 ply, but has not achieved a master status (like Berliner's program which uses heuristic evaluation functions). This should not be taken as a sign that the approach is intrinsically weak, only that it is harder to write a smart program than give a dumb one a CRAY to run on.

9. **What is AI?**

We decided that we would not try to define AI today because that would probably be pointless and would waste a lot of discussion time. This question was to be kept in mind, however, and we would attempt to answer it at the end of the quarter. Several useful books and articles address this issue, and reading them is recommended to get a feel for the cultural issues in AI.

- Nils Nilsson : *AI prepares for 2001 AD* : AI Magazine 1984
- Fischler and Pentland : *Up against the wall, Logic Imperialists !* : AI Magazine 1984
- Introduction to the COMTEX microfiche series (SAIL, MIT) : AI Magazine
- Schank : *One man's opinion of the state of AI* : AI Magazine
- McCarthy and Lighthill debate transcript
- Minsky : *Steps to AI* : in Computers and Thought
- Feigenbaum : *IJCAI77 address*
- Boden : Artificial Intelligence and Natural man
- McCorduck : *Machines who think*
- Dreyfus : *What computers can't do* : 2nd edition
- Anderson : *Minds and Machines*
- Weizenbaum : *Computer power and human reason*
- Winograd : *Computers and Cognition*

Chapter 3

Search and Heuristics II

3.1 Breadth-First Search vs. Depth-First Search

Space complexity(B = branching factor; D = depth):
BFS:B^D, DFS: B*D
Best First: intermediate between the requirements for DFS and BFS.
Hill climbing: 1
Both searches are of the form

1. make a queue consisting of just the root
2. remove the node at the front of the queue
3. expand this node into its children
4. insert the children into the queue
5. go to (b)

The difference between the searches is in step(d). In DFS, the children are inserted at the front of the queue. In BFS, the children are inserted at the back of the queue.

3.2 Uniform-Cost Search

In a uniform cost search, the path with the lowest cost is extended at each iteration. This search strategy thus expands the lowest cost frontier. This cost of the path is the sum of

the costs of the edges in the path. (Note: Breadth first search is just a special case of this which arises when the cost of all edges is equal)When the algorithm is about to extend a path that has a goal node at the end of it, this path is guaranteed to be the least cost solution. This corresponds to the standard shortest path algorithm.

3.3 Shortcomings of various search algorithms

Depth first search can be bad for infinite search spaces. Some such spaces arise in theorem proving and symbolic integration, and other spaces that have cycles/recursion. Breadth-first search can be bad in very branchy domains. It is also a losing strategy for finding the combination of a lock, unless partial success information can be obtained (some tumblers disengaging, for instance).

3.4 A^*

You should know the basic outline of the admissibility proof of $A*$. The monotonicity/consistency condition of a heuristic function h(X,Y) (the approximated cost of getting from X to Y), is:
$h(A,C) \leq h(A,B) + c(B,C)$
where C is a descendant of B and B is a descendant of A. c(X,Y) is the minimum cost of getting from B to C. See page 82 of Nilsson's text for the implications of the monotone restriction. If $h1(n) \geq h2(n)$ for all nodes, and both heuristic functions are admissible, the nodes expanded when using h1 will be a subset of those examined when using h2.
Note that the cost above has two components: there is the search effort, measured by the number of nodes expanded, and the solution cost, which is the length of the path found. In practical problems, we wish to optimize some combination of both of these cost components. BFS gives you the shortest solution path, but it is not optimal wrt search effort.
Defn: A search method has more heuristic power than another, if the averaged combination cost of the first is lower than that of the second.
Evaluation functions on nodes are based on various ideas.

1. probability that node is on the best path.

2. distance or difference metric between that node and the goal set.

3. In board games and puzzles, a configuration is often scored on the basis of features that it possesses that are thought to be related to its promise as a step toward the goal, e.g the number-of-tiles-out-of-place heuristic in 8 puzzle.

There are a number of ways to reduce the average computation time of a search using A^*.

1. use an h that is not admissible
2. multiply an admissible h by a c > 1
3. only calculate a quick approximation to an admissible h

Some measures of performance of a search algorithm are

1. penetrance (pg 91, Nilsson's text)
2. effective branching (pg 92, Nilsson's text)

We worked out the best first search (Winston's and Nilsson's) on this example.

3.5 A Taxonomy of Blind Search Methods in AI

This section is thanks to Andy Golding.

1. Depth-first search
2. Breadth-first search
3. Best-first search
   At each iteration, best-first search picks the lowest cost node n, where
   cost(n) = C(r,n) + E(n,g)

where C(r,n) is the exact cost of getting from the root node to node n. E(n,g) is the estimated cost of getting from node n to the nearest goal node.

This algorithm is admissible (i.e guaranteed to find the optimal solution) if E(n,g) never overestimates the true cost of getting from node n to the nearest goal node. If E(n,g) does overestimate this cost for a node on the optimal path, the algorithm may skip that path, thinking that it is worse than it really is. The following are special cases of best first search that are of interest:

i. Winston's misguided idea about best first search

Set C(r,n) = 0, i.e pick the node that you EXPECT will lead you along the cheapest path to a solution. The idea is to get to ANY solution as quickly as possible. It doesn't matter that the solution may be suboptimal.

ii. Shortest path algorithm

Set E(n,g) = 0, i.e pick the node that is closest to the root node. Because we are visiting the nodes of the search tree in order of increasing distance from the root, we will always find the shortest solution path first. Optimality also follows from the observation that E(n,g) = 0 could not possibly overestimate the true cost (costs are assumed to be non negative).

4. Branch and Bound

   This is just best first search, enhanced to make it more efficient. We keep track of the best solution path seen so far, and whenever a current path reaches that length, we can throw it away.

5. The A^* algorithm

   This is the best first search modified to work on directed graphs as opposed to search spaces that are trees. The new bookkeeping involves matching new nodes against old ones to make sure we do not have two copies of a node. Also, whenever there are multiple paths to a node, we discard all but the shortest one.

3.6 Miscellaneous

1. Integrating heuristics into the search framework.

   (a) Numeric evaluation functions like A^*.

(b) Strategies at nodes *a la* Georgeff.

(c) Constrained generation like in Dendral, by the use of domain knowledge.

(d) Advice taker formalism (declarative control information).

2. Criteria for comparing search algorithms,

(a) How, at each stage of the search process, a node is selected for expansion.

(b) How operators applied to that node are selected.

(c) Whether an optimal solution can be guaranteed.

(d) Space requirements.

(e) Will a given node be considered more than once.

(f) Under what circumstances will a particular search path be abandoned.

3. What factors determine if you use forward or backward search?
The shape of the search space determines this, (see figure on page 157 of Winston's text, 2nd ed) This should help one understand why forward search is used in most game playing situations.

4. Definition of a heuristic
See the proposed definitions in Section A of Chapter 2 of the AI handbook, pages 28-30. Some examples of heuristics -

(a) The greedy algorithm for solving the TSP. Note that the use of this heuristic reduces an exponential time problem to a polynomial one, but does not bound the resulting error.

(b) The heuristic evaluation function in A*. This is better than the previous one because we can get bounds on the error and the assurance of optimality.

(c) Samuel used a heuristic evaluation function for evaluating the nodes generated in the game tree. He used this to decide which node to expand next, which successors to generate, thus pruning large sections of the tree.

(d) In LT, heuristics were used to determine which operator to apply next in the generation of a proof.

(e) In AM, Lenat used heuristics of the sort: If there is an interesting function of 2 arguments, study its behavior when the two are identical: These were used to focus the efforts of AM in the discovery of interesting concepts in mathematics.

(f) Heuristics were used in Gelernter's geometry theorem prover, Guzman's segmentation program.

5. Approaches to reducing search.

(a) Reformulate the problem to reduce search space- e.g the mutilated chessboard problem. Also the missionaries and cannibals problem in the Amarel 1968 article.

(b) Use heuristic knowledge from problem domain to focus search.

(c) Abstraction of search spaces as in ABSTRIPS.

6. How search was reduced in Gelernter's geometry machine.

(a) Syntactic symmetry was exploited. Thus if parallel goals were symmetric, only one proof would be done, and the other would follow "similarly".

(b) Use of a diagram to prune search paths.

Chapter 4

Knowledge Representation I

4.1 Semantic Nets

Q: Nilsson's version of semantic nets seems to be no more than a graphical rendering of "units," and a rather cumbersome one at that. Why have semantic nets at all?

A: Semantic nets have proven useful, e.g. in Winston's analogizing program. But note that Winston just used a subset of the theory – it is not clear that semantic nets in their full glory are implementable.

In 'What's in a Link?' Woods gives perhaps a clearer exposition of semantic nets. They were originally proposed by Quillian (1968), although Frege's (1879) two-dimensional diagrams may have been a precursor. An early motivation for the nets was natural language research.

It is argued that semantic nets are relatively content-free. For instance, the well-known ISA link seems to be a conflation of two distinct relationships, member-of-set and subset-of-set. Thus semantic nets do not appear to have a well-defined, consistent semantics (although Hewitt suggests a way using articles to clarify ISA use). Another strike against them is that they are 'flat',i.e. they lack data types – all nodes are of the same 'class' in some sense.

In defense of nets, it should be noted that many of their flaws crop up with other representations as well.

Recommended reading: Brachman et al.'s Krypton paper, which is on reserve in the Math/CS library. It combines two formalisms, semantic nets and logic, in an attempt to get the advantages of both.

reality; in particular, they can be used to model the 'spreading activation effect' and the related 'priming effect', see e.g Anderson's work.

4.2 Knowledge representation vs. use

The way that knowledge will be used ("what to do with it") clearly interacts with the representation chosen ("how to say it"). This was elegantly demonstrated by Amarel (see "required papers"). This may be one reason why there are so many different formalisms for knowledge representation – each is suited to (and designed for) a particular application, but none is "the" definitive, general answer.

There have, however, been attempts to develop general languages for knowledge representation, e.g. KRL (Bobrow, Winograd, et al.). The KRL project collapsed under its own weight because it tried to be everything to everybody ("the PL/I syndrome"). It spawned two offspring: Winograd's Aleph specification language, and Bobrow et al.'s Loops package for Interlisp.

One other knowledge representation language of interest is RLL (Lenat and Greiner). RLL is a frame-oriented language with 'knowledge about itself', in that it is a language to talk about representation languages. Lenat's Eurisko program was written in RLL. Teresias(Davis 76) also shares some features of RLL.

4.3 Content vs. form

Is there a clean separation between content and form? Related to the distinction between representing and using knowledge. To illustrate the difference between content and form, consider the Roman and Arabic numeral systems. Both have the same content, in that they both behave according to Peano's axioms of arithmetic (excluding the lack of a zero in the Roman system, etc.). But it is far easier to describe algorithms for integer multiplication (e.g.) in the Arabic system. The only way to account for this difference is to attribute it to the different *forms* of the two representations.

A related distinction is that between implicit and explicit knowledge. Suppose, for instance, that we would like to represent the fact that California is in the USA, and that the USA is in North America, and we would like to be able to conclude that California is in North America. Using a logical representation (CA is-in USA, USA is-in NA), the desired

conclusion is only implicit (i.e. it can be derived, but is not immediately known); but with a graphical map-like representation, the transitive inclusion would be explicit, by virtue of the transitive property of inclusion in graphs. One caveat: it is not completely clear what it means for graphs to intrinsically incorporate this transitive property. A computer would have to "do processing" to figure out that CA is in NA; in particular, it would have to skip over the interposed USA boundary to determine that one region "is in" the other. This corresponds to the processing involved in applying the transitive law to logic sentences.

Implicit knowledge can sometimes be difficult to tease out of a program. Say we have a chess program that makes good moves because it happens to pick the first move on its list - then its knowledge of the goodness of moves is implicitly coded into its selection procedure. As another example, consider a program that keeps track of bank accounts. But rather than storing the correct balance for each account at all times (which would require a great deal of updating, especially if interest is compounded daily), the program uses a compound interest formula to compute balances on demand. Thus your balance is not stored explicitly - it is somehow embedded in the code of the program.

Related topics are discussed in Marshall McCluhan's *The Medium is the Message.*

The form vs. content issue also arises in epistemology. Take the sentence, *The Morning Star is the Evening Star.* This is in fact a tautology, because the Morning Star is Venus, and so is the Evening Star. Yet although people agree that X = X is true, many don't realize that the above sentence is true too. Frege tried to explain this paradox by distinguishing between two sorts of content, what he called *sense* and *reference.* The name 'the Morning Star' merely designates an object, namely Venus (its reference). The sense of 'the Morning Star', however, varies from person to person. For some people, the sense of 'the Morning Star' differs from the sense of 'the Evening Star'.

4.4 Logic

(This section and the next are answers to the questions on the reading list.)

- *program* : Strips uses a predicate logic representation and applies resolution theorem proving to tell whether it has reached its goal state. Between states, however, it performs non-logical operations such as deleting and adding logical formulas to its world description.

- *advantages* : Its semantics are clear and uniform.

- *disadvantages* : It is not always obvious how the behavior of the program will change if new facts are added to its initial state. Because first-order predicate logic is only semi-decidable, the program could go into an infinite loop trying to establish an untrue condition. Another disadvantage of logic representations in general is that they seem unwieldy for representing structures such as geographical maps, as an inordinate number of logic formulas would be required (consider what would be involved in writing Winograd's SHRDLU using a first-order logic representation).

- *example* : McCarthy claimed that the mutilated chessboard problem (remove two diagonally opposite corners of an 8-by-8 chessboard and then try to cover the resulting board with 31 2-by-1 dominoes) could not be solved by standard logic methods because of the ensuing combinatorial explosion. However, Lwas used a hairy resolution theorem prover to solve it.

- *issues* : Logic representations cannot deal adequately with uncertain knowledge (a Mycin-like framework would not work for rules containing quantifiers), beliefs, etc. One approach designed to combat the combinatorial explosion problem is to reason about your reasoning, i.e. to reason at the meta level (e.g. Bundy et al.'s Mecho project).

4.5 Procedural representations

- *program* : SHRDLU – for each sentence, it constructs a program out of templates, and then runs the program to recognize the sentence. All planning programs could also be said to use procedural representations, as they represent tasks as sequences of actions (e.g. Sacerdoti's procedural nets).

- *advantages* : To update a database, we can just change it directly. Compare this with the logic formalism, where we have to first recopy all data except the part we wish to change. The frame problem appears to be much less formidable when we have the full power of procedures at our command.

- *disadvantages* : Programs that use procedural representations tend to be ad hoc and difficult to prove correct and modify.

Chapter 5

Guest session for Knowledge Representation

Guests : Ron Brachman (FLAIR) and Michael Genesereth (Stanford)

Ron Brachman gave us a historical overview of semantic nets. Semantic nets first appeared in Quillian's work.He used them for representing words and the links between nodes stood for relationships between the words. Spreading activation was used to relate words. This was part of a psychological modelling experiment. It is hard to characterize semantically what this was all about. More and more work in semantic nets followed Quillian's. Gary Hendrix devised partitioned networks, for representing quantification. His system was able to represent only a subset of first order logic. Schubert et al. produced a notation with which almost all of first order logic (and modal operators as well as time) could be represented. It is interesting to note that he used first order logic for describing what the nets stood for. Semantic nets thus constitute a two dimensional representation for the linear propositional notation. KLONE and other related systems are derivatives of semantic nets. They take a part of first order logic that is useful for representing certain kinds of knowledge (e.g taxonomic knowledge) and provide a mechanism for reasoning with them efficiently. They provide computational efficiency at the cost of expressiveness. The full first order logic is semi-decidable. So expressive power and computational efficiency seem to be at odds. Brachman, the designer of KLONE feels that it did not represent a 'natural' subset of first order logic. It is hard to determine what subsets of first order logic that the

that all semantic nets and frame based systems had over first order logic, was the ability to handle exceptions and default in a computationally tractable manner. See Genesereth's memo : Fast inference methods in semantic nets. See Etherington's paper in AAAI-83.

Genesereth then gave the following position statement. Classical knowledge representation research has confused too many issues. One formalism cannot deliver representational adequacy, inferential adequacy and perspicuity, because each of these make conflicting demands on the characteristics of the representational formalism. So concentrating on any one language and tuning it to achieve all the above is both impossible (?) and (thus) unwise. Instead have a grab bag of special purpose languages [English (!), circuit diagrams, maps, musical scores] and choose the most perspicuous (from the user's point of view) language for the current task. But what of inferential adequacy? Have a grab bag of inference mechanisms, apply the inference method that is best suited for the computation.

Brachman's rejoinder : He agreed in principle, to what Genesereth said, except that he had a cautionary statement to make about the grab bag approach. One has to be careful about the interactions between these formalisms and also check that they are consistent. He proposed the following : Have an internal lingua franca (logic) to mediate between these special purpose representations. There is no knowledge representation scheme and inference mechanism that is good for all tasks.

The discussion then took a question answer format.

- What formalisms have been used to represent fuzziness?
  Answer : Disjunctions and negations incorporate a kind of fuzziness. Another formalism is Zadeh's fuzzy logic. Though he makes a good intuitive case for it, the technical results are not too impressive from an AI standpoint. Attaching probabilities to statements in first order logic was proposed by Elaine Rich (AAAI-83). For example : Elephant is a mammal with a probability of 0.9. Ginsberg (AAAI-84) showed how you can use this to do default reasoning. Nils Nilsson works out the semantics of fuzziness in his "Probabilistic Logic" (SRI Tech note).

- What are the current issues in KR research?
  Answer : 5 yeas ago the main issues being investigated were inheritance with exceptions, how to extend other notations to increase their expressivity. But issues of interest now are

  - Knowledge level analysis

- Non monotonic logics
- Compilation
- Rational reconstruction of KR research
- Control knowledge
- Reformulation and the vocabulary choice problem

- What formalisms have been proposed to deal with time, space and causality?
  Answer : For time we have the situation calculus approach, look at James Allen's work.

These are further questions prepared by the TA which could not be asked during class.

- What is a representation?
- Why is KR an important area in AI?
- What formalisms have been proposed over the years, and in what contexts?
- Why is there such a diversity in KR formalisms?
- Lessons learned till now in the long history of KR. Major landmarks in this history.
- Current thrust of KR research. What the important issues?
- In the context of
  - Building more complex expert systems
  - Building systems that can reconfigure themselves to adapt to a fluid environment.
  - Representation reformulations a la Amarel.

  what are the specific KR issues?
- Compilation as a solution to inefficiencies of logic based KR systems. Comment on this. Note Minsky's complaint : A heuristic compiler will eventually need more general knowledge and common sense than the system it is trying to compile.

  What is the viability of a bootstrapping solution?
- What is the utility of taking Marr's view of KR?

- Winograd levels the following complaint on traditional KR – structures in the nervous system do not represent the world in which the organism lives but the structural coupling and interaction lead to behavior consistent with the possession of an explicit representation. Cf. the display hack program at MIT that draws circles but has no representation of circles, radii and centers. What is the response of the declarativists to this position?
- Hayes remarks that the frame problem is a representational artifact – what does he mean?
- To make up for poor efficiency of problem solvers using logic based KR's and resolution theorem proving, it has been suggested that adding control info will alleviate the problem. What are the issues here? Why has it not yet been done?
- The real question most often is not how to represent something but what the knowledge is that we need to represent. Comment.
- Understanding new terms generation in learning systems calls for a close linking between learning and KR research. How is this best achieved?
- What could be good test bed for investigating scale effects in representation?
- Minsky has the following problems with the logistic approach.
  - How do you get at the knowledge?
  - Answering the relevance question : adding more knowledge always slows down a theorem prover.
  - Monotonicity of classical first order logic.
  - Control knowledge – not adequately addressed by the logic folks.
  - Scaling effects

  What are the methods proposed to deal with these questions?

Chapter 6

Knowledge Representation II

Continuation of discussion about various knowledge representation schemes with respect to the issues outlined in TA's handout:

1. **Example of a use of the KR formalism**
2. **The operations that can be performed on it**
3. **An AI program that uses it and why**
4. **Advantages and disadvantages of the formalism with an example of a case where it is completely hopeless and another where it is extremely useful.**
5. **Current research issues**

6.1 Conclusion of discussion of procedural representation

Problem/issue: "can anyone ever understand it?" i.e. can you formalize this sort of work, so that we can all learn from it and it advances the field? The assumption here seems to be that procedural representations do not facilitate maximal explicitness or understandability in the representation of the world. Their behavior is hard to characterize because the technology of program verification has not been advanced to the point where we can prove any but the simplest procedures correct. Also the technology of specification is not very advanced either. If we have a behavioral specification of a program in a logical language

we would make no distinction between a procedural and a declarative representation. This was said in response to the question *Are procedural representations a hack?*.

- More on the *procedural representations a hack?* question :
  Claim: So what if you have a proof that a program satisfies its behavioral specs, if no one can understand it? You get nothing more out of the proof that you had before. Understandability of either representation is what is at issue.

  Counterclaim: The good thing about a proof is that you may get a good notion of the domain of validity of the program than you had before; i.e. you prove that it works over such-and-such a domain rather than just over the test cases.

- Question: What is the content of the *just a hack* remark anyway? Is it a derogatory equivalent to *incomprehensible* or a derogatory equivalent to *currently lacking formal underpinnings*? If the former, then procedural representation is neither more nor less a hack intrinsically than production rules (for example) although it may be easier (debatable) to abuse the representation. However, it is arguable whether the representation of activities as isolated facts with sequencing buried in the workings of some 'inference engine' is more comprehensible than a procedural representation of that activity. For example, the production rule system that does addition (see example in the Davis and King article) is extremely non-transparent.

- Question: Is there any KR scheme that people would be willing to defend as "not a hack"?
  Reply1: No.
  Reply2: As long as it has a model theory.
  Reply3: As long as it is *clear* enough.

- Question: Why do we believe logic proofs more than other kinds of explanations? We have strengthened our confidence in the soundness of our formal deductive machinery by the development of "model theory" which gives a rigorous semantic interpretation to the symbols of our logical system.

- Question: Should we require a model theory to back up any formal syntax? Reply: This is far to strong a requirement. Modal logics were useful for quite some time before a formal semantics was developed for them(by Kripke).
  Remark: A big problem in work on learning is that there is no 'semantics', that there

is no model theory for the kinds of syntactic manipulations done.

Question: Isn't this something of a conflation of formal semantics and meaning-in-the-world. There is formal semantics to explain the kind of junk done in the logic-based formalisms anyway; the problem is that it makes no sense in the world.

Hayes in *In Defense of Logic* - The procedural/declarative debate is foolish since it focusses on the wrong issues. There are two kinds of subject matter - how and what, and procedural and declarative representations are suited for representing one and not the other. We thus need both forms of representation, and the real issue is to determine when to use which.

It is also possible to obtain many of the advantages of declarative system without paying the performance penalty, by compilation and the use of semantic attachments. Explanations and debugging and modifiability suffer, however. Also declarative systems allow explication of control. Compilation will allow reformulation of a declarative spec of a computation along with control hints (that may be declarative or procedural) into a very efficient object code.

- See Elaine Rich's characterization of the plusses and minuses of declarative and procedural reps. This should help you understand when which is appropriate.

6.2 Frames

Question: Aren't frames a generalization of semantic nets? Reply: Aren't they a specialization of them? Reply: Aren't they actually equivalent?

Problem: Represent the transitivity axiom in frames and semantic nets. i.e. Represent : Vx Vj/ *Vz* R(x,y) and R(y,z) =* R(x,z)

Frame derivatives :

Scripts : for describing a sequence of events (Schank)

Stereotypes : for user modelling (Rich in the Programmer's Apprentice project)

Rule models : Describe a common set of features shared among a set of rules in a production system. (Davis 82).

Question: what are rules of inference in frames?

Operations on frames: (See Hayes, "Logic of Frames" in W and N) inheritance, default

values, instantiation, unification, "criteriality assumption" (the assumption that slots are necessary and sufficient criteria for membership), matching

What is 'matching'?: - "not just simple syntactic unification but depends on assumptions of domain" -finding instances of one frame type which can be viewed (because of some of their contents) as instances of another type - there are N zillion varieties of matching.

Programs using frames: AM, EURISKO, GUS

Advantages/disadvantages: Very useful because certain sorts of inference (property inheritance) can be done extremely efficiently as compared to an equivalent logic based system. Indexing is brought into the syntax of the language, which was the original motivation for the development of frames (grouping together related items).

6.3 Conceptual dependencies and other forms of semantic primitives

Advantages :

- All facts will be represented in canonical form.
- The rules that are used to derive conclusions from that knowledge need only be written in terms of primitives rather in terms of the many ways in which the knowledge would have appeared.

Disadvantages :

- A lot of work to convert to primitive form.
- A lot of space needed.
- hard to determine a set of primitives which can cover all that you wish to capture. Tradeoffs in the level of representation. The finer the primitives, the more detailed the inference becomes.

6.4 Analogical (direct) representations

For example: maps, circuit diagrams, musical scores, building layouts etc. Operations: depends on actual representation. Programs: Gelernter's geometry program

Question: Does SYNCHEM use analogical rep? What exactly is this distinction anyway? Is the X-C=O-Y type of representation used in bond-breaking rules analogical? It would seem so.

Fregean vs Analogical representations – see A. Sloman in AI Journal (early 70's). Analogical representations supposedly 'homomorphically' mirror certain relations in the domain being represented. Claim: The distinction does not look very solid, this is probably a continuum rather than a categorization.
The following is compiled by the author.

6.5 Pot-pourri of KR facts

1. A short summary of Hayes' *In Defense of Logic paper* :
   Hayes argues that modern formal logic is the most successful precise language ever developed to express human thought and inference. The real contribution of logic is not its sparse syntax but its semantic theory. He states that a formalism without a model theory does not constitute a representation language. He also re-examines the old procedural/declarative controversy and says that the distinction between procedural and declarative languages is false – there are two kinds of subject matter rather than two kinds of languages.

2. The problems with analyzing meanings using the CD structures of Schank is that although simple to use, the use of CD structures uses the assumption that there is a finite collection of basic words in terms of which the meanings of a sentences can be explained. (Too reductionist !). Other problems are :

   (a) This is perilously vague. It is hard to judge when two English sentences have the same meaning.

   (b) This is essentially a linguistic view of meaning.

   (c) Provides no useful guidance for how a system might use the representation [i.e what sort of inferences can it make?]

3. Contrast the above view with that of KRL – where the underlying philosophy is that a description cannot be broken up into a single set of primitives, but must be expressed through multiple views. Other features of KRL are

(a) Built on top of INTERLISP. It facilitates the representation of knowledge in frame structures.

(b) Knowledge is organized around conceptual entities with associated descriptions and procedures.

(c) It is possible to represent partial information about an entity and also provide multiple descriptions which describe the entity from different viewpoints.

(d) An important method of description is comparison with a known entity.

(e) Reasoning is dominated by a process of recognition in which new objects and events are compared to stored sets of expected prototypes, and in which specialized reasoning strategies are keyed in to these prototypes.

(f) Information is clustered to reflect use in processes whose results are affected by resource limitations and differences in information accessibility.

(g) Default values for slots.

4. Why proceduralists attacked the theorem proving paradigm for problem solving.

   (a) There were no theorem provers that could solve really hard problems that were also general.

   (b) The theorem provers being general purpose had no bias toward any particular domain and the result was that classical theorem provers knew little about what to do and were incapable of being told it.

   Jack Mostow's work tries to attack the advising part above and Genesereth and Smith's work tries to provide a framework for hanging control advice/ heuristics in a general theorem proving system.

5. Hayes' definition of knowledge representation is : A way of systematically representing knowledge in a sufficiently precise notation that can be used in a computer program. This scheme should be formal – given a particular collection of symbols we should be to able to say whether or not that is a legal sentence. Also the scheme must have an associated semantic theory. A semantic theory is an account of the way or ways in which particular configurations of the scheme have as their meanings particular arrangements in the real world. A semantic theory is necessary to answer questions relating to the equivalence of formalisms. Also the role of deductive,inductive and

analogical reasoning cannot be tackled without a clear model theory of the systems under discussion.

Brian Smith has advanced the following thesis which is called *The knowledge representation hypothesis.*
Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers take to represent a propositional account of the knowledge that the overall process exhibits, and (b) independent of external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge.

See Stan Rosenschein's view of KR (to be presented in the notes for Jan 25) to get an alternative view.

6. The advantages of a rule based representation are :

   (a) They are modular, so that rules can be added,deleted or modified without directly affecting other rules.

   (b) Uniform structure,syntax aids in knowledge acquisition, explanation and also if rules are being examined by another program.

   However, this modularity is forced, especially when there are implicit groupings of rules that apply in specific situations and at certain stages in the problem solving process. These can be represented and handled well in a frame based system (cf CENTAUR).

7. The following is thanks to Russ Greiner who was kind enough to answer some of the questions on RLL and KR in general that the author asked him. RLL arose out of the motivation to design expert systems for designing knowledge representation languages. Those were the days when there were a plethora of knowledge representation languages and it was thought that KR language design could be (semi)-automated. In this sense RLL and AGE had similar goals (AGE was to be an expert system building consultant). Also there was emphasis on an adaptable representation (cf Cognitive Economy). And to do that RLL needed to represent itself to itself *a la FOL's META.* Unlike FOL, RLL was ad-hoc, a collection of hacks. MRS incorporates the features of RLL in a cleaner and more principled fashion. RLL was a frame based system. It was better than Stefik's system UNITS in that (among other features) it allowed multiple

ISA links. Here follows a very simple example that indicates the power of the RLL machinery. Suppose RLL had the following information : An aunt is the wife of an uncle. Now suppose it were queried : Who is Fred's aunt? It would look up the uncle slot for Fred and then look up the wife slot of Fred's uncle. RLL had information on the domain and range of functions. For instance, it knew that aunts were always female. High level specifications were compiled into LISP code (semi-automatically), i.e the high level specification of aunt would get compiled into a procedure which executed the slot lookups in the order indicated above. RLL had rudimentary reason maintenance capabilities. For instance, if a new slot was added and the system had to be reconfigured, RLL would make changes without having to recalculate everything, because it had a representation of itself. In case this is fuzzy (as I am pretty sure it is, thanks to my presentation not Russ's exposition!) please look at the RLL paper for an example. This paper is in AAAI-80. There is a demo of RLL in the Building Expert Systems book, but it is not recommended by Russ for the purpose of learning about RLL.

8. Advantages of semantic nets over predicate calculus (from Schubert)

(a) They are more natural and understandable because of the one to one correspondence between nodes and the concepts they denote, the clustering about a particular node of propositions about a particular thing and the visual immediacy of interrelationships between concepts.

(b) They lend more readily to associative and comparison algorithms of the kind described by Quillian and Winston.

(c) Property inheritance and transitivity deductions extremely easy and efficient to do in this framework.

(d) Schubert has extended semantic net formalism to handle logical connectives, quantifiers, time, n-ary predicates, lambda abstraction and modal operators.

9. Desiderata for a good representation.

(a) Representational adequacy (i.e epistemological adequacy)

(b) Inferential adequacy

(c) Inferential efficiency

(d) Acquisitional efficiency

10. Limitations of pure first order logic representations. (from E. Rich)

(a) It is very hot today. *How do you represent relative degrees?*

(b) Blond haired people often have blue eyes. *How can amount of certainty be represented?*

(c) If there is no evidence to the contrary assume that every adult you meet knows how to read. *Inferring facts from the absence of other facts*

(d) It is better to have more pieces on the board than your opponent has. *Heuristic information*

(e) I know Bill thinks that the Niners will lose but I think they will win. *How can different belief systems be represented?*

apter 7

inning, Problem Solving and Ltomatic Programming I

Knowledge Representation trivia

fE was cited as being a knowledge representation that was actually being used (by)NSUL project at ISI - referenced in IJCAI '83 in an article by Lutakis et. aL). used to represent knowledge used for parsing. More recent knowledge representa- nguages are MRS (developed at Stanford by Genesereth, Greiner and Smith) and TON (developed by Brachman et. aL, see IJCAI 85 for references)

Constraint propagation *a la* Winston

i discussion of what the point was of Winston's example of logic proofs (in Chapter s book, 2nd edition) being done with constraint propagation, the consensus was that mainly done for expositional value. It was suggested that there where many other as that could easily maintain all of the pointers to relevant things that the constraint ration maintained, and that these schemes could handle truth maintenance just as

7.3 Winston's credo

Winston is of the opinion that not much research effort should be spent on search control. He seemed to think that better advances to solving a problem could be had by concentrating on finding the right representation.

7A Issues in Planning

A number of active planning issues were mentioned. The first of these was *representation of actions.* This has two sub-problems. The first is that every action should have a list of prerequisites that describe all the possible conditions when the action can take place. Unfortunately, in the real world, this list can never be complete. This is the qualification problem. A solution to this is circumscription proposed by McCarthy (see article on Circumscription in the Webber and Nilsson collection). Also, there is the famed "frame problem", which arises from the need to represent all the invariances (non-effects) of an action. Solutions to the frame problem are explained in Hayes* article on this problem in the Webber and Nilsson collection.

The second problem is how can conditional plans be made. It was pointed out that all possible conditions can't be planned for. Hence there will probably have to be many iterations of some interleavings of plan, execute, and verify steps.

The third problem was how to do planning of parallel actions in the context of multiple agents. The role of communication in the synchronization of such plans as well as issues of cooperating agents arise in this context. Very little work has been done in this area and McCarthy cites this as one of the most important areas of research in AI in his list of problems in his 1980 paper "Epistemologicai problems in AI".

7.5 Planning methods

Currently, there is a wide variety of different planning methodologies. Some of these are: hierarchical planning, non-hierarchical planning, opportunistic planning, and script based planning. It was strongly recommended that people know examples for each method, problems that it can't handle and why it can't handle the problem. A nice summary article on various methods of planning is Sacerdoti's "Problem Solving Tactics".

Most of these methods were developed in the context of single agent planning. Very few address the issues in monitoring plans (STRIPS had an elementary facility to monitor plans and replan in the face of unexpected events) and this is still an important practical problem in real world expert systems that plan.

A problem related to plan generation is plan-recognition. This involves critiquing an input plan and possibly filling in details. The main example that was mentioned was the Programmer's Apprentice. The MACSYMA advisor is another. All ICAI programs need to do plan recognition.

7.6 Questions on planning

Relationship between planning and search. Once the actions are formalized, finding a plan amounts to searching in the space of all possible sequences of actions for one which achieves the goal.

SIPE was mentioned as a system that takes resource constraints into account. This is something that STRIPS can't handle. SIPE can also salvage failed plans using a teleological commentary on its plans. Libra is an AP system that reasons about its resource constraints. Common sense planners : Advice Taker, some from the domain of circuit design (recent work at MIT and some at Stanford).

Green uses frame axioms to deal with the frame problem. Can STRIPS be encoded into the situational calculus formalism? Look at Hayes' paper to get ideas on how to do this. Another important contribution of Green's thesis was the answer extraction method (Winston calls it Green's trick). This is covered in Nilsson's text and also in the Green paper in the readings.

Goal Regression as is used in Waldinger is an important technique for synthesizing plans. (STRIPS also uses it in learning MACROPS – see Fikes, Hart, and Nilsson.)

What are the main ideas in MOLGEN *a la* Stefik? Read the two Stefik articles in the readings. You should be able to illustrate constraint posting and partitioning the control problem into levels with examples.

Thought question : What if some of the expert systems with very simple control schemes (MYCIN, AM) had a control component as complex as that of MOLGEN. What improvements (if any) can we expect in them? Will they be able to handle a wider range of tasks?

Chapter 8

Planning, Problem Solving and Automatic Programming II

8.1 Planning systems

Chapter six of Winston was cited as a source of examples of the generate and test paradigm. Specific systems using it are Dendral and Acronym.

Question: Do all planning systems use backward chaining? Not necessarily - R1 does forward chaining.

Next came a rather religious discussion of when a system is a planner. One group seemed to be saying that when looked at in the proper way almost any program becomes a planner. One "proper way" involves relaxing the object/action distinction so that the configuration of the solution is a plan. Another is to note that the steps taken by a program in finding a solution constitute a plan of sorts. The traditionalists objected that the steps used in getting an answer are different than a sequence of actions that constitute an answer.

GPS/Means-ends analysis: The class's opinion seemed to be that GPS was general purpose only when compared with the other problem solvers from GPS's time period. Perhaps the most significant contribution of GPS was the separation between knowledge and inference. GPS would be awkward for use in a MYCIN like system, largely because it doesn't handle uncertainty well and also because GPS basically forward chains. It would work much

subgoals could cause excessive backtracking.

On page 163, Winston gives his six problem solving paradigms:

1. describe and match
2. goal reduction
3. constraint propagation
4. search
5. means end analysis
6. logic (Theorem proving)

Abstraction defers details and increases efficiency in many cases. There's a good example of this in the AI Handbook, Vol. 3, page 529. Two good example systems are ABSTRIPS and Noah.

Many planners use the linearity assumption: that subgoals are independent and can be achieved in any order [Sussman's def.] The handbook has an example of a case where this is violated - stacking three blocks in a specified order. Example systems are Interplan and its successor, Nonlin.

The radicals and the traditionalists of the class agreed that automatic programming is planning, because a program is clearly a sequence of actions acting on data structures to achieve a goal.

8.2 Pot-pourri

1. Is PARADISE (Wilkins' *Using Patterns and Plans in Chess* an expert system for playing chess? How does it differ from a conventional robot planner? How was the knowledge engineering done? How does this compare with Berliner's system that uses evaluation functions on board positions?
2. What is Waldinger's contribution to AI? How does his goal regression approach differ from STRIPS and HACKER? How is the frame problem handled in his system?
3. The most readable presentation of GPS is in Chapter 5 of Winston's book (2nd edition). Operating on differences is the key idea in GPS. GPS is a metaphor for problem solving that stresses states, differences between states, and operators for reducing differences. The

observed difference determines what operator to try next, but forward progress is not guaranteed, because the difference measures may be crude. The GPS control structure commits itself to a depth first search of the state space.

Thought question: Think of a common activity like cooking which requires several activities going on simultaneously and synchronously. Show how you would cast it in the GPS framework (repeat with the situation calculus framework).

4. Give examples of some constraint satisfaction problems.

- Cryptarithmetic
- Any design task (circuit synthesis, program synthesis, MOLGEN's task)
- labelling of line drawings(Waltz labelling algorithm)
- The map coloring problem
- Sussman and Steele developed a language for stating and implementing constraints in the digital ckt world.
- SAFE (Balzer) used constraints to verify specifications (in AP).
- Programming languages like Planner and Conniver were developed to make specification and use of constraints easy.

5. Notes on the Advice Taker

This is a summary of the paper *Programs with Common sense* by McCarthy.

The Advice Taker is a program for solving problems by manipulating sentences in a formal language. The difference between this and LT and the Geometry Machine of Gelernter is that in the latter cases, heuristics for problem solving are embodied in the program control structure as opposed to being stated explicitly in the same language that the program uses for representing problems. The Advice Taker is intended as a basis for an introspective system. The advantages of making control (problem solving) knowledge explicit is that behavior of the program will now be improvable merely by making statements to it, without knowledge of the program itself. The program will deduce for itself the immediate logical consequences of anything it is told and its previous knowledge. In fact, a program is said to have common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it previously knows.

The stated objective of the Advice Taker experiment is to use this framework to make machines learn from experience as effectively as we do. If we want a machine to discover an abstraction, the machine the machine will have to be able to represent it in a relatively simple way. One known way of making a machine capable of learning arbitrary behavior is to make it possible for it to simulate those behaviors. The problem with this approach is that the density of interesting behavior is low. Also small important changes in behavior at a high level of abstraction do not have a simple representation, (i.e small changes in representation do not correspond to small changes in behavior).

McCarthy then lists a set of requirements of a system which is to evolve intelligence of a human order. These are

- **All behavior must be representable in the system. (You cannot learn what you cannot represent)**
- **Interesting changes in behavior must be expressible in a simple way. (This axiom was rediscovered in the context of the AM and EURISKO experiments)**
- **All aspects of behavior except the most routine must be improvable. (Including the learning ability)**
- **The machine must have or evolve concepts of partial success.**
- **Should evolve notions of interestingness, utility of subroutines that it has.**

The first step in this process is to make a machine that can learn by being told (After all, if it cannot learn from being told, how can it learn by discovery or other more independent means of learning?). McCarthy advocates the use of logic as a representational medium and sketches out the use of situation calculus in the remainder of the paper.

The version of this paper that was presented in London (in 1959) has the discussion that this paper generated in that conference, it is between McCarthy and Bar-Hillel. (For copies see Prof. McCarthy's secretary).

Chapter 9

Guest Session on Planning, Problem Solving and Automatic Programming

Keith UUt

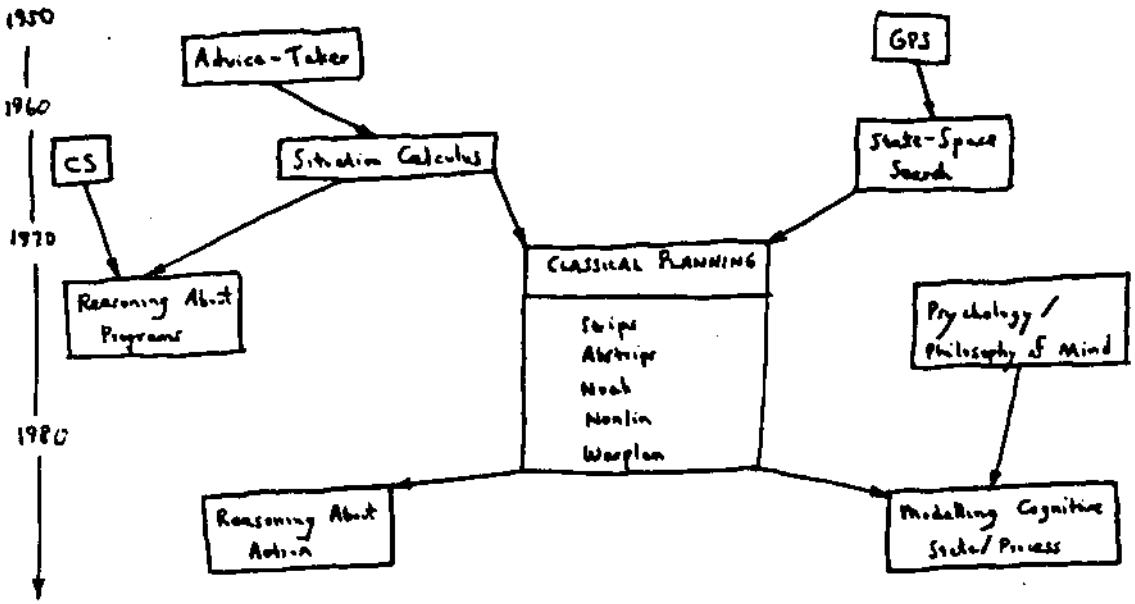
PLANNING

by Stan Rosenschein, SRI-AI

Main Points

1. Planning is best seen as part of a "Grand Strategy" for implementing [illegible] & mind/rational action.
2. Logic has two distinct roles to play.
3. The G^J 3...ategy has been useful, but there are problems and alternate strategies.

Historical Perspective

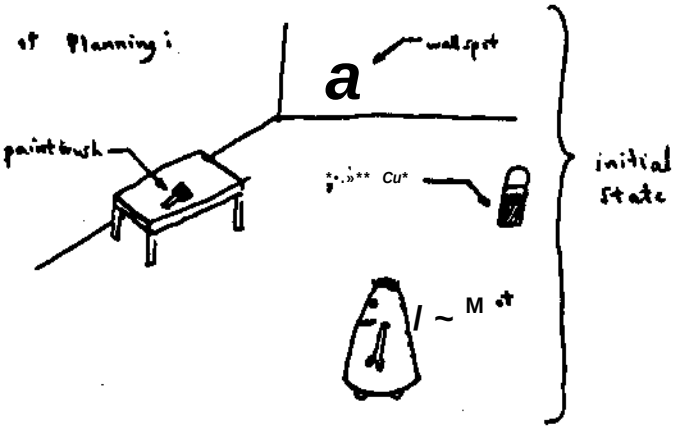


Classical Planning Problem

Given
1. an initial state S_0
2. a goal G predicate

Find a sequence of actions $a_1, a_2, \ldots, a_n$ which will transform the initial state to a state satisfying the goal condition G.

...le of Planning:



goal predicate { PAINTED (WALLSPOT)

(I'll use caps because t
this way LISP work

... Descriptions

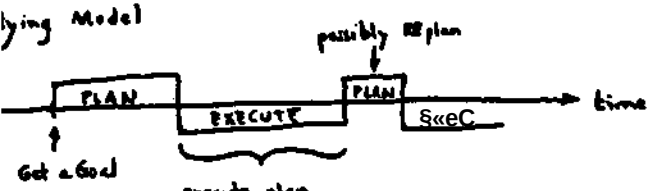
f*|o»t»om*uf	Operator	Preconditions
	GoTo (L)	At < ^ « . ;
	Grasp (X)	Holding (X)
Painty (X) Paintbrush (X) Holding (X) At (robot, Y)	Draw (X, Y)	Painted (Y)
Paintbrush (X) Paintcan (Y) ¬ Empty (X) Holding (X) At (Robit, Y)	Dip (X, Y)	Painty (X)

The Plan

< GoTo (table),
Grasp (paintbrush),
6§i© f I^A^CAM 'o
Dip (paintbrush, paintcan)
GoTo (wallspot),
Draw (paintbrush, wallspot

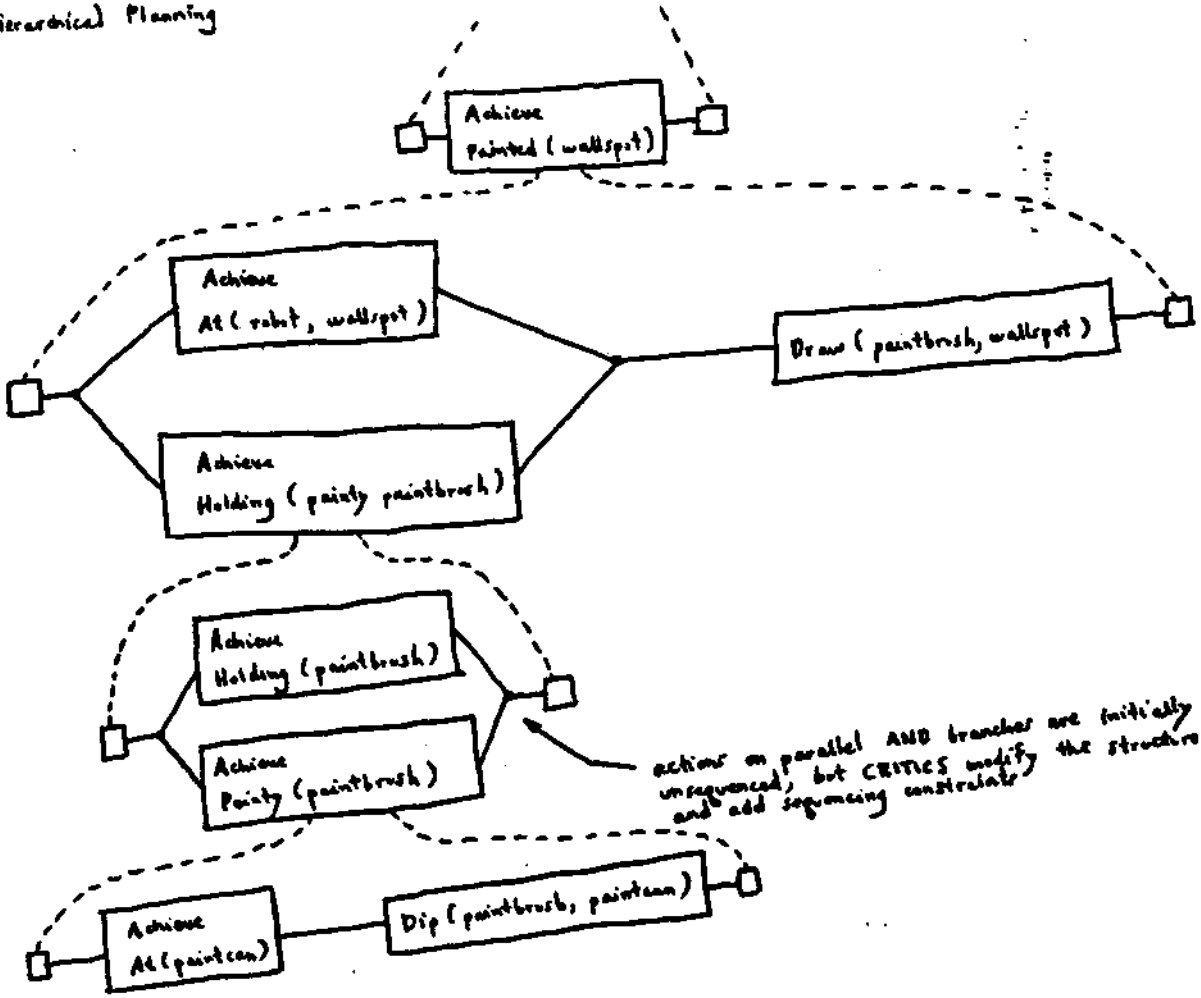
The plan achieves Painted (wal

...lying Model

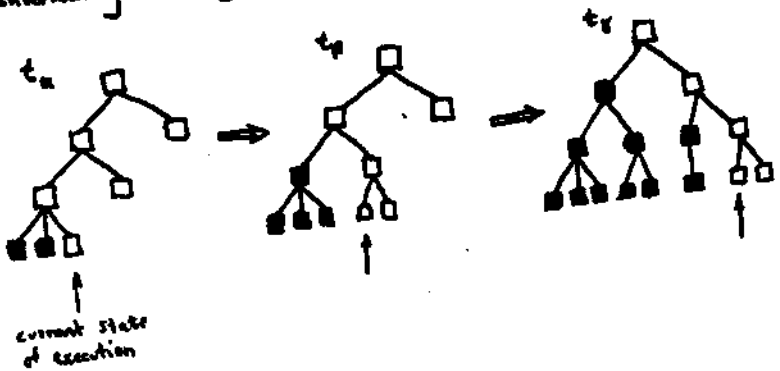


execute plan step-by-step, testing preconditions on actions } for —r'l'', before executing Dip (paintbrush, paintcan), test whether ¬ Empty (paintcan)

Hierarchical Planning



Interleaving Planning and Execution in Hierarchical Planning:



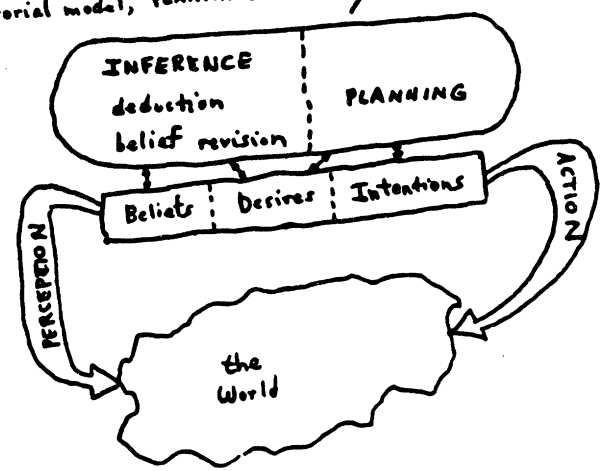
with "semantic nets" of the same era, there was confusion about the precise meaning of
- based planning structures.

n network seems to be an encoding of at least three types of information:
he syntax of complex action descriptions, e.g., AND-split, OR-split, sequencing.
he internal logic of a plan — roughly, a hierarchical proof that the plan achieves a certain conditi
he intentional state of the agent (which things he has decided to do).

n reasoning processes:
ying to do too much with a single structure, classical planning ends up providing neither an adequate
of action nor an adequate model of intentional state. Still, it represents a first step toward
fied computational theory of mind and action.

o STRATEGY — formalize and implement "folk psychology":
psychology views agents as having beliefs, desires, and intentions, which vary over time.
mponents of mental state are described in ordinary language as relations between an agent
a proposition:
X believes P
X desires P
X intends to α

ollowing pictorial model, PLANNING is really INTENTION MANAGEMENT:



The management of intentions ove
is analogous to the management of

The purpose of belief revision is to
in tune with reality as revealed by

The purpose of intention formation
is to keep intentions in tune with
and desires according to principle
rationality.

of Donkey:
key is midway between water and hay and is equally thirsty and hungry. Neither choice is
ferable, so decision is deadlocked.

Formalizing "form" psychology:

$B^{*,*}_{x,t}(t)$ — X believes P at time t

$P \rhd Q$ — According to what X desires at time t, P is preferable to Q.

$\mathbf{V}$

X intends at time t that P be made true.

$\sigma_{x,t}$ — X observes stimulus σ at time t.

$\alpha_{x,t}$ — X performs action α at time t.

Ji, - ...s which may — ; dart, — ...l structure desired.

[knowledge] $B_x P \supset P$

[introspection] $X_x P \supset B X_x P$ for $X \in \{B, D, I\}$

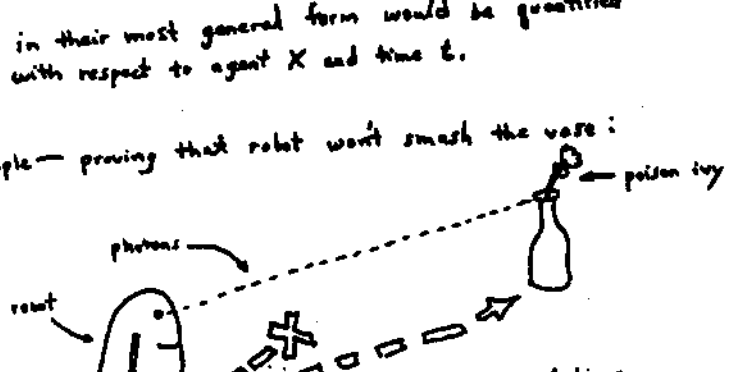
[consistency] $B_x P \supset \neg B_x \neg P$

["seeing is believing"] $\sigma \supset B_x \sigma$

[rationality] $\alpha \supset \neg \exists \beta \, B_x (\beta \rhd_D \alpha)$ — Note: this notion tends to exclude actions as irrational rather than predict specific actions

→ in their most general form would be quantified with respect to agent X and time t.

Example — proving that robot won't smash the vase:



Actions

Stop (robot, t_i)

Turn (robot, t_i)

Move (robot, t_i)

Stimulus Predicate

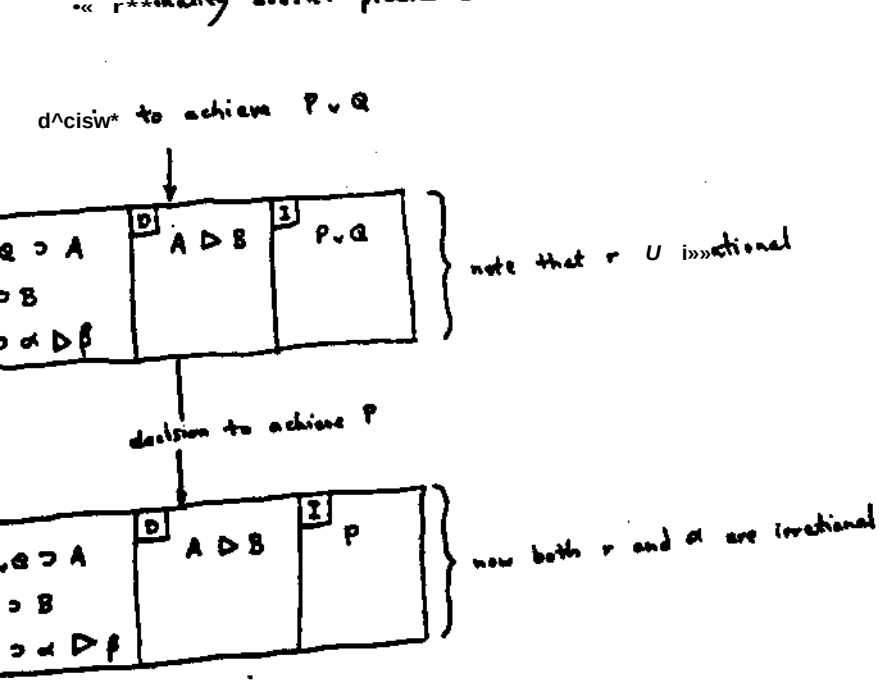
See Ahead (robot, va

Since alternate path toward right does not smash vase, and this is preferable to smas

tion: Why have intentions?

er: Robot must decide whether to stop and turn or to continue moving, even though a ration

«« rationality doesn't predict which he -o«« *».



this view of intention, what is a plan?

A plan is a collection of intentions, possibly involving complex action expressions, which, taken

can be viewed to "force" some preferred state of affairs.

dvantages of this view:

1. the full power of logical language can be used to express the content of individual inten
   Example: Water all the plants without getting the shelf wet.
2. Plan elements do not have to be temporally contiguous.
   Example plan for seeing a r^u/: buy tickets Oct. 17;
   drive to theater on Nov. 3 at 7pm; ...
3. tv* logic of action and the representation of decisions made have U- separated.

ALIRING BDI-psychology and rationality is only half of the GRAND STRATEGY.

t must be implemented; in principle, there are many ways of defining a computational

tate that can be interpreted as a [B, D, I] triple, faithful 4. the required constraints.

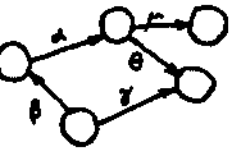
ttention has focused on one:

The GRAND TACTIC — the symbolic representation approach:

Operationalize < B, D, I > as collections of symbolic data structures (e.g., encoding
expressions of a knowledge-representation language) #PLUS* some deductive closure o
(possibly non-monotonic). This tactic is so pervasive that some consider J+ a necessary
condition of machine intelli^c e / C^schein disagrees.]

...ting the underlying model right.
...aking the formalism convenient for the designer.
...aking reasoning with the formalism efficient for the machine.

...s an underlying model the notion of actions being state transitions:



...ous formalisms can be used to describe the properties of actions:

1. Situational Calculus (McCarthy) $P(s) \supset Q(\alpha(s))$
2. STRIPS (Nilsson et. al.) Op. α — Pre: P; Add: Q, R; Del: —
3. Dynamic Logic (Pratt, Harel) $P \supset [\alpha]Q$
4. Temporal Logic (Pnueli) $P \wedge \alpha \supset \circ Q$

...nal Calculus and Circumspection can eliminate the ...ed for frame axioms.

some underlying model, but different properties:

	Suppression of State Variables	A...
Situational Calculus	No	
STRIPS	Yes	
Dynamic Logic	Yes	
Temporal Logic	Yes	

...underlying model: actions are considered as predicates on time intervals:

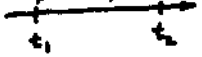
analogue to situation calculus would use full logic with explicit time variables

...xample 

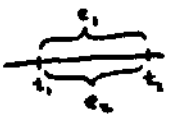
$P(t_1) \wedge e(t_1, t_2) \wedge \exists t\,(t_1 < t \wedge t < t_2 \wedge \alpha(t_1, t))$

This is expressive but cumbersome.

...iables can be easily suppressed by introducing operators on event predicates:

[concatenation] 

$(e_1 ; e_2)(t_1, t_2) \equiv \exists t\,(e_1(t_1, t) \wedge e_2(t, t_2))$

[conjunction, disjunction] 

$(e_1 \wedge e_2)(t_1, t_2) \equiv e_1(t_1, t_2) \wedge e_2(t_1, t_2)$

[theorem] 

$(\Box P)(t_1, t_2) \equiv \forall t\,(t \in [t_1, t_2] \supset P(t))$

y embeddi$_{M}$ temporal abstractions into the full logic, convenience, efficiency, and express
an be traded off more easily:

Can write (a 'c 1 —* OP .* can use variables.

"Higher-order" notation can be eliminated Jr reification of event predicates.

These techniques of notational engineering are well known and widely practiced in logi
programming. [Editor's note: notational machinery can occasionally be found in metho

MMARY

lamenting the "folk psychology" m

1. Current planning research is part of
of mind and action.
2. Planning consists of two parts:
   a. intention management
   b. reason about actions
3. I. choosing the "logic of action", the role of the logic is the key:
   a. logic for the designer's analysis
   b. logic for runtime deduction

Alternatives to the models of * » psychology need to be investigated.

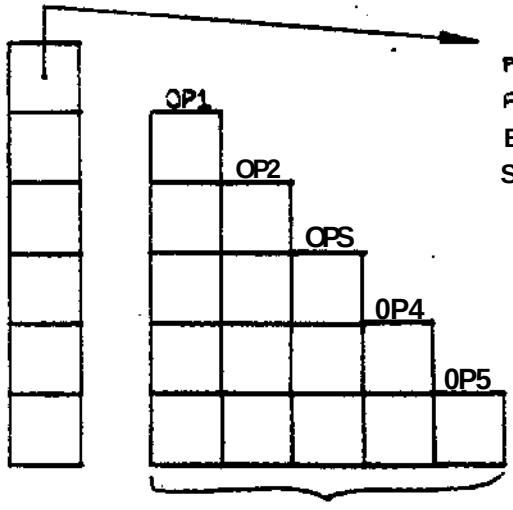
search in planning / Outstanding problems:

1. multiple agents
2. reflexive actions in BDI
3. the management of conjunctive goals, hierarchical plans, the integration of tempo
logic into planning, and reasoning about continuous change.

wo recent planning systems are NONLIN and DEVISOR.

OPI		OP3
PACTS *m* OPTS ADD LIST	FACTS TRUE AS *h* RESULT OP SC4UENCE OPI	
PACTS IN CELL ABOVE MINUS THE FACTS DELETED BY. OPI	PACTS IN OP2'S ADO LIST	FACTS TROE ASA RESULT OP SEQUENCE OPI, OPt
FACTS IM CELL ABOVE MIMUS THE FACTS DELETED 8X OP3	FACTS IM CELL ABOVC MINUS THE FACT5 DELETE* BY O*»3	FACTS IH OP3'S ADD LIST
FACTS ADDED BY OPI AND NOT 50B5C<9U6NTtY D6LETE0	FACTS ADDED BY OfZ AMD NOT SO&SEdtV&tfTL/ DELETED	F*CTS ADDED BY OP3 AMD NOT SUBSEQNENTLY D6U5TBD

6A5IC TRIAAfQtE TABLE



PRE REQUI. ES NEEDED FOR OPt S'JT NOT SUPPU6D BY PREVIOUS OPERATOR »N SiT^ JEKiCS CALL MARKED\

FACTS INHERE NEEDED AS PRE-REQUISITES FOR OPERATOR AT R'IGHT IS MARKED

AUGMENTED TRIAAJGLi

To FIND NEW OPERATOR SBdfUENCe USIMO TRIANGLE TABLES

① RND ROW WHICH HAS FACTS IIK6 OOAL ~>LAST OPERATOR

② MOVE BACK. UNTIL OPERATOR IS FOUND WITH ALL MARKED FACT* T/
... CURRffjJTLY *THUS* ~> FIRST OPERATOR

Planning: deciding on a course of action before acting - John

GPS, 1960, Newell and Simon, CMU
What: Theorem proving, et al
How: Hierarchical planning by ignoring aspects of propositional structure
Overview: One application of the GPS system was to theorem proving, where the program abstracted by first trying to develop a proof that ignored certain aspects of the structure of what it was trying to prove (negation for example) and then extend that to a proof of the un-abstracted proposition.

STRIPS, 1971, Fikes and Nilsson
What: Simple task planning
How: Search over a situation calculus representation
Overview: Did planning for sort of a blocks world robot. Used means-ends analysis with operators that represented actions and had preconditions and effects. Did backward chaining from the goal description. Conflicting subgoals often lead to lots of backtracking or inability to reach a solution.

HACKER, 1973, Sussman, MIT
What: Program (ie plan) synthesis
How: Patching candidate plans by recognizing and fixing "bugs"
Overview: HACKER assumed that subgoals wouldn't conflict, and went ahead and built a program. When subgoals conflicted, HACKER tried to debug its program by determining what went wrong,, and modifying the program. It could detect subproblem ordering conflicts and re-order subproblems to avoid the difficulty.

ABSTRIPS, 1974, Sacerdoti, SRI
What: Simple task planning
How: Search over a situation calculus representation, with hierarchical plan refinment
Overview: Similar to STRIPS, but automatically determined a hierarchy state space features and initially built a plan to solve the "most important" features. It then added details to the plan to accomodate less important features. Feature importance was determined by examining how the available actions could act on the features.

NOAH, 1975, Sacerdoti, SRI
What: Assembly tasks
How: Expanding and modifying a semantic net representation of a plan
Overview: The system used a semantic net of a plan. The plan starts out as a single node in the semantic net. Plan consists of repeatedly expanding the current plan to m refined levels. Embedded procedures in the semantic n provide a rich amount of knowledge about how to refine plan. Critics notice subgoal interactions as expansio proceeds, and revise the plan to avoid them.

INTERPLAN, 1975, Tate, Edinburgh
What: Simple task planning
How: Subplan re-ordering and subgoal promotion
Overview: If solutions to goals conflict, INTERPLAN first tries to re-order them. If that fails, it tries promo the subgoal at which the failure occured to be a goal, and tries to plan again. This allows it to solve some problems for which goal ordering alone is insufficient

Waldinger's system, 1977, Waldinger
What: Simple task planning
How: Non-commitment to ordering, combined with goal regression
Overview: The system attempts to achieve goals one at a time. If achieving a subsequent goal would destroy a goal already achieved, the attempt to achieve the subsequen goal is moved earlier in the plan.

MOLGEN, 1980, Stefik, Stanford
What: Molecular genetics experiment planning
How: Constraint posting
Overview: Molgen builds a skeletal plan, and records conostrai among the pieces. These constraints can propagate thr various parts of the plan. When the plan is refined, constraints may be posted, and constraint propagation begins again.

MOLGEN, 1979, Friedland, Stanford
What: Molecular genetics experiment planning
How: Skeletal plan refinement
Overview: The system picks a skeletal plan, and then refines i It uses extensive expert knowledge about how to pick a skeletal plan and how to refine it.

Hayes-Roth model, 1980, Hayes-Roth and Hayes-Roth, Rand
What: Planning everyday activities
How: opportunistic planning with blackboard model
Overview: application of blackboard model to planning

Chapter 10

Deduction and Inference I

10.1 Points to cover in the discussion

1. Quick historical overview
2. Why do we need logic?
3. Higher-order logic – problems
4. Non-monotonic logics
5. Logic programming
   (a) PROLOG vs LISP
   (b) Logic programming vs theorem proving
   (c) Role of logic programming in AI
6. Resolution theorem proving : compare with natural deduction, strategies for resolution theorem proving
7. Unification and the *occur check*
8. Problems w/ theorem proving as a problem solving paradigm
9. Reasoning with equality– problems
10. Closed world assumptions

11. Modification of MW planner to handle side effects

12. Semantic attachments

13. Circumscription

14. Optimization of logic programs

15. Uncertain reasoning

16. Bayesian updating

17. Semantics of probabilistic schemes

18. BMTP

19. Heuristics for natural deduction systems

10.2 Timeline

1956 LT
1960 Wang's algorithm
1963 Gelernter's geometric reasoning program
1965 Robinson – resolution rule of inference
1968 Advice taker (SIP) McCarthy (proposal)
1968 Fischer-Black (SIP) tried to realize the above pipe dream
1969 Green – QA3; used resolution w/ answer extraction trick.
Control problems as well as the frame problem encountered.
1975 Minsky outlined reasons not to use logic
1975 Hewitt denounced logic
1978 D. McDermott and Doyle – non-monotonic logic
1979 Kowalski book : Logic for Problem Solving
1980 Moore's thesis – reasoning about knowledge and action using possible-world semantics
1980 Nilsson's (AI is applied logic) book : Principles of AI
1983 Science mag. (@Sept) interviews with Minsky and McCarthy:
Minsky: logic only captures part of human reasoning
McCarthy: to understand human reasoning need to develop the right logic and study it

10.3 Why do we need logic?

This is explained in the Introduction section of the handbook chapter on automatic deduction under *Why the deduction problem will not go away.* Claim is that certain notions such as implication, negation, etc. will always be things we want programs to deal with and if we want to do this we need something of the power of logic.

10.4 Higher-order logics - problems

Inherently undecidable. No notion of most general unifier.
Predicate abstraction:
If want to do higher-order logic computations within 1st order logic you run into problems with things like ?(John) given Butcher(John) or Baker(John)

Question : Don't you get this problem in 1st order logic with Butcher(?) given Butcher(Jolu or Butcher(Ralph)?

Let's see how you get an answer out in either case:

Butcher(John) or Butcher(Ralph) [1]
not Butcher(x) or Answer(x) [2]

Resolve 1 ft 2 w/ x <- John:

Butcher(Ralph) or Answer(John) [3]

Resolve 3 ft 2 w/ x <- Ralph:

Answer(Ralph) or Answer(John) [4]

So if the theorem-prover is smart enough to recognize something consisting of multiple Answer literals as an end to resolution, then this works fine. Try the 2nd order case:

Butcher(John) or Baker(John) [1]
not x(John) or Answer(x) [2]

Resolve 1 ft 2 w/ x <- Butcher:

Baker(John) or Answer(Butcher) [3]

Resolve 3 ft 2 w/ x <- Baker:

Answer(Baker) or Answer(Butcher)

So if your unifier is clever enough to handle this then it appears to work. Are there any problems with this? Even if not, you still cannot handle the *Butcher or Baker* predicate within 1st order logic as an object; you can just do this kind of simple question-answering. McCarthy suggests reification of relation names to get parts of second order logic into first order logic.

10.5 Intensional vs Extensional

Classically, an intensional context is one in which the law of substitution of equals for equals does not hold. (Also known as an opaque context.) For example, if we have *The number of planets is even* the falsity of this statement does not change in a normal (extensional) context by replacing *the number of planets* by anything identical to it e.g. *Nine is even.* In an opaque context, however, this does not hold. Thus, in *Galileo believed that the number of planets is even* does not necessarily have the same truth value as *Galileo believed that nine is even.* Intensional contexts are typically associated with belief, knowledge, desire, and similar mentalistic notions.

These contexts are problematic because one cannot perform purely syntactic manipulations to do inference.

Aside: how you regard this and the unrealistic axioms of possible-worlds depends on what you think logic is really doing: modelling people's thinking, providing a tool for mathematical inference, or providing a mode of explanation.

10.6 Possible-worlds

A pocket history: Some time ago people came up with modal logic.(See the book by Lewis for a good introduction to modal logic). The standard one has two modal operators box

and diamond , read as 'necessarily' and 'possibly', respectively. (Another interpretation reads these as 'eventually' and 'infinitely often'.) There was no semantics (i.e. model theory) for this logic until Kripke came up with a scheme which is called possible-world semantics.

P is true iff P is true in all *possible* worlds, where the set of *possible* worlds is defined by an accessibility relation. Bob Moore axiomatized all this into 1st order logic, for reasoning about knowledge and action. One big assumption that needs to be made for this to work is that you know everything deducible from what you know (i.e your knowledge is closed under deduction). This is clearly bogus if we are talking about real human beings. Recently Konolige is trying to retract this assumption by saying that people believe sentences rather than facts, with different models of the extent of closure (wrt deduction) in different people. There are problems with this syntactic approach as well.

10.7 Logic programming

This is: view things as descriptive rather than procedural and get some external mechanism (i.e. the language runtime) to grind through your axioms and get some useful work out of it. There is a tech report by Genesereth and Ginsberg (Hpp-85-??) which is a good description of the concepts and methods in logic programming.

10.8 PROLOG vs LISP programming

Paper by Pereira, Pereira, and Warren (refd. on pg 585 f Handbook's bibliography, vol 3) in 1977 extolling the virtues of PROLOG in comparison to LISP.

10.9 Advantages/disadvantages of PROLOG

1. You get a fixed interpreter (L to R, backtracking), so you don't have to hack up your own.

2. Limited to Horn clause logic.

3. Prolog compiles to run as fast as LISP.

4. The interpreter may be inefficient for your application and there is not a lot you can do about it.

5. **You cannot add additional control structures to take advantage of domain information.**

Chapter 11

Deduction and Inference II

11.1 Resolution

Developed by Robinson in 1964, resolution is a complete rule of inference. Over the years many strategies have been suggested to improve its efficiency. These include:

- **Set of support - One of the two clauses being resolved is the negation of the goal, or a clause already derived this way (from the "set of support"). This method is complete.**
- **Linear input - One of the clauses being resolved is from the original set of axioms (two derived clause never resolve). This is NOT complete.**
- **Ancestry-filtered form - A clause being resolved is either an original clause, or else is an ancestor of an original clause. This strategy is complete.**
- **Unit-preference - A heuristic to help resolution. Pick a single-literal clause (a "unit") as one of the clauses to be resolved. New clauses are thus shorter than the resolved one, and hence "guides" the search to the empty clause.**
- **Breadth-first - Resolve clauses a level at a time. This is complete, but inefficient. It guarantees finding the "shallowest" refutation.**

The main advantage of resolution is that it is complete. However, it is very time consuming; it is difficult to put control knowledge into the process; there is no distinction between facts and goals; and proofs are not *natural* - proofs generated by resolution are sometimes

11.2 Unification

See unification flowchart.

The basic idea of unification is to find a set of substitutions for variables to make two expressions the same. Usually we desire the "most general unifier" - mgu - to make two expressions equal. Unification differs from pattern recognition in that the latter only one pattern has variables, to be matched against a constant expression.

A key part of the unification algorithm is the "occurs check" that guarantees non-cyclic substitutions. An example is attempting to unify P(f(y),y) with P(x,f(x)). A set of substitutions would be (x $\leftarrow$ f(y); y $\leftarrow$ f(x)). However, this is cyclic. Prolog (usually) does not do this occurs check, since is it very time consuming. At the syntactic level, the occur check is necessary to keep the order of quantifiers straight. An example illustrates this.
Fact : $\exists x \forall y\ P(x, y)$
Goal : $\forall y \exists x\ P(x, y)$
Skolemizing Fact : $\forall y\ P(a, y)$
Skolemizing Goal : $P(x, b)$
These unify : $x \leftarrow a, y \leftarrow b$
Suppose however,
Fact : $\forall y \exists x\ P(x, y)$
Goal : $\exists x \forall y\ P(x, y)$
Skolemizing Fact : $P(F(y), y)$
Skolemizing Goal : $P(x, G(x))$
If we unify without the occur check we will have :$x \leftarrow F(y), y \leftarrow G(F(y))$. Without the occur check we would have failed at this point.

11.3 Non-Resolution Techniques, Heuristics

Many different approaches to non-resolution theorem proving have been suggested. Bledsoe's article describes many of them. Popular methods include:

- Forward Chaining - the basic global mechanism for proceeding from the axioms to some desired theorem or conclusion. In theorem proving, forward reasoning (using 'demons') proves useful (see the Handbook Chapter 12, page 99 for a nice example).

- **Rewrite rules - This technique uses rules that specify how to simplify expressions. It can be viewed as directional equalities -. e.g., if x+0 = x , we usually want to simplify from x+0 to x. If a set of rewrite rules has the finite termination property and the unique termination property it is said to be complete. The Knuth-Bendix algorithm can decide if a set of rewrite rules is complete and also extend an incomplete set to a complete one.**

- **Domain specific reasoning - The basic idea is that certain domains may be better suited to special purpose techniques. One example of such a case is in algebra. A special purpose "algebra unifier" would succeed in unifying k+2 with b+5 (k=b+3) where standard unifiers would fail.**

- **Type Information - Typing in used extensively in math and computer science. A theorem prover could use typing to help guide its reasoning process.**

- **Decision procedures and procedural attachment - Quite often it is more straightforward to compute something rather than derive it. Instead of showing 5 X 100 by finding an x such that 5 *prec* x and x *prec* 100, it is easier to simply call a procedure to check 5 -< 100. Likewise, to find x= 17+23, it is easier to compute it by procedure instead of keeping tables for all possible pairs of numbers to be added. This is the idea behind semantic attachments in FOL. Food for thought: how can you prove the correctness of a semantic attachment?**

- **Models and counter-examples - Gelernter is the classic example. Hypotheses that are not consistent with the diagram for the problem are eliminated. Human beings use examples of a theorem to help convince them of its truth. And they conjecture counter-examples when normal attempts at proving the theorem fail. These abilities should also be built into a theorem prover. The Bledsoe article has several examples of this.**

The advantages of non-resolution techniques are that control information can be easily supplied and used, deductions are natural, and such systems are easy to interact with (such as to supply lemmas to help the prover).

11.4 Boyer-Moore Theorem Prover

The Boyer-Moore Theorem Prover (BMTP) has been one of the most successful examples of theorem proving. It has been used in many different domains on substantial proofs, representing everything in a LISP-like functional language without quantification and the like. For each axiom, it proves totality, producing an induction schema in the process to be used in the proof itself. Proofs are tried in levels: first rewrite rules (some discovered by the theorem prover) are used to simplify the axiom, with no backtracking; if still unproved, it tries rewriting with provided axioms; if still unsolved it finally tries an induction schema. The user may supply lemmas to assist in the search for a proof. Proofs using the BMTP have been done for the Boyer-Moore fast string algorithm and prime factorization. The BMTP is not complete.

11.5 Problems with Theorem Proving as a Problem Solving Paradigm

Problems with theorem proving as a problem solving paradigm are discussed in Green's paper. The two major issues are that due to its generality, theorem proving is often slow. It also requires the domain to be formalized, usually in logic. This is not always easy or possible.

11.6 Reasoning with Equality

(taken from lecture notes of CS400b, taught by Dr. Robert Moore of SRI, Fall 1983)

Using the axiom $\forall$ x = x does not work too well. In common-sense reasoning, we want equality reasoning to reason about individuals. For example, if we have the axiom $\forall$ x (P(x) $\Rightarrow$ (x = A or x = B or x = C), and we have a fact P(G0037), we would like to be able to conclude that G0037 is one of A,B or C. Also handling $\exists$ by Skolemization requires reasoning about the equality of the arbitrary function names introduced in the process.

There are at least three different uses of the = symbol. One usage is to denote simplification (e.g -(-x) = x). Equalities also serve to define computations (e.g reverse(x.y) = reverse(y).x). Equalities also state abstract relations between quantities(e.g $x^2 + y^2 = z^2$).

Paramodulation is a technique for doing resolution theorem proving with equality built

in.

```
Fact : P(t1)
Fact : t1 = t2 or Q
----------------------------
Fact : P(t2) or Q : resolvent
```

Note that the equality is used in one direction.

11.7 Uncertain Reasoning - Bayesian Updating

Uncertain reasoning occurs when either the reasoning process is only approximate, or the axioms being used are not totally believed. Statistical techniques such as Bayesian updating suggest methods to handle these types of reasoning tasks. PROSPECTOR used a formulation of Bayes rule that used odds and likelihoods instead of probabilities. The basic formula is O(H|E) = kO(H) where O(H) is the odds of H being true, O(H|E) is the odds of H occurring, given that E is known to be true, and k is the likelihood of E given H, i.e., the ratio of the probability of E being true, when H is true, to the probability of E being true when H is false. More is in the Duda, Hart, and Nilsson article. Other statistical methods include the certainty factors of MYCIN and the Dempster-Schafer theory of Uncertainty. Read the MYCIN book for more details on the certainty factors in MYCIN.

11.8 Approaches to non-monotonic reasoning

There have been two basic approaches

- Extend the logic system in various ways.
  - McCarthy: extend the ontology of the logic: create *lack of oars* as an object in your domain of axiomatization.
  - Reiter: extend the logic by having a theory consist of the usual set of axioms and a set of defaults. A default has pre-requisites, a consequent and a set of assumptions.
  - McDermott and Doyle: add the operator M to the logic, where Mp means that p is consistent with what is known.

Reiter, McDermott and Doyle are trying to provide an inference of the form *If p is consistent with what you know, assume* p, where consistency is defined in terms of provability in first order logic, which is semi-decidable.

- **The Meta Approach**

  — **Winograd: resource limited reasoning: p is consistent means that p or (not p) can be concluded on the basis of some bounded computation.**

  - **Doyle: TMS: A truth maintenance system is a formal system of constraints among objects representing theorems in a first order theory. Consistency is judges wrt a finite set of theorems and constraints.**

11.9 Reiter's framework for studying default reasoning

Default reasoning denotes the process of arriving at conclusions based upon patterns of inference of the form: *In the absence of information to the contrary, assume*..... We translate that to read *If p cannot he deduced from the database, then assume*.... Examples of such reasoning include:

1. **Default assignment to variables**
   If Unprovable(Hometown(x) =• Palo Alto) then Hometown(x) = Palo Alto
2. **closed world assumption**
   Consider the following IS-A hierarchy (reproduced at the end of this chapter). A network interpreter does deduction by traversing links. The theorem prover which works on the logic representation and the network interpreter work isomorphically in the deduction of positive facts. Consider proving the statement *Not(Reptile(Fido))*. The network interpreter would prove this by noticing that there is no directed edge from the *dog* node to the *reptile* node in the hierarchy. The theorem prover will fail because there is nothing in the logic representation that states that the categories mammal and reptile are disjoint. We need additional facts like V x mammal (x) => not (reptile(x)). To make the theorem prover's behavior isomorphic to the network interpreter, we need to augment the theorem prover with the additional inference rule:
   If Unprovable P(x) then Not(P(x))

An example of a closed world default is: if a flight is not listed in the airline database, it is reasonable to conclude that it does not exist.

3. **frame default for causal worlds**
   The frame problem stems from the need to represent those aspects of the world that remain invariant under certain state changes. In a first oder representation, it is necessary to explicitly represent all the invariants under ail state changes. Instead we can use the following rule of inference to get the same effect:
   If Unprovable (Not (P(x))) then P(x)
   Note the similarity between the frame default(FR) and the closed world default(CW). FR permits inferring a positive fact by failing to prove its negation, whereas CW permits inferring a negative fact by failing to prove its positive counterpart.

4. **Exceptions as defaults**
   The classic example is: unless there is information to the contrary (e.g Tweety is an ostrich, Tweety is dead, etc.), conclude that Tweety can fly given that Tweety is a bird.

5. **Negation in PLANNER**
   Negation in PLANNER is interpreted as failure to prove. This is true in the case of Prolog too.

Some of these defaults provide clear representational and computational advantages over corresponding first order theories.

11.10 Optimization of logic programs

This is a relatively unexplored area of research. Some techniques have been developed to compile programs in subsets of first order logic (e.g Horn clause subset) into efficient assembly language code.(Warren), recent work at Stanford (VanGelder) addresses compilation of logic programs in NAIL into relational algebra. Rewriting logic programs into more efficient logic programs (i.e target language for the compilation is logic itself) is open territory. Excellent work on ordering conjuncts in rules and controlling recursive inference has just been completed at Stanford (David Smith, 1985).

There is also work going on at Utah (Reddy) which concentrates on compiling logic programs into functional programs.

11.11 Semantics of probabilistic schemes

Nilsson made an interesting start in this direction in his "Probabilistic logic" paper. This is published in the AI journal.

11.12 Handling side-effects in the MW planner

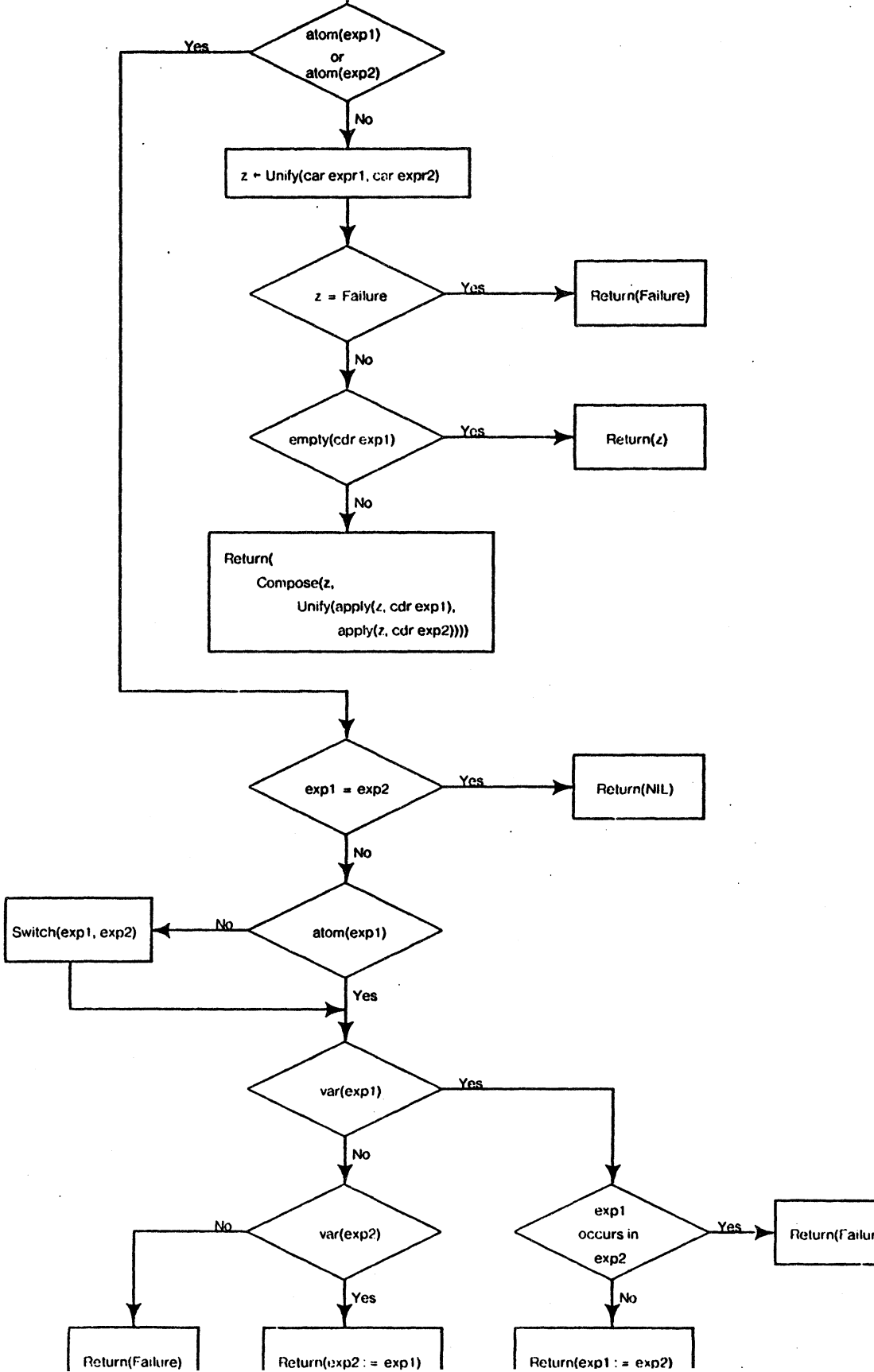
Jeff Finger at Stanford has developed such a planner as part of his PhD thesis. Here is a short summary of his technique called residue. Residue is a deductive procedure for design synthesis. It attempts to find a set of facts R, such that when added to the initial world model W, the design goal G is included in the deductive closure of Union(W,R). In addition, Union(W,R) must be consistent and the facts comprising R must be members of a constraint language of implementable design constraints.

In the above approach, both finished and unfinished designs are expressed as a set of facts *about* the design. This enables residue to reason about the design even before it is completed. By forward reasoning from design decisions already made, residue can preclude parts of the design space from ever being generated, if the design is to remain consistent. In addition, by using logic to express design decisions, residue has a huge, pre-existing vocabulary of constraints, thus allowing arbitrarily fine-grained decisions to be made about the design. In other approaches, unnecessary backtracking is often the result of the inability to express sufficiently fine-grained design decisions.

11.13 Some more questions on Deduction and Inference

- Show how forward chaining and backward chaining are special cases of resolution theorem proving.
- How does the BMTP construct induction hypotheses?
- What are the relationships between Prolog and Planner?
- What are the main approaches to uncertain reasoning?

- What is a complete inference method?
- What does the closed world assumption(CWA) buy us in database query evaluation? What CWAs are made by concept learning systems?
- What is the main idea behind the use of connection graphs in resolution theorem proving?
- What is circumscription?


atom(exp1)
or
atom(exp2)
Yes
No
z ← Unify(car expr1, car expr2)
z = Failure
Yes
Return(Failure)
No
empty(cdr exp1)
Yes
Return(z)
No
Return(
Compose(z,
Unify(apply(z, cdr exp1),
apply(z, cdr exp2))))
exp1 = exp2
Yes
Return(NIL)
No
Switch(exp1, exp2)
No
atom(exp1)
Yes
var(exp1)
Yes
No
No
var(exp2)
exp1
occurs in
exp2
Yes
Return(Failu
Yes
No
Return(Failure)
Return(exp2 := exp1)
Return(exp1 := exp2)

Chapter 12

Guest Session on Deduction and Inference

Guest : Mark Stickel, SRI

12.1 Important Developments in Automated Theorem Proving

Mark Stickel first discussed what he considered to be the two most important developments in Automated Theorem Proving (ATP) in the last decade or so.

- Prolog and Logic Programming
  While logic programming is distinct from ATP, the two are still closely related. The Prolog restriction to Horn Clause logic is really a severe limitation from the standpoint of ATP. The main contribution of Prolog and similar systems is that they are *usable*. There was a lot of earlier work (such as Hewitt's Planner) which wasn't as efficient as Prolog in performing deductions. Warren's fast DEC-10 implementation of Prolog made people take the notion of unification/resolution based AI programming languages seriously.
  The implementation technology from Prolog will be useful to the ATP community. Hopefully, it will lead to fast model elimination techniques that can be

(at the Argonne National Labs) is being adapted to use the same type of compiled pattern matching as PROLOG, but for general resolution.

One problem still to be tackled in PROLOG is its minimal specification of control. The current types are left-to-right ordering (within the clauses of a rule), top-to-bottom ordering (for the rules themselves), and the 'cut' operation (which allows pruning of the search tree).

Another area that needs further attention is how to reason about equalities. TABLOG and EQLOG are extensions of PROLOG that handle equality reasoning.

- Knuth-Bendix procedure

The Knuth-Bendix algorithm is a decision procedure for a class of equational theories(AI Handbook, Vol 3, pp 98-99). Although this technique is not always applicable as a procedure for handling equality reasoning, it is very powerful when it does apply. It can be shown that the technique can be extended to 'inductive completion'. Unlike the Boyer-Moore theorem prover which uses explicit inductive schemas, this technique does an 'inductionless induction'. So far, the applications that Knuth-Bendix has been applied to are in more mathematical domains, such as deciding problems in abstract math. The procedure may be regarded as a general framework for expressing other algorithms. It is described more fully in the Bledsoe reference on the reading list.

A classical equality reasoning technique, called paramodulation, allows one to conclude p(B) from A=B and p(A) and is similar to the 'superposition' operation in the Knuth-Bendix procedure, though Knuth-Bendix only allows this replacement in a single direction.

There is still a problem with the large number of operators that are commutative. Since in Knuth-Bendix everything that is done is a reduction, and termination is required, you cannot have (x+y —• y+x) since it does not simplify the expression. A solution to this is to generate equivalence classes modulo the commutativity.

One of the most important future extensions of Knuth-Bendix is conditional reductions. Without it, Knuth-Bendix only could be applied to groups and rings but not to fields; this is because without conditionally it couldn't handle division (because of the non-zero division condition).

A second useful extension would be to enable Knuth-Bendix to work for infinite equivalence classes (non-terminating rewrite rules).

12.2 Significant Open Problems in Automated Deduction

- Control Problems
  We need to have more control specification then is currently available in Prolog. There will not be much mileage gained from just tinkering with current deduction procedures.
- Reasoning With Equality
  Equality reasoning is not yet done with adequate efficiency. Principal present techniques are: use of equality axioms, paramodulation, and E-resolution(Morris, IJCAI 69). E-Resolution allows such things as concluding that B$\neq$A from p(B) and not(p(A)).
- Use of types
  One method of trying to get better efficiency is through the use of *sorts* or *types* (as in Weyhrauch's FOL). See also the AAAI articles of Walther and Cohn on this subject.
- Directions for Future Research
  It appears that in order to achieve success in ATP, we will have to incorporate a whole battery of methods in our programs – it is not a good idea to depend on a single technique. Resolution can serve as the basis or weak method for a program, but we should also have decision procedures to direct the resolution.(Nelson and Oppen, Shostak) We don't want all our methods to be working "on the same level".

  For example, suppose we are given a rule for transitivity of less-than, and we would like to show a$\prec$b. If we use ordinary resolution, negate the clause, i.e. not(a$\prec$b), and try to proceed, we will find that there are an infinite number of resolvents: we can resolve with the transitivity law and introduce x, yielding a$\prec$x$\prec$b, and continue to insert new variables into the inequality ad infinitum. The defect is that we have expressed all our knowledge at the level of axioms.

The proof goes through if we instead build in transitivity as an inference rule. In that case we would not get lost in an infinite loop because we would not be able to say anything until we had both of the literals a-<b and b-<c.

- Hyper-resolution

Hyper-resolution was devised in the late 1960's (see, e.g., Chang and Lee). It differs from ordinary resolution in that multiple clauses are resolved in a single step. Example:

```
Al or Bi }
         }          not(Al) or not(A2) or ... or not(Am) or C
A2 or B2 }(electrons)            (nucleus)
  ...    }
Am or Bm }
```

BI or B2 or ... or Bm or C (hyper-resolvent)

Each clause on the left (Ai or Bi) is called an *electron;* the clause in the upper right (not(Al) etc.) is called the *nucleus.* Note that hyper-resolution resolves m+1 clauses in one fell swoop, whereas regular resolution would take m steps to derive the same result. The trade-off is that all of the combinatorics of selecting parent clauses to resolve is compressed into one step in hyper-resolution; but the cost of generating all the intermediate result clauses is eliminated.

Hyper-resolution is used by the Argonne group.

A current innovation is the idea of theory-resolution. The basic idea is the recognition of unsatisfiable clauses wrt a theory. In full theory resolution, you can make resolutions like:

a<b or A, b<c or B, not(a<c) or C *«> A or B or C

This is related to hyper-resolution, since it could have been accomplished by hyper-resolution using not(xXy) or not(y-<z) or a^c as a hyper-resolution nucleus resolving on all three of its literals.

In partial theory resolution, you can do the following:

```
a<b or A, b<c or B ==> a<c or A or B
```

Linked-inference is a more concrete variation of this in which the 'theory' (in this case the theory of inequality transitivity) is embodied in clauses that are searched in a very restricted manner.

12.3 Strong vs. Weak Methods in AI

The trade-off between strong and weak methods is one of efficiency for generality. Linear programming is a strong method and it has narrow applicability. Weak methods in ATP include resolution and GPS-style reasoning. They use little knowledge about the domain. The 'power' of an implementation of a weak method is in how easy it is to add control information to it.

One way to combine strong and weak methods is to use a blackboard approach, where wffs are posted on the blackboard. In general, it is hard to combine strong and weak methods. The handling of multiple overlapping domains is an open problem.

12.4 Problems with Prolog

Prolog was originally intended as a computerized incarnation of logic, and indeed most of the language is strongly logic-based. Yet there are a few important non-logical features, in particular the cut operation, the i/o operations, and the assert and retract commands for maintaining a global database.
Q: Why were these features included in Prolog?

A: The cut operation is often used for efficiency – it allows you to prevent Prolog from continuing to backtrack and look for other solutions, when you are sure it is not worthwhile to do so. Its use is less defensible when the cut operation is used to eliminate solutions so that the program has different semantics (and not different performance when the cuts are removed).

On the other hand, i/o, assert, and retract are inherently non-logical operations (you can't "backtrack" after reading an input). Nevertheless, programmers have found i/o and database operations to be indispensable, and there's no obvious way to reconcile them with the notion of programming in logic.

Another problem with Prolog is that it doesn't support a modular rule base or the typing of variables/rules, EQLOG is intended to be a remedy for this. EQLOG does equality reasoning on abstract data types by a 'narrowing' operation which is a sort of paramodulation with orientation. Alternatively it can be thought of as rewrite-rules with unification.

Lastly, Prolog has the drawback that it searches for proofs purely by depth-first search. This can mean trouble if it hits an infinite search path before it reaches a correct (finite) one. This could be fixed by simulating breadth-first search, or by implementing some sort of parallel searching scheme, as in PARLOG (which was designed with parallel machine architectures in mind).

If to PROLOG we added the model elimination and reduction operation (a reasoning by contradiction rule), used contrapositives of rules, performed sound unifications (including the occur check), and used a complete search strategy, we would have a complete prover for first order predicate calculus. Many of the techniques for implementing PROLOG efficiently would apply to this complete extension of PROLOG as well.

Chapter 13

Expert System Principles I

Even though this is a session on the principles of expert systems, it is extremely useful to understand and explain these in the context of concrete examples. Hence the references to real systems in this discussion. The class was not expected to have read about specific systems (we would do that later under 'Expert System Applications'), and it was this author's duty to make references to these examples to motivate reading about them later. The reason for this choice was that the reading for this week is sufficiently intense – and the top down approach of learning about the abstract principles first and then instantiating it with examples was chosen over the bottom-up alternative where specific systems would be studied before we aggregated the principles on which they were based. The systems in Chapter 7,8,9 of the AI handbook are presented in great detail and it was considered wise to give an orientation to the class before letting them read those chapters.

- **What is an expert system**

  The Buchanan and Duda paper identifies three dimensions that characterize an expert system. An expert system is a computer program that provides expert level solutions to important problems and is

  * heuristic
  * transparent
  * flexible

* expertise : high performance, efficiency
* Reasoning by symbol manipulation
* Complexity
* Reasoning about self : explanation
* Type of task : interpretation, diagnosis, prediction, instruction, monitoring, planning and design.
* Use of fundamental domain principles and weak reasoning methods.

We decided that the type of task cannot be a defining measure for an expert system. Complexity is hard to measure. NM reasoning is something that humans are so good at and seem to do with relative ease, yet it is a major challenge to build a machine that can retract assumptions and reason with them (see however de Kleer's latest work on Assumption-based TMS). Symbol manipulation in itself does not constitute a defining characteristic. The work on optimizing compilers is symbolic in nature. The earliest expert systems (MACSYMA and DENDRAL) had no concept of self and MYCIN has a minimal explanation facility. This cannot be thus used as a yardstick for deciding if a program is an expert system or not. The three characteristics proposed in the Buchanan and Duda paper are more reasonable.

How does an expert system differ from a conventional program? Is a chess-playing program an expert system? Greenblatt's chess player : fails on the transparency and flexibility count. What of Wilkins' PARADISE?

– **Key ideas and assumptions in rule-based expert systems**

* IF-THEN rules for the representation of knowledge
* Modus ponens is the primary rule of inference
* Variations in this framework (CF's) were to cope with the complexities of the real world.
* Knowledge is power assumption. I.e weak inference methods would suffice if enough knowledge about domain were present.
* Separation of knowledge from the control. This factorization in itself does not lead to modularity and flexibility. (see Clancey's paper on the Epistemology of explanations).

* No explicit attempt to model human problem solving, but the system uses the knowledge and control that human experts do.

- **What sorts of information are hard to capture in a rule based framework**

  Read the Davis and King article in the MYCIN book. A pointer to it is the reading list for KR. Hint : Try writing a production rule system for addition.

- **Short history of evolution of expert systems**

  A very crisp review is in Section 2 of the Buchanan paper 'New Research in expert systems' which appears in MI 10.

- **Key components of a rule base expert system**

  See the diagram on page 9 of 'Principles of Rule-Based Expert Systems', the main parts of EMycin: interaction handler knowledge acquisition aid explanation subsystem

  BES p. 17 diagram shows the anatomy of an "ideal" expert system

- **What is the character of a domain for which an Expert System is appropriate?**

  This is a hard question. The systems we have can be considered as data points for drawing an answer. BES tries to answer the question, by identifying classes of problems for which an expert system solution is possible, (see Chapter 1). Here are some of the dimensions that the class came up with

  * Domain continuums to consider: analysis/synthesis, level of formalism/mushiness.
  * Expert systems do well with "human-created" systems.
  * For purely formal systems an algorithmic approach is probably better (if an efficient one exists, cf Travelling salesman problem - the algorithmic solution is computationally expensive).
  * The domain must be narrow enough (so all the relevant knowledge needed for problem solving in that domain can be captured).
  * There must be a way to combine different experts' opinions (an open research issue in expert systems).

- **Limitations of Expert Systems**

  * Do not know their limits, fragile behavior at the boundaries

* Lack common sense
* Have no independent means of verifying their own conclusions
* Narrow domains
* Limited expandability
* Single expert as Czar
* Limited assumptions about problem and solution methods

See the "New research on expert systems" paper by Buchanan to get some idea on proposed solutions to these problems.

- **For each expert system you should be familiar with**
  * Task
  * How expertise was encoded
  * Control strategy used
  * How extendable it is/knowledge acquisition tools used
  * Explanations
  * Utility
  * Reasoning under uncertainty
  * Validation

  Comparison questions are also important. E.g the representational; formalisms of PROPSPECTOR and MYCIN. The uncertainty management schemes in PROSPECTOR and MYCIN were quite different. What are the differences? Why?

- **Architectural issues**
  See BES chapter 4, p. 91

- **Representational formalisms in Expert Systems**
  Read Buchanan and Duda and then BES chapter 9. What are the key issues here? What work is going on in this area?

- **Inference methods in Expert Systems**
  See Buchanan and Duda section 3 for a clear exposition of the basic inference strategies used (forward and backward chaining). The complications that result from uncertainty are also well explained here. Name an expert system that uses constraint propagation.

- **A short note on LOOPS**

  LOOPS is a vehicle for knowledge programming (or techniques for representing knowledge in computer programs). The underlying philosophy is that different paradigms are appropriate for different purposes – so LOOPS provides all of them and lets the programmer mix and match methods based on his own cost metrics (cost of learning, cost of modifying, cost of debugging, cost of running). The four programming paradigms in LOOPS are

  * Procedure oriented programming
    or INTERLISP-D. This is the base for all the other formalisms provided.
  * Object oriented programming
    Here information is organized in terms of objects which combine both instructions and data. Objects communicate by messages. There are message protocols for interpreting messages. Inheritance in a class lattice enables the specialization of objects.
  * Access oriented programming
    is useful for programs that monitor other programs. Its basic mechanism is a structure called active value, which has procedures that are invoked when variables are accessed. E.g to drive gauges that display values graphically.
  * Rule oriented programming
    Specialized for representing decision making knowledge in a program. Rules are organized into rule sets which specify rules, control structure and description of rules. There are techniques for factoring control info from rules and also a dependency trail facility which provides a mechanism for explanation and belief revision.

- **MYCIN and common sense**

  * MYCIN cannot predict the consequences of the use of medication that it prescribes.
  * It cannot consider a hypothetical treatment.
  * It has no sense of time, causality. Does not have hospitals, patients and death in its ontology.
  * It does not know its own limitations.

  - It has no facility for the integration of naive physics and technical medical knowledge.

- When will machine learning be cost effective for knowledge acquisition
  Open question.
- What is the blackboard model? What are its advantages and disadvantages?
  See Barbara Hayes-Roth's paper on blackboard systems in the recommended reading.
- What is knowledge engineering? Who coined this term?
  Feigenbaum used (and defined) this word in the 1977 invited address at IJCAI.
- Compilation of knowledge bases.
  What is it? What work has gone on in it? What are the research issues?
- An expert system that uses default reasoning
  Name one such system. What other existing systems would benefit from it (i.e how would their capabilities be extended)?.
- Turing Test and Expert Systems
  Really a question on validation issues in expert systems. Look at Chapter 8 in *Building Expert Systems.* The main issues are : the purpose of evaluation, what characteristics to evaluate, when to evaluate and (of course!) how to evaluate.
- Why are MYCIN's rules no help to medical students
  - They do not separate key factors from control setting factors.
  - Omit causal mechanisms.
  - Embody a strategy of medical diagnosis that is not made explicit.
- An expert system is good for
  - A classification problem in which data are explained or covered by hypothesis from a pre-defined data set.
  - A problem of sufficient difficulty that practitioners turn to text books for advice.
  - A problem of sufficient difficulty that experts require time for reasoning.
  - Narrow enough that a knowledge base can be built in reasonable time.
  - Closed world - the vocabulary is small and bounded.

- **Sources of uncertainty in a rule based system**
  The data, the rules and the conceptual framework. The last mentioned is the hardest to detect and recover from. The usual method to recover from this is redundancy. See Buchanan and Duda's paper (section 4).
- **Design criteria dictated by user engineering issues**
  See Shortliffe's paper in the Webber and Nilsson collection.
- **Why should we abstract control knowledge in expert system design?**
  Read Clancey's paper in the recommended readings.
- **Issues in meta-knowledge and meta-reasoning in expert systems**
  You need to know work to date in this area, sources of meta-knowledge in expert systems, examples and use of meta-knowledge. Most of this is to be found in Chapter 7 of *Building Expert Systems.* You should also be familiar with MRS (read the Meta level architecture paper by Genesereth et. al.) and Weyhrauch's FOL system.
- **Languages and Tools for Knowledge Engineering**
  There is a whole inventory of these systems in Chapter 9 of *Building Expert Systems.* Comparison questions are important. Also some knowledge of the limitations of these systems (e.g can you program a cryptarithmetic puzzle solver in EMYCIN?).

Chapter 14

Juest Session on Expert ystem Principles

Guest: Bruce Buchanan, Stanford University

Buchanan was our guest for this session. We had a mock qua! session, where Buchanan asked questions in rounds. Most of the questions were on the topic or the week.

What is an expert system?

The following criteria were taken to be the defining ones.

- **High performance**
- **Transparency**
- **Use of symbolic reasoning**
- **Utility**
- **Flexibility and Modularity (extensible KB)**

Vhy is expert systems research part of AI?

Vhat is the distinction between expert systems and AI programs in general?

tnhat are the defining characteristics of an expert system?

/hat is the approximate size of the MYCIN data base? How many organisms id it know about and thus how large was the number of diagnostic possibilities had to consider?

6. What issues has expert systems research precipitated?
   Research in representation, inference, knowledge acquisition, explanation tutoring and problem solving.
7. Why are expert systems important for the cognitive scientist?
8. What has expert systems research revealed about the nature of expertise?
9. What constitutes an explanation? What kind of explanations does MYCIN give?
10. What are framework (expert system building) systems? Give examples. OPS5, EMYCIN, KAS, UNITS, KEE, AGE, LOOPS, MRS, PROLOG. Look at the Buchanan/Duda paper for a way of organizing these systems according to the representational framework they use. EMYCIN and KAS are called framework systems.
11. What is the distinction between a framework system and a programming language?
12. What is the special feature of the framework system EXPERT? It is written in FORTRAN and is a simple forward chaining system.
13. What is the distinction between a framework system and a programming language?
14. What are some of the landmark papers in expert systems research?
15. What is the major lesson from PUFF?
16. How does search figure in expert system work?
17. How should we measure the complexity of a task or a system? Is the number of rules in an expert system a good measure of its complexity?
18. What features of a problem make an expert systems solution to it less likely?
19. What is PANDEMONIUM? Why is it important for data fusion problems?
20. Why do we need to worry about inexact inference?
21. What are some of the methods proposed to deal with inexact inference?
22. Explain the following terminology
    - Rule-based
    - Knowledge-based
    - Knowledge engineer

- **Trigger**
- **Meta-rule**
- **Deep vs shallow models**
- **Transfer of expertise**
- **Certainty factor**
- **Inference net**
- **Knowledge cliff**

23. **How can expert systems be validated? How have some of them been validated?**
24. **Why is knowledge engineering hard?**
25. **What are the major trade-offs in the design of expert systems?**
26. **What are the open issues in expert systems research?**
27. **What are the common criticisms of expert systems work? What research effort is aimed at this?**

Chapter 15

Learning I

15.1 Outline of discussion

1. Definition of learning.
2. History of research in learning.
3. Learning Techniques.
4. Research Issues.
5. Summary of learning systems.
6. Other knowledge acquisition methods.

15.2 Definition of learning

(with a 3 min. time limit !).

The performance(or behavioral) definition: Learning = adapting behavior to the environment resulting in iterative improvement to performance.

Problems with this definition: defines learning in the context of a performance system. This accounts for the *skill refinement* type of learning. Learning in humans (and we hope in machines) encompasses a much wider range of behaviors than skill refinement. Also this definition gives us no clues as to how this behavioral change can be obtained.

The knowledge definition: Learning = acquiring *new facts*, i.e facts that cannot be deduced from what is known previously.

Problems with this definition:

For a definition of learning, see Chapter 1 of the book *Machine Learning*. Also see Simon's article in that book *Why should machines learn?*. Try answering some of the questions that he raises there (especially people intending to do a thesis on machine learning).

15.3 Techniques

Following the performance definition, we can split up the various learning techniques as follows: (using the classification in the AI Handbook)

1. **Rote learning.**
2. **Learning from examples**
3. **Advice taking = learning by being told**
4. **Parameter adjustment and Connectionist learning**
5. **Discovery**
6. **Learning by doing (experimentation)**
7. **Learning by Watching**
8. **Learning by Analogy**
9. **Learning by Reconceptualization**

- **Rote learning**
  Definition:
  Given input X and output Y, remember association (X.Y) so that next time X is encountered Y can be looked up rather than recomputed.
  Issues :
  1. **space/time tradeoffs**
     Also known as *the recompute/store* tradeoff. For example, it is silly to store away addition tables since it is easier for the machine to just recompute results.

2. Cost of lookups
   Lookups also involve 'effort' (time). There is a lot of work in the database world to make lookups of facts very efficient by devising good indexing methods.
3. Selective Forgetting
   A rote learner needs to throw away associations that it will never need. An LRU scheme can be used to throw associations away. In case, it is needed, it can always be recomputed. Throwing the fact away, makes room for storing newer associations in memory bounded learners.
4. Updating associations
   A rote learner has to contend with a frame problem: it has to ascertain that the value that it has stored away is still current (a check would necessitate recomputation, which was what we wanted to avoid in the first place!).

Example system :

One of the numerous instantiations of Samuel's checkers programs saved board positions it had already seen with their evaluation on a huge tape using clever indexing schemes.

Rote learning can actually result in a performance improvement. In a checkers program, for example, saving evaluations of board positions can increase the effective search depth because the saved evaluation need not be a simple static one, but can be based on the static (or saved) evaluations lower down in the game tree.

- Learning from examples:

  Question: Why has almost all of the machine learning work in the early 70's to the early 80's concentrated on *learning from examples*?

  The issues in learning from examples are well covered in Chapter 14 of the AI Handbook. The two space model of Simon and Lea (pp361) can be further refined to include the relationship between instance selection and interpretation, which are inverses of each other. This requires making explicit the *articulation theory* between the instance space and the rule space.

  The single-representation trick essentially translates to the requirement that the instance space be a proper subset of the rule space. This eliminates a translation step in between.

There has been new progress on the problem of new terms. Paul Utgoff finished a thesis at Rutgers in October 1984, that explained generation of new terms like *odd* and *even* in the context of the LEX symbolic integration system. The method adopted was generating a problem solving trace, and then propagating the solution conditions through the trace, to find the weakest pre-condition under which the same problem solving steps could be used. If the weakest precondition could not be expressed in the vocabulary used by the problem solver, a new term (and definition) was created. More work on this problem has since been done by Pearl[IJCAI85], Kibler and Porter[IJCAI85]. The new term problem still remains essentially open.

Comments on some systems that learnt from examples:

Winston's arch program (1970)

Winston presented the learning program with a well chosen sequence of instances of arches, each represented as a semantic net. The learner had the generalization hierarchy on its concept language (also represented as a semantic net). It matched its current concept description, with the currently presented example, and generalized (or specialized) the nodes and links using the generalization hierarchy that it had. It backtracked, if the current instance could not be made consistent with its current concept description using any of the operators that it knew about. Mitchell characterized the search behavior of this program as a depth first search in the space of all expressible generalizations. Winston's learner had to maintain all the instances it had seen, because it had to restore consistency with whatever it had seen, every time it had to backtrack to a new candidate concept description.

Its main limitations were:

- It needed a well-structured set of learning examples, otherwise it would lead to hopeless backtracking behavior.
- Misses had to be near-misses, so teacher had to know the details of the representation of the problem domain in the system to be able to generate good training sequences. This requirement provided a solution to the credit assignment problem.

What is a near-miss?
All features but one are the same. Note: *highly* representation- dependent. WRT Winston's program if an instance labelled a near-miss is not really a near-miss, the program does nothing, following Minsky's "no-guessing principle", i.e. "when in doubt as to what to learn, learn nothing."

META-DENDRAL
Meta-Dendral learns rules of mass spectrometry from example mass-spectrum molecular structure pairs. It used a half-order theory to constrain search (i.e. model-driven generate and test) in the rule space. The search for rules proceeded in two phases:
RULEGEN – using positive examples only, produce possible rules. This was a coarse search.
RULEMOD – using negative examples to refine results of RULEGEN; a finer-grained search.
This was a powerful demonstration of the use of AI techniques for acquiring specialized knowledge about a domain from a large number of examples. Food for thought: How good is the Meta-Dendral paradigm for learning common-sense knowledge?

Mitchell's version space algorithm
Its main merit was that it provided a new framework in which to view learning systems that learnt from examples. Generalization was formulated as a search through a space of descriptions guided by the training instances provided by the teacher. It is a data-driven algorithm that keeps all hypotheses consistent with the instances that it has seen. It usually operates on a space of descriptions that is restricted by the use of linguistic bias (only conjunctions can be learnt). The algorithm is a least commitment algorithm and learning occurs by successive refinement of the *version space* of a concept. Its main limitations are that it does not handle noisy instances. Mitchell suggests a method in his thesis for handling this: this has theoretical appeal, but does not seem to hold practical promise. Utgoff has extended the version space method to allow relaxation of its strict linguistic bias. Mitchell has extended the essentially syntactic nature of the generalizations made by the version space algorithm into his new *Explanation based generalization*[Mitchell et. al. 1985]. The learner is given an unoperational

definition of what it is to learn (the *goal of* the learning process), and it produces an operational definition by using a domain theory (assumed to be correct and complete). The operational definition is proved to be equivalent to the unoperational one using the axioms in this theory. The proofs are generalized to get the most abstract version of the operational definition.

SOAR

SOAR generalizes by *chunking.* This concept is best explained in [Rosenbloom85]. Roughly, this corresponds to caching away macro moves in the solution space. SOAR takes its problem-solving trace and generalizes on the conditions under which that sequence of operators is applicable, by assuming that the relevant features to generalize on, are exactly those that occur in its short term memory. Sometimes this is not quite correct, and under those circumstances, SOAR overgeneralizes.

BACON

BACON takes in as input, tables of data relating several features and attempts to find the higher level relationships between these features (e.g Ohm's Law from data about V,I and R). BACON has heuristics that help it search the space of all possible algebraic relationships between the parameters starting from the simpler forms to the more complex forms. BACON could form part of a discovery system, that sifts out the relevant parameters to it, so that it can find high level regularities in that data. It will be an interesting exercise to characterize the sort of bias that BACON uses, and check if it can be generalized.

FOO

Foo was one of the best (probably, also the only direct attempt) attempts to build an *Advice Taker.* Foo works in a card domain: the game of Hearts. It operationaiizes high level advice like *Do not take points* into strategies for playing a hand. This is extremely difficult, and Foo used about 200 domain specific transformation rules to do this. The control problem that arises in the application of these rules is horrendous and is yet to be automated.

Bias

Mitchell defined bias to be any information (other than consistency with the observed instances) that the learner may bring to bear to choose between candidate generalization. Two types of bias have been used (and studied) in inductive

learning: *procedural bias* (bias in the learning algorithm, e.g in AQ11, the first disjunct covers the maximum number of positive instances, because the algorithm always tries to add a positive instance to that cluster first), and linguistic bias (bias in the language being learned, e.g the version space algorithm has a restricted vocabulary, only conjunctions can be represented in the learner).

- Parameter adjustment

  From a set of examples, each associated with a measure of goodness, and a set of features, find the coefficient of a linear (or non-linear) polynomial whose terms are those features. Coefficients are tuned iteratively. Sometimes terms have to be changed. cf. Samuels' checker player. e.g. Yet another of the many instantiations of Samuel's checkers program

  Credit assignment in this program was via

  1. book moves = oracular source of wisdom,
  2. one set of parameters playing another

- Connectionist learning

  This is revival of the perceptron learning movement that had its heyday in the early 60s. Minsky and Papert's analysis of the limitations of the perceptron put a damper on this research effort and gave way to knowledge intensive learning methods of the early 70's (Winston's ARCH being one of the first examples). The problem that Minsky and Papert had pointed out was the *credit assignment problem.* New research by Rumelhart et. al has resulted in the development of a special class of multi-layered connectionist networks (perceptrons) for which a simple credit assignment rule (called the delta rule) can be devised. Such a network has been used to do a complex speech encoding problem (Sejnowski) and it shows great promise (because of inherent parallelism) in the areas of speech and vision. (signal to symbol transformation problems). Maybe connectionism will be to symbolism (the dominant school of thought in AI and learning), what statistical thermodynamics was to classical thermodynamics.

Chapter 16

Learning II

16.1 Outline of Discussion

I. Important systems

1. Samuel's checker player
2. Foo
3. Version Spaces
4. BACON
5. CLS/IDS
6. INDUCE
7. AQ11
8. Meta-Dendral
9. AM
10. Waterman's poker player
11. Hacker
12. LEX

For each, consider:

a. representation

b. search strategies/learning algorithm/procedure

c. generalizations

d. validation - was it built in?

II. Knowledge Acquisition methods Major issues approaches systems

III. Mitchell's Computers and Thought lecture

16.2 Important learning systems

Samuel's Checkers Player.

Signature tables - abstract spaces for development of appraisal of board position; n-dimensional arrays.

Polynomial evaluation functions.

Rote learning - to improve the look-ahead power; save the results of previous partial gametree searches.

The performance element make moves by conducting a minimax game-tree search.

***FOO* (Mostow):**

Lists.

Advice-taking - advice transformed by applying a variety of "operationalization methods", which reformulate expressions to more specific ones.

No search. The control is driven by human users.

Version Spaces:

Sets - G-set and S-set.

Represented single conjunctive concepts.

Learns from both positive and negative instances

Search - Breath first from both general and specific ends.

BACON:

Number vectors with labels.

Curve-fitter.

Search - generate and test.

CLS/ID3:

Decision trees - with feature at each node.

Search - start each item at the root, trickle it down to a leaf, which is a class (concept).

At each step adds a node to the tree that has the highest discrimination value.
Used information heuristic called "entropy"
ID3: classification of events; used one-shot algorithm
(See reference in "Expert Systems and Micro-electronics")

INDUCE:
"Structural" representation
Nested feature vectors, predicate calculus
Model-driven generate and test

AQ11:
Uses VL1 (extension of propositional calculus)
Learns multiple concepts by an iterative version of version spaces algorithm.
It uses positive instances first – takes as many positive instances as possible without using negatives, then removes a clump of positives, starts over.

Meta-DENDRAL
Ball & stick model of molecules; * for breaks
Rulegen and Rulemod

AM:
Frames – represent heuristics in frames with IF & THEN slots.
Uses best-first search with respect to "interestingness" values.
Concept creation used combinations of concepts
Concept evaluation used "interestingness"
Simple control
Implementation of control structure- agenda

Waterman's Poker Player:
Production rules with feature vectors in IF and THEN slots

Hacker:
Planner-like language.
Generalizes to make plans, generalizes bugs.
Learning by doing.

Validation of learning systems:
Is the learning built in or does the system actually learn?

(This is not a simple dichotomy, as all induction requires bias.)
Was it built-in? AM, Eurisko and Meta-DENDRAL
Has it gained new knowledge?
Has it become faster? Checkers Player

16.3 Mitchell's C & T lecture

Motivations – KA bottleneck in expert systems; it's about time
Problems with version spaces – syntactic, unjustified inductive jumps
Answer – goal-directed learning
Version spaces good for representing partial concepts

Chapter 17

Guest Session on Learning

Guest : Paul Rosenbloom, Stanford

17.1 Main Resources

Three main resources on learning work are the AI Handbook (ch. 14), the Proceedings of the 1983 Machine Learning Workshop, and the machine learning book by Carbonell, Michalski, and Mitchell. Secondary sources are the articles in the learning section of AAAI and IJCAI.

17.2 Dimensions of Learning Programs

The machine learning book characterizes learning programs along three dimensions.

The first dimension is the amount of effort the program must expend to learn. At one end of the spectrum are rote learning programs, which hardly do anything; programs that learn by analogy or from examples are intermediate; and discovery programs do the most work.

Second, programs may be distinguished by the type of structures they use to express the knowledge they acquire. Frames, slots in frames, rules, etc., have been used in

do with the knowledge once they learn it – the actual form of the information is more a second-order effect.

The third dimension is the domain in which learning is performed, e.g. medicine, games, etc. Ultimately we would hope for learning programs that are independent of the domain.

17.3 Other Distinguishing Properties

There are several other properties we might use to distinguish among learning programs:

- Symbolic vs. numeric
  Most recent AI work has concentrated on symbolic representations, although earlier parameter-adjusting research dealt with tuning numeric parameters (e.g. Minsky's Perceptrons, or Selfridge's Pandemonium). A synthesis of these two approaches will be interesting.
- All-at-once vs. incremental
  Recent work is oriented toward incremental learning, as it is harder and arguably more plausible. A batch-mode learner has all the data the incremental learner has (i.e. the latest example), plus all the preceding examples. Batch programs can also deal with noisy data more easily than incremental learners can; one strategy is merely to average over all examples.
- Knowledge acquisition vs. skill refinement
  In knowledge acquisition, the system gets new knowledge from an external source, whereas in skill refinement, the system just makes the use of its existing knowledge more efficient. This distinction may be bogus, because once the system has initial knowledge (which can even be implanted by rote methods), the two techniques can be regarded as essentially the same.
- Deliberate learning vs. learning as a side effect
  A deliberate learner explicitly sets out to acquire knowledge. Another way to learn is to do some sort of problem solving, and to incidentally pick up knowledge along the way (this is the Soar philosophy).

- Common sense vs. scientific knowledge
  This distinction is related to the previous one (deliberate learning vs. side effects), in that you learn common sense knowledge in the course of everyday life, but you usually make a conscious effort to acquire scientific knowledge.

17.4 Major Theoretical Challenges in Learning

Several major challenges in the field of learning were discussed.

- A Better Breakdown of Learning Programs
  First, we need a better breakdown of learning strategies. The three dimensions above and the other properties discussed are rather unsystematic and ad hoc. Part of the reason for the inadequacy of the characterization is that learning research was out of fashion until about five years ago.
- Understanding Generalization
  We also need to get a handle on generalization/concept-learning. The key question appears to be *bias*, i.e. how does the system select a particular generalization out of the infinite number of possibilities?
  One way to make generalizations is to follow a prespecified ISA hierarchy; then to generalize a particular concept, we simply move to the parent concept (e.g. the generalization of "elephant" could be "mammal").
  Alternatively, we could generalize by transformations, such as dropping conditions from a rule, replacing constants in a rule with variables, and so on.
  Another way to generalize is to apply some sort of problem-solving procedure to determine the appropriate generalization. This last method is the one used in LEX (constraints that arise during problem solving are propagated back to the inputs to find the most general condition on them) and in Soar (the conditions included in a rule are exactly those that the problem-solving component needed to look at). Mitchell calls his generalizations "justifiable," because the program has good reasons for deciding on that particular level of generality, just from one example. Soar's generalizations could be called "justifiable" for similar reasons.
- Analogy and Generalization
  A third challenge in learning is understanding the relationship between analogy

and generalization. John Anderson and Russ Greiner have done relevant work here. Greiner does analogy by generalization, i.e. to map one problem onto another, he first identifies a general framework to serve as an intermediary in the mapping. In contrast, Anderson does generalization by analogy – he attempts to map old examples onto the one at hand, and if it works, he is able to abstract a generalization that enabled the mapping. Rosenbloom prefers Anderson's technique, because it is in the spirit of learning as a side effect. The program does not set out to find a generalization, it just derives one as a by-product of applying old examples.

- Applicability of Learning Mechanisms

  Many learning mechanisms account for only one kind of learning, e.g. learning by rote. It would be desirable to find a unified theory that would explain all types of learning; in particular, language acquisition, skill acquisition, and knowledge acquisition.

  The chunking mechanism in Soar (see the Rosenbloom reference on the reading list) is one proposal for a general learning mechanism. So far it covers skill refinement. Current research is extending this to cover knowledge acquisition (Soar is being given the ability to accept knowledge from external sources through some brand of interaction), strategy acquisition (that is, learning control information), and generalization.

- Making Learning Systems Robust

  A learning system should be robust enough that it can be in constant use, improving the performance of the program. Most learning systems to date are too fragile for such heavy duty use; they are only invoked in a controlled environment. There have been successes in relatively narrow domains, however, such as Samuel's checkers program, Berliner's backgammon program, and Lenat's Eurisko.

- The Relationship of Learning to Problem Solving

  Some issues that arise here are how learning and problem solving interact, how closely the two should be coupled, and how the partitioning of their capabilities should be reflected in the structure of the system. A general observation is that if the learner and the problem solver are loosely couples, the resulting system is easier to analyze. A tightly coupled system is harder to analyze but has a more

interesting range of behavior. One good way to investigate these issues appears to be simply to try implementing systems.

- Discovery
  So far we have discussed the use of knowledge to achieve goals; but knowledge can also be used in service of discovery. Not very much has been done in this area. The main projects are Langley's BACON, AM and Eurisko, Meta-Dendral, and MOLGEN. AM and Eurisko are somewhat difficult to classify because they are given all of their domain knowledge at the outset, and so, in a sense, they are not discovering anything that is truly *new*. Anderson's ACT and Langley's AMBER are two other attempts at building discovery programs. One approach to writing a discovery system is to try to model how people make scientific discoveries. When do they try experiments? What sort of blind alleys do they go down, how and when do they detect them? How do they gather data to confirm/disconfirm a hypothesis? How do they interpret data with evolving hypothesis?

- Noisy training instances Samuel's Checkers program deals with noise by altering its weighting functions very gradually. It takes a long time to make any significant changes, but little harm is done if a parameter is tweaked the wrong way because of one piece of noisy data. Data driven concept learning programs are very susceptible to noise.

17.5 AI and Psychology

Cognitive psychology has, as yet, little to say to AI about learning. This is partly due to a disparity between the research methodologies of the two fields. Also the research emphases are different. In psychology, incomplete theories are not tolerated. A theory must account not only for empirical data, but for the observed errors as well. Thus many psychological studies are highly specific and thorough, e.g an investigation of stimulus/response pair learning. In AI, on the other hand, the emphasis is on demonstrating some sort of reasonable behavior.

17.6 Is the answer built in? Or how to validate a learning program

The main difficulty in validating a learning program is in demonstrating that the knowledge that the system acquired was not built in. In some sense, the knowledge is necessarily built in, because it was specified in the form of a learning algorithm that converts certain inputs to the desired knowledge. The question then becomes, how interesting is the path from the inputs to the result: how much of the burden of the learning was on the program. For example, a program that simply memorizes its inputs is not usually regarded as interesting. This brings us back to the first dimension of learning discussed above.

The most convincing programs do not have a lot of hard-wired knowledge about their domains, rather they know how to learn facts in that domain. This is what UNDERSTAND(Simon and Hayes) did.

17.7 Work on Analogy

References on analogy include Carbonell, Winston, Greiner, Gentner, Kling and Burstein. Not too much work has been done on analogy.

17.8 Comparing Two Systems

A good question to think about for the quals is how you would map one problem-solving system onto another (e.g. compare production systems with a blackboard architecture) and the difficulties involved in doing so. This tests not only your knowledge of the systems, but your understanding of their functionality as well.

Chapter 18

Vision I

We answered the list of questions below. We were fortunate in having Jitendra Malik of the Vision Lab at Stanford lead the discussion on these questions. Thank you, Jitendra.

18.1 Why is vision a part of AI?

Class: Some people suggested vague answers: "it does perception", "it does classification". The vision problem is to interpret a scene. Much of the low-level work like edge-detection is very math-intensive; this part isn't AI. For the higher-level stuff like recognition of 3D objects, symbolic reasoning is needed. Brooks' work (ACRONYM) used "AI techniques" effectively. The vision problem is a signal to symbol transformation problem like speech understanding. Binford characterizes the problem as making sense out of perceptual data.

JM: The best way to look at vision is to see it as the inverse of the graphics problem. The graphics problem consists of constructing an image of a scene given a description of the objects in the scene and their properties (e.g texture), the position of the camera and the location and intensity of the light sources. This mapping between the objects and images is many to one. And the vision problem is to invert this mapping. Obviously, to do this, we need to assume additional constraints. An example would **hp an aissiimntion tha,t ttie world is built nn of nierewise continuous snrfa,r,ps. A vision**

system would need to make assumptions and be able to recover gracefully from wrong ones.

Understanding the nature of the above mapping is a key to understanding human vision. This problem comes under the purview of AI because it is ill defined and AI techniques can be brought to bear on it. There is a need to integrate amorphous sources of knowledge in the interpretation of an image. Also we need to do assumption based reasoning to handle contradictions and to recover from overly constraining assumptions.

Additional remarks

Nature is perverse; we need to make generic assumptions in this process. This is what distinguishes blocks world work from the more recent work in vision[Malik 84].

Optical illusions only arise when psychologists are playing games with us by depriving us of many of the vision cues that we use. In nature, there are many more cues; many are due to the properties of light. For example, the famous Necker cube illusion would never arise in nature because if it were a wire frame cube, we would know it by the gleam of the wire.

18.2 What issues does vision share with the rest of AI?

As pointed out above, issues in representation (how do we represent scenes, objects, knowledge needed for the interpretation process), how to organize and use the knowledge, non-monotonic reasoning and dealing with uncertainty and noise arise in vision research also.

Vision seems to be one area where there is a confluence and positive feedback in the works of scientists from various fields – neurobiology, psychophysics, psychology and AI. Hubert and Wiesel won the Nobel prize for proving that there are specific cells in a cat's visual system that detect directional edges.

18.3 Why is Vision hard?

JM: Vision has to deal with a large volume of data. The data can be noisy. Also the image underconstrains the scene. Understanding an image requires a priori knowledge

of the task domain, i.e image understanding systems are 'blind' to objects that cannot be matched to stored representations.

18.4 Chronology of early work

Look at the summary sheet of major systems in vision handed out earlier. The early work in vision was much like work in the rest of AI. Choose a suitable microworld and build programs which operate well within it. Roberts' program (1965) assumed that the objects were polyhedral, with not more than three surfaces meeting at a point, well lighted objects so that the effects of shadows were minimized and a very small object library. The problem was that these methods were too tied to the simple domain and did not generalize to real world situations. Huffman and Clowes brought in the distinction between scene features and image features and the vision problem was treated as that of finding relationships between them. Their junction labelling scheme imposed geometric constraints on the interpretations of line drawings of polyhedrals. Waltz extended the label set by considering the global constraint afforded by shadows cast by a single distant light source. The number of interpretations of each line rose from 4 to 12, but the number of consistent labellings of these reduced dramatically. Mackworth extended Huffman and Clowes work by considering the gradients of the lines in the line drawing (Huffman and Clowes' scheme would label a truncated pyramid and a cube the same way). For more details on this see the Brady article.

Horn's work broke away from the microworld trap – it tried exploiting natural constraints and made more generic assumptions. Horn determined surface direction at each point in an image by exploiting the way the surfaces in the image were shaded. The light source position was assumed to be known and also the nature of the surface (reflectivity characteristics : matte or specular) This problem then reduced to the numerical solution of a differential equation. This work is useful in the industrial context where with structured light sources, the above two assumptions are valid.

18.5 What is Marr's theory of vision?

(see figure at end)
The image is typically a 512 by 512 array of intensity values. The line drawing is called a primal sketch by Marr. It records significant changes in intensity (edges and lines). The assumption used is that significant changes in intensity in the image correspond to significant features in the scene. Lines in the line drawing either

- have geometric significance : tangent plane discontinuities and self-occluding surfaces (e.g sphere).
- correspond to surface reflectance properties in scene : e.g a chalk line on a board.
- are shadow lines
- are texture lines (like the walls of the Law school at Stanford).

Ofcourse, one could think of the texture and reflectance characteristics as having micro-geometric significance.

There is quite a bit of information in the primal sketch that can help in the inversion of the mapping. The blurring of a shadow edge gives partial information on the width of the light source as well as its orientation.

Shape-From-X methods are used to construct surface descriptions from line drawings.Surface Descriptions go by many other names: 2 1/2-D sketch; needle drawing; fundamental sketch. The X above can stand for shape, stereo, motion, color, shading, texture, direct ranging.

The process of inducing object descriptions from surface descriptions is called interpretation/recognition; that of determining surface descriptions given the object descriptions is called prediction. Predictions are necessary to allow hidden surfaces to be hypothesized; they use Object Models.

18.6 Algorithms for vision

Waltz filtering is a special case of the consistent labeling problem. This problem is NP complete. The consistent labeling problem is : given a network of nodes and a set of variables associated with each node, a set of values over which each of these

variables range, and a set of consistency constraints, find assignments to the variables. This problem is well laid out in Bernard Nudel's IJCAI 83 paper. An example of a consistent labeling problem is cryptarithmetic puzzles.

Huffman and Clowes developed a theory of labelling line drawings of trihedral polyhedra. They did not consider illumination effects. Waltz extended the label set proposed by Huffman and Clowes by including edges that result from shadows, and suggested an algorithm for doing edge labeling. The labeling problem is studied in a more systematic fashion in Mackworth's "Consistency in a network of relations " (see Readings for week 1). Why is the Waltz algorithm called a relaxation method? The term is borrowed from the literature in the numerical solution of differential equations. This uses local constraints and updates the information in a node based upon that of its neighbors, in each cycle. Consistent labeling can be thought of as relaxation where only discrete variables are allowed. Global constraints are hard to express in a relaxation framework.

18.7 Main methods for edge detection

An edge occurs at a place where there is a sharp change of intensity. However, step edges are never found in nature (noise corrupts them). There are in general three methods to handle edge detection.

- based on first derivative : find the maxima of first derivative. Need to threshold.
- based on second derivatives : find where the second derivative has a zero crossing. A variant of this first smooths the image and then computes the second derivative (Gaussian) - this is called difference of Gaussian (DOGs). This method gives too many meaningless edges. We can try to compensate for this by using different levels of scale. Marr thought that the human low level vision modules also use DOGs. However, it appears that the limited psychological data that that was based on has since been refuted.
- template matching - for finding directional edges. Very good for industrial vision.
- methods based on surface fitting - these work the best.

Edge detection is really a signal processing problem not an AI problem.

18.8 Methods for connecting edges

The three major methods are Hough transforms (Handbook, page 222), iterative end-point fit (Handbook, page 221) and tracking (Handbook, page 220).

18.9 What are generalized cones?

A generalized cone represents an object as a surface area swept along a curve, in space. For example, a cylinder is a circle swept on a straight line perpendicular to the surface of the circle. It is good to use because it has few parameters; also, it can be used for multiple-level descriptions. These are used in ACRONYM.

18.10 What are some methods for measuring depth?

Direct measurement of depth is done by range-finders using time-of-flight data. Depth is also got by triangulation in stereo views.

18.11 Comparison of processing in human visual cortex and low level vision.

JM: For some computer methods, there is some pretty hard evidence that they correspond to vision in animals. Hubert and Wiesel proved the existence of special cells in the eyes of a cat that detect directional edges. For the Difference of Gaussians (as a method of edge detection), the jury is still out.

18.12 Representations explored in the context of vision

An image is represented as an array of intensity values. A line drawing is represented as a set of splines (together with their topological relationships). A surface can be represented numerically as an array of gradient values or symbolically as a set of parameterized planes (the facet approximation). One can also conceive of having multiple representations for a surface both symbolic and numeric to facilitate different

kinds of computations. Object level descriptions are completely symbolic. The generalized cone is a very good way of representing man made objects because it captures a description with very few parameters, also it can be used for multi-level descriptions. Polyhedral models of the kind used in graphics are too detailed to be of use (not to mention hard to determine by means of the inverse mapping we are doing here).

18.13 Moravec's solution to the stereo problem

JM: He just used 5 or 8 views instead of 2. He then spotted corners and matched them up. Quad trees are used to allow fast lookup. More on this can be found in the answer to the question on shape-from methods.

18.14 What is verification vision?

JM: This is used in industry. You know what the object is, and you have a good idea of the range of locations where the object might be; verify that the object is at place X.

18.15 Applications of vision

These are legion. The introduction to Brady's paper gives a complete inventory. The Chin paper reviews work on automated visual inspection in industry (in particular, the inspection of PCBs, photomasks and ICs).

18.16 What are the various shape-from methods?

Shape-from methods are used in the inferring of surface descriptions from line drawings. For instance, Horn's shape-from shading technique computes the surface normal from variations in intensity in the image. Shape-from stereo requires two images and computes the depth of field by first matching up corresponding points in the image and then using triangulation to compute the depth. Hans Moravec had an interesting solution to the stereo correspondence problem – he used a interest operator for finding

regions of a picture where unambiguous matches are more likely to be found. This interest operator is described in detail on Page 250 of the Handbook chapter. It tends to selects corners first. Shape from texture – the leopard skin example. We view the elliptical spots on the leopard's skin as views of a circle and then compute the slant and tilt of the surface from the major and minor axis of the ellipse (see Nevatia 177-181, if you want to know how it is done).

18.17 The BB architecture for vision

The vision problem requires integrating multiple sources of information and multiple representations (image, primal sketch, surface rep and object rep) and the blackboard framework suggests itself as a natural candidate for the organization of a general vision system. Add to this the fact that we need to cope with the problem of noisy images too, which again speaks for the use of the BB model. The disadvantages of the BB model in this context are that it hides the sequential nature of the interpretation process. It is not clear if we want the feedback loops from object rep. to line drawing (though that is what Shirai's model based line finder did). It seems that the most appropriate place for a BB model is the step between the line drawing and the surface description. Using the Blackboard model arbitrarily leads to unneccesary increases in complexity. Feedback loops are costly, since incorrect assumptions must be retracted. High-level knowledge shouldn't be used to compensate for an inadequate low-level component.

Another issue here is whether we need multiple representations or a single uniform representation. Marr's proposal was to convert everything into surface normals (e.g depth is the first derivative of the surface normal). Barrow and Tenenbaum proposed the use of multiple representations called intrinsic images (an intrinsic image consists of values for some intrinsic characteristic at each image point, together with information about where ther are discontinuities in that characteristic ; examples of intrinsic characteristics are reflectance, surface direction, depth).

18.18 Handling of noise

Filtering & smoothing of data. Model-directed interpretation.

JM's remark : "One man's noise is another man's signal."

18.19 Major successes in vision research.

- Binocular stereo
- Edge detection
- Interpretation of surface contours
- Determination of surface orientation from structure
- Computation of motion
- Representation of 3D objects

[JM] Horn solves *a* problem; but, it is not the right problem. His work goes forwards; that is, he assumes that he knows about the light sources. To go backwards, you need many assumptions about light sources, about the nature of surfaces and reflections, and about noise. Horn's work may be useful for industrial applications.

On Moravec's solution : Normally, stereo deals with two views; the 'solution' simply went to using five views !

18.20 Trends in vision research

- Move from domain dependent work to general principles.
- Representations have been developed that make explicit the information computed by a module.
- The mathematics of vision are becoming more sophisticated.
- Locally parallel architectures have been developed.
- There are growing links between image understanding and theories of human vision.

[JM:] The current trend is to follow Marr's general structure for a vision system.

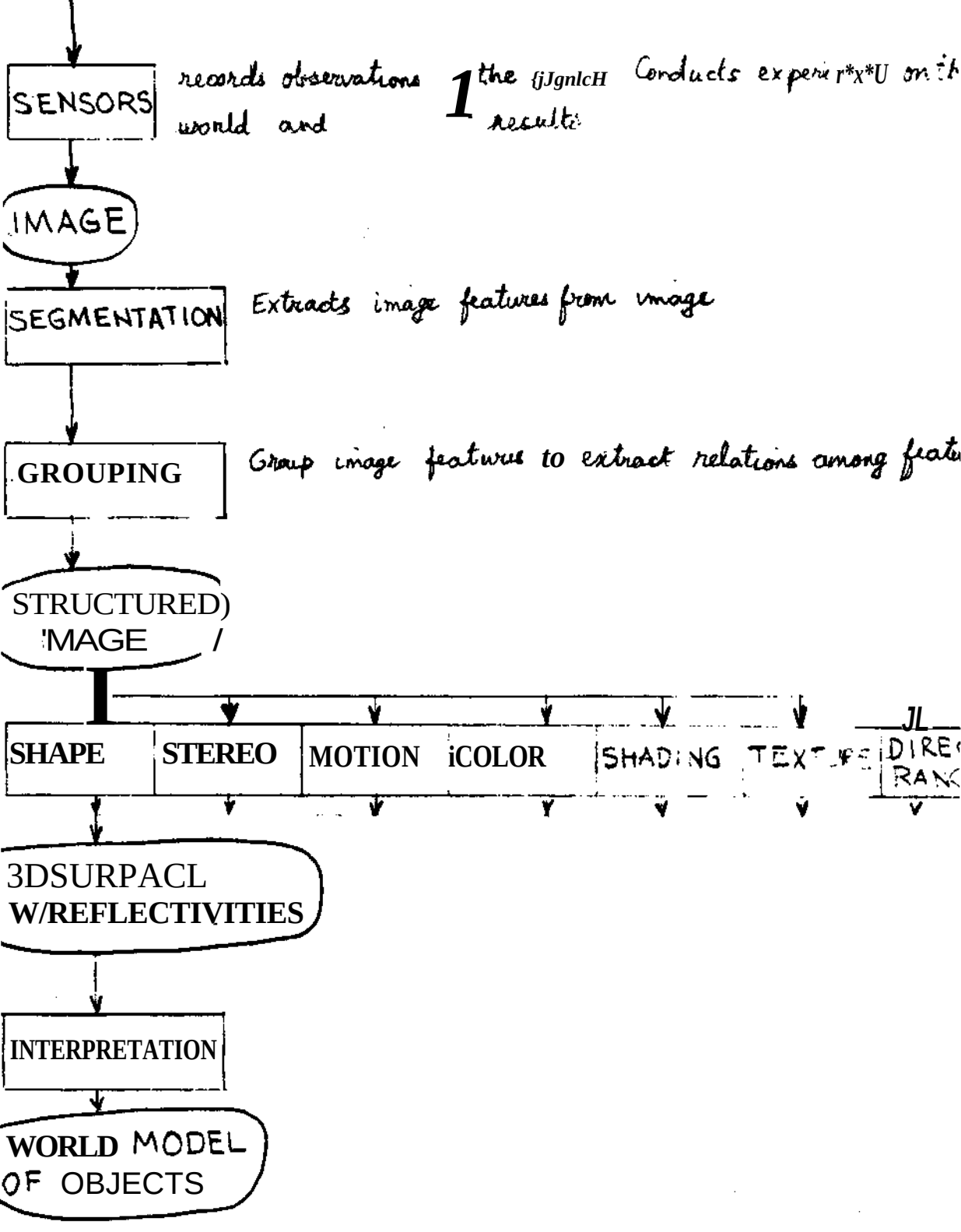


FIGURE Z: PtRC£?T\ON

(from Tom Binford's class notes)

V I S I O N

This is organized like the handbook.It gives summaries of the most important vision systems and modules.

1. Blocks world programs
   a. Roberts
   b. Guzman
   c. Falk
   d. Huffman and Clowes
   e. Shirai
   f. Mackworth
   g. Kanade
2. Edge and line finding programs
   a. Differences method
   b. Edge masks
3. Integrated vision systems
   a. VISIONS
   b. Multiband aerial photo interpretation system
   c. Query oriented vision system
   d. ACRONYM
4. Applications
   a. Transistor wire bonding system
   b. INSIGHT
   c. Mobil vision module

A short description of each of the systems above follows

Section 1 --- BLOCKS WORLD PROGRAMS

* Roberts - MIT - 1965

1. Pioneering work in vision -- first program to understand a polyhedral scene consisting of polyhedral objects in arbitrary spatial config.
2. Bottom up approach -- preprocessing,edge detection,line drawing constr, modeling objects and matching

* Guzman - MIT - 1968

1. Developed a program (SEE) which analyzed scenes of known object without prestored models of objects.Started with a line drawing and identified all separate objects in it,even partially occluded ones.Only did segmentation,no 3D description provided.
2. Used local constraints on vertices and junctions to derive segmentation into regions.Used heuristics to do this.
3. Problems -- too adhoc,did not consider the geometrical and physical constraints inherent in the problem.(something which Huffman and Clowes did)

* Huffman and Clowes - MIT (?) - 1971

1. The first systematic approach to polyhedral scene analysis.
2. Clean distinction between image and scene features.
3. Identified the goal of picture interpretation to be to interpret elements in the image domain as properties of the scene domain.
4. Gave a clear theory of labelling line drawings which contained all of Guzman's work on segmentation.
5. Set a trend in vision research which was to account later for some spectacular successes.

* Falk - Stanford - 1972

1. Falk's INTERPRET worked on the problem of identifying the visible

egmentation method was an improvement over Guzman's.

- MIT - 1972
---
ded Huffman Clowes labelling to include cracks,shadows and
ably concave edges.
oped a clever labelling algorithm called the Waltz filtering
ithm.Could label complex polyhedral scenes in near linear time.
tant result from this work was that inclusion of more detailed
mation constrains and facilitates (rather than complicate) the
pretation process.

- MIT - 1973
---
research was concerned with the problem of finding lines
tly from the intensity image.
edge about polyhedral scenes was used to guide the line
tion process.
program simultaneously generated and interpreted the line
ng using the partially developed interpretation as top
info to search for lines.
mmary,a semantic approach in which knowledge of the task
n was used to direct low level vision processes.

rth - MIT - 1973
---
orth's POLY used the gradients of planes to constrain
pretations of lines in an image.Was an improvement on
an and Clowes qualitative labelling scheme.
research contribution was a new representation -- the
ent space and its use in interpreting polyhedral objects.

- Stanford or CMU (??) - 1979
---
work sheds light on the issue of multiple interpretations
uantitative shape recovery of natural interpretations in the
mi world.
regularity heuristics -- parallel line and skewed line heuristic
dition to constraints imposed by Waltz labeling and surface
tations.
egularity heuristics were used to filter out unnatural
pretations. cf. three interpretations of a cube.

2 --- EDGE AND LINE FINDERS

difference operators

ts Cross (1965)
operator (?)
feld and Thurston (1971) -- variable window size of edge operators.

difference operators

and Hildreth (1980) -- Laplacian

n matching (edge masks)

el operator (1971)
ia-Babu operator (1979)
rd-Horn operator (used in Acronym)

inding methods

ing (done by Shirai line finder)
ative end point fit (Duda and Hart) 1973
transform

3 -- INTEGRATED VISION SYSTEMS

* Interpreting multiband aerial photographs (Japan,1978)

1. Interprets a class of multiband aerial photographs well.
2. Uses the blackboard model with independent KS's with a hierarchical structure.

* VISIONS (Hanson and Riseman,1978,UMass)

1. Is directly patterned after the HEARSAY -II system.
2. Details seem irrelevant !

* Query oriented vision system (Ballard,Brown,Feldman,1978)

1. Abandons the idea of exhaustive processing at the lower levels and just processes enough to answer a query.
2. All processing done in the 2d image domain,no 3d models are used.
3. Three layered information structure -- image data structure, a model layer and a sketchmap.

Section 4 -- APPLICATIONS

* A transistor wire bonding system (Japan,1976,for Hitachi)

1. One of the first production robotic vision systems.
2. It visually locates a transistor chip and automatically bonds gold wires between the electrodes on the chip and the outer leads.
3. Template matching used.
4. Can assemble 2000 chips/hour with an accuracy of 99% which is twice the speed of semi automatic bonding machines.

* CONSIGHT (Holland,Rossol and Ward,1979,for GM)

1. This is a vision based robot system that picks up parts that have been randomly placed on a moving conveyor belt.
2. Uses structured light to overcome the difficulties of working in a noisy environment where contrasts are low.
3. Also uses run length coding for computational efficiency in edge and connectivity analysis.
4. Numerical shape descriptors calculated for recognition.

* SRI Vision module (Agin and Geason,1979)

1. This is a package of useful programs with all the necessary hardware for many visual sensing and inspection tasks.
2. Modules include run length coding for efficient connectivity analysis, numerical shape descriptors and recognition of parts with a nearest neighbor method.

* ACRONYM (Brooks et al)

1. A domain independent.model driven interpretation system.
2. A user describes the objects expected in an application domain,along with possible relationships and the system interprets images as specialization of the domain.Extracts 3D info like shape,structure,location and orientation too.
3. Model based vision -- modeling,prediction,description and interpretation.
   Modeling done using the gen cylinders rep and using algebraic constraints.
   Prediction involves geometric reasoning and results in a prediction graph.
   Description works from the image using the Nevatia-Babu line finder and forms the picture graph.
   Interpretation consists of finding subgraph isomorphism between prediction and picture ~~graph~~.

Chapter 19

Guest session on Vision

Our guest was to be Prof. Tom Binford of the Stanford AI lab. He could not attend the session due to a scheduling conflict. The questions prepared by the author for this session are listed below.

Vision

- Organization of vision systems – what is the theme in the research on SUCCESSOR?
- Methods of segmentation – figure/ground distinctions.
- Low level vision seems to be closely tied into neurobiology and psychophysics. What is the nature of the information feedback between these disciplines?
- Traditionally vision has stood on the periphery of AI. But you claim that perception is the source of representations in AI. What is the intuition behind this claim?
- What are the current trends in vision research? i.e where is vision research heading?
- What lessons have we learnt from previous work? Historical review or a rational reconstruction of work in vision.
- How can results in vision be fed back into general AI?
- How much will the development of massively parallel architectures help vision?
- Connections between learning and vision.

Robotics

- What part of robotics comes within the purview of AI? What are the current issues being pursued?
- Integrated robotics and vision efforts.
- Open problems in robotics.
- Main centers for robotics research.

Robotics at Stanford

This was a lecture given by Prof. Tom Binford, Head of the Vision and Robotics lab at Stanford on 11 October 1984. This is a transcription of the notes I took at that lecture. It details the current efforts in robotics at Stanford, touches upon the research issues in the field and also mentions work going on in other centers of robotics research.

- **Problems in Robotics**
  There are three main problems being attacked at the robotics lab at Stanford.
  - Sensing and perception
  - Planning and decision making
  - Execution and action

  Perception consists of making sense out of sensing, it is the signal to symbol transformation problem. Planning is for acting upon the real world. The model of a robot being used is :

  This is quite similar to Stan Rosenschein's BDI model.

- **Sources of information**
  There are four courses offered at Stanford which cover material in robotics and vision. They are

- CS227A - offered in Autumn every year - covers manipulators : kinematics, dynamics and control
- CS227B - offered in Winter every year - covers vision
- CS227C - offered in Spring every year - follow up to the above two courses, where typically a substantial project is undertaken by each student.
- CS327 - offered every quarter - Robotics seminar

The following books give information on research in vision and robotics.

- Robot Manipulators : Paul (MIT Press)
- Machine perception : Nevatia (Prentice-Hall)
- Vision : Marr (Freeman)
- Computer Vision : Ballard and Brown

The Nevatia and Ballard & Brown texts are introductory texts in computer vision. The Robot Manipulators text is a compendium of recent papers in Robotics.

- **Structure at Stanford**

  - SIMA : Stanford Institute for Manufacturing
  - CAMS : Center for Automation of Manufacturing Sciences
  - CDR : Center for Design of Research
  - CTRIMS : Center for Teaching Research in manufacturing Science
  - CFMF : Center for Materials Forming

  The groups that should interest AI'ers are CAMS which does research on vision and robotics and CDR which deals with human interface aspects.

- **Who's who in Vision and Robotics**

  - Stanford :
    The vision lab was set up in 1965 and pioneering work has gone on since. Languages for programming robots have been developed (AL). General purpose vision systems have been built (ACRONYM

and SUCCESOR). Work on stereo vision, geometrical representations and reasoning, perceptual organizations and learning of physical descriptions from functionality sp ecifications are also being pursued.

- MIT :
  Another pioneering center for research in vision and robotics. There has been a strong emphasis on the close relationship between vision, psychology, neurobiology and psychophysics. Research in this vein has been conducted under the leadership of David Marr and this tradition continues under Poggio, Hollerbach and Lozano-Perez.

- CMU :
  Five years ago the Robotics Institute was set up at CMU under the direction of Raj Reddy. It is an inter-disciplinary venture. Research in manufacturing, job shop scheduling arc pursued there. The emphasis is on applications oriented research.

- SRI:
  Research on vision and manipulation (verification vision), industrial automation, largely application oriented image understanding project.

- Other Universities :
  There are quite a few universities in this country which offer programs that range from application oriented to fundamental issue-driven research. Notable among these are the Universities of Rochester, Maryland and Amhcrst.

- Industry :
  Japmiese companies like Panasonic and Hitachi are doing apllications directed research in computer vision. U.S companies - Unimatiou, PUMA and IBM do some vision and primarily robotics research. GM uses simple computer vision methods for car door assemblies. Most of these* applications arc oriented toward manufacluring.

• **Current issues in Robotics Research**

- Real time obstacle avoidance
- Mobile obstacle avoidance
- Link collision avoidance

- **Current issues in vision**
  The underlying theme is that perception supplies representations for intelligence. Research questions being investigated are

  - Inferring 3D objects from 2D images.
    In particular, interpreting line drawings [Malik 85].
  - Perceptual organization.
    Issues like texture perception, brightness and colour constancy are studied here. See David Lowe's recent thesis.
  - Classical AI questions in representation and reasoning
    Representation of objects (see Scott's paper on quasi-invariants) and assumption based reasoning in the interpretation of line drawings (Malik 84). Vision is a rich environment for pursuing research in learning.

- **Issues in spatial reasoning**
  This has been explored in the context of SUCCESSOR.

  - Knowledge and data at various levels of detail.
  - Translation between the various levels of knowledge and data into geometric constraints. Formalization of geometrical manipulations.
  - Interpretation ate multiple levels. Involves propagation of constraints across different levels.

  This problem is similar to the 3D interpretation problem that PROTEAN is trying to solve, note that the same issues crop up there too. PROTEAN combines knowledge at different levels by using the blackboard framework.

Chapter 20

Natural Language I

This set of notes has been substantially extended by Mary Holstege for the benefit of those for whom NL is not a strong area. Thank you very much, Mary.

20.1 Sources of information

Terry Winograd's book Language as a Cognitive Process, Volume I : Syntax is a very good source both for explanations of various grammar formalisms and for a brief history and outline of some systems of importance. The Handbook is (at best) an indifferent source for such things. It is also rather out of date. The book Computer Models of Thought and Language, edited by Schank and Colby has some good descriptions of some systems as well as some other papers of more general interest.

20.2 Outline of topics covered in discussion

I. Overview of natural language
   A) History
   B) Approaches

II. Machine translation

III. Grammars

A) Formal languages
   i. Chomsky hierarchy

B) Transformation grammar
   i. Chomsky and the REST
C) Systemic grammar
D) Case grammar
E) Lexical functional grammar (LFG)

IV. Parsing
A) Issues and strategies
B) ATNs
C) GSP (chart parsing)
D) Stack and buffer parsers

V. Text generation

VI. Examples
A) Early syntax-oriented systems
B) Wilks' machine translation system
C) LUNAR
D) SHRDLU
E) MARGIE/SAM/PAM
F) LIFER
H) DIALOGIC
I) TEAM

Editorial remark: the Handbook's use of the term 'formal grammars' is both non-standard and misleading, since all the grammars described in the Handbook and indeed any grammar called such by a linguist is by definition 'formal'.

20.3 Overview of NL

In the 1940's the production of concordances and word indices was one of the first application of computers to natural language. This use of computers to aid linguists (aka *Computational Linguistics*), continues to this day.

In 1949 Weaver suggested the use of an *interlingua* for machine translation. Machine translation was regarded as a code-breaking problem in which a word-by-word substitution was followed by some syntactic post-pass to rearrange the words and add inflections. While it was recognized that to get high quality translations human pre- and post-editors may be required, translation was conceived of as a large but relatively straightforward problem.

Oettinger's system (circa 1955) exemplifies this approach: each word in a Russian text was rendered as a set of possible English translations. An example of the output is given on page 235 of the Handbook.

It should be pointed out that in these early days the grammars available were relatively primitive and technical problems (such as storage space) predominated.

In 1957 Chomsky came out with his classic work *Syntactic Structures*, in which he presented transformation grammar (see grammars, below). The so-called standard theory was put forward in his later book *Aspects of the Theory of Syntax* (1965).

In 1960 Bar-Hillel pointed out that a purely syntactic approach to machine translation was doomed to failure, citing such examples as "The pen is in the box" and "The box is in the pen" in which the correct choices of the senses of pen and box depends on knowledge of the sizes and shapes of things in the world.

In 1966 the ALPAC report reiterated this view in an official context and funding was cut.

In the early 1970's ARPA funded a five-year program in speech understanding.

Recent work is characterized by (a) the hiring of linguists to develop well-motivated grammars for use by natural language computer systems and (b) the use of a great deal of domain knowledge to alleviate ambiguity difficulties.

20.4 Machine translation

The history of machine translation has been sketched above. The basic lesson from this failure is that semantics (in the ES jargon 'domain knowledge') is necessary for doing machine translation. [Editorial Remark: Unfortunately this means it is also necessary for doing any sort of interesting work in natural language which means that a natural language system either needs to work in a highly restricted domain (cf SHRDLU et al) or needs to have knowledge about a domain which includes all common knowledge about the world. Clearly natural language provides a strong counter-example to the notion that throwing facts at a problem makes it go away as there are far too many 'facts' required for this to be a sufficient answer]

Recently interest in machine translation has resurfaced, using more sophisticated grammars and more knowledge of semantics. There is work going on at Austin, Texas on translation of technical manuals.

20.5 Grammars

- **Formal languages**

```
The Chomsky hierarchy:
   Level 0:
      recursively enumerable languages
      Turing machine languages
      grammar rules of the form: anything --> anything
   Level 1:
      context sensitive languages
      linear bounded automata languages
      grammar rules of form:
        sequence --> longer or equal sequence
   Level 2:
      context free languages
      languages of finite state automata with a stack (or RTNs)
      grammar rules of form: non-terminal --> anything
   Level 3:
```

```
regular languages
languages of finite state automata
grammar rules of form:
   non-terminal –> non-terminal terminals
                      OR
          non-terminal –> terminals
```

Where does English fall in this classification?

English is context free except for such constructions as the 'respectively construction', e.g. "John, Martha, and Bill ate lobster, salmon, and squid, respectively." Notice that we only need context sensitive rules if we want the parse tree to come out 'right' with 'John' as a sibling of 'lobster', 'Martha' a sibling of 'salmon', and 'Bill' a sibling of 'squid'. Linguists have argued that it is unclear that the tree should be of this form anyway.

Similar constructions exist in other languages but they have the properties of being (a) rare (b) incomprehensible beyond a stacking level of three or so and (c) almost always subject to the argument that context sensitivity is only required to get the tree to come out 'right' where what is 'right' is debatable.

Note that number and person agreement (for example) is NOT a context sensitive construction since it is possible to construct context free rules to handle them. HOWEVER, these rule sets are extremely redundant (in linguistic parlance *unrevealing*) so linguists use more powerful grammars than they really need in a formal sense.

- **Transformation grammar**

This too is Chomsky's fault. The current mainstream view of the world in linguistics is called the revised extended standard theory (REST). The basic idea is that a base set of CF rules generate the *deep structure* of a sentence, then an ordered set of transformational rules apply recursively (from innermost sentences to outermost sentences) to generate the *surface structure* to which other sorts of rules (such as morphological and phonological rules) apply to generate the actual utterance. (Note : this gets turned around if you are looking at things in terms of comprehension rather than generation.)

NOTE: Clearly this is in no way a description of what goes on in a person's head when they speak English. For one thing, in generation, the choice of lexical items

(words) happens after the structure of the sentence has been chosen (various extensions and revisions actually help repair this particular difficulty, but there are others). So what is it? Here is where (and why) Chomsky does a fast shuffle and invents the competence / performance distinction (see below).

Advantages (over phrase structure grammars i.e. CF grammars) "same meaning same form" "similar meaning similar form"

Disadvantages Horrendous combinatorial explosion if you try to apply them directly (Some attempt to do this using analysis by synthesis – Riesbeck)

Criticism that syntax and semantics are too isolated; that you need some semantics to be able to parse efficiently/with human preferences. That is, almost any sentence has many possible parsings; in Chomsky's model, the syntactic component hands a set of parsing to the semantic component to sort out, but people only seem to be aware of one parsing. (There are also various timing studies to determine whether the existence of other possible parsings slows down one's recognition of the preferred one.) Example: "The man saw the boy with a balloon."

Competence

Competence is "what the native speaker knows about his language" It is a model of what a speaker would believe about his language if he didn't stumble over his words, change his mind halfway through a sentence, didn't impose arbitrary limits on the depth of nesting he could parse, didn't confuse semantic anomaly with syntactic bogosity, didn't make acceptable syntactic structure so dependent on choice of lexical items (i.e. didn't use fixed expressions), etc., etc., etc. In short, if he behaved like a well programmed machine with infinite processing resources. For example, the knowledge that the following sentence is good English is part of your competence as a native speaker of English, allegedly: "The mouse the cat the dog the man saw chased bit died."

Performance is what the native speaker actually believes, says, understands, and so on. A complete model of performance would explain errors and limitations.

Note that it is unclear where the boundary between these two lies and some have argued that it is a bogus distinction anyway. For example, are metaphor and discourse structure properly handled in a competence or a performance model?

Adequacy of various kinds (Chomsky)

Empirical adequacy: explain the data

Descriptive adequacy: explain the data, but also explain how the data were generated

Explanatory adequacy: explain all that and how the processes that generate the data got to be that way (e.g. how learned?)

- **Systemic grammar**

  In a systemic grammar the sentence (or whatever) is represented as a set of choices from a system of choices. Some choices are made independently, some are forced by making other choices, and some are made available only when other options have been picked.

  Example: see pp. 304-306 in Winograd. See Example 1 at the end.

- **Case grammar**

  The fundamental idea in case grammars is that in addition to some syntactic relationship between, say, noun phrase and verb, there is a semantic relationship or *case role*. Some case roles are Agent, Affected, Instrument, Location, ... Each verb has a set of *case frames* which specify the set of case roles that can co-occur with that verb in any particular sentence. Certain rules about the relationship between case and surface syntax are added.

  ```
  Example:
   In all sentences:
   Agent = John; Affected = the window; Instrument = the rock

          "John broke the window with the rock"
          "The window was broken by John with the rock"
          "The rock broke the window"
          "The window broke"
         *"John and the rock broke the window"
              (Agent and Instrument may not be conjoined)
  ```

- **Lexical functional grammar (LFG)**

  The main people here are Bresnan and Kaplan. A good description is in Winograd starting on page 334. The main idea is to have a structure with some semantic information as well as a standard parse tree. The tree is called the

c-structure and the other representation is called the f-structure. LFG has the advantage of being relatively straightforward to implement (using unification). Example: see pg. 335 in Winograd. See Example 2 at end.

20.6 Parsing

- Issues and strategies These are pretty much what you expect:
  * precision vs flexibility tradeoff
  * parallel examination of choices or backtracking
  * integrate 'knowledge sources' (semantics,syntax,morphology)
  * if so, in what way
- ATNs

  What one is:

  Consider a FSA. Now consider a collection of networks, each with a name and allow arcs to have the name of another network, rather than just a terminal as was the case in the FSAs. Such a beast is an RTN (recursive transition net). Augment this structure by having registers which can be set and examined and by having conditions and actions associated with each arc. Such a beast is an ATN (augmented transition net).

  Example: see Chapter 5 of Winograd. See Example 3 at end.

  Advantages and disadvantages
  * Ghastly to try to understand and write correctly
  * So powerful that they are not really a theory of language hence restrictions need to be made
  * Can represent transformational grammar rules as ATNs to get a more efficient way of parsing with a transformational grammar
  * They are easier to deal with than some other things
- GSP (chart parsing)

  The basic idea here is to keep all a record of all possibilities considered in the parse so that work does not need to be redone. A chart parser has a chart which has links representing possibilities to examine and an agenda of things

to do. Arcs are marked as active or inactive and have some other information associated with them. When the entire sentence is covered by a single completed 'S' arc the parse is finished.

Example: see pg. 118 of Winograd. See Example 4 at end.

Note the similarity between this idea and Waltz labelling; propagating constraints and keeping track of multiple possibilities.

- **Stack and buffer parsers**

  Main example: the Parsifal system. A stack and buffer parser has a "window of opportunity" of limited size. When it examines a word at the head of the window, it may either deal with it at once, or move it to one side for later. There is no backtracking in this scheme, and this is a major advantage. There is also some psychological plausibility to this approach. For example, the parser gets confused on the 'garden path' type of sentence that also confuses people, (e.g. "The horse raced past the barn fell.")

20•7 Text generation

There is more to this than the weird and amusing stories of RACCTER and TALE-SPIN: text generation is needed for translation, for systems which are supposed to give natural responses, and for paraphrase systems.

Early systems were either random (for amusement purposes) or using fixed patterns (e.g. ELIZA). More successful things have been done using more sophisticated grammars and more semantic information. Recently, Doug Appelt (anyone else?) has been looking at text generation as a planning problem in which one is trying to satisfy multiple goals.

Problems: coherence of the discourse : handling anaphora and focus

level of information given : avoiding saying too much or too little

2(L8 NL Systems

The following page has a short summary of important NL systems compiled by John Lamping. For TEAM see the required paper on the reading list for NL.

f"structure (solution*** -fm «^M«tions)

$X_1 = X_3 =$ [Subject = X_2
Object = X_4
Object-2 = X_5
Tense = Past
Predicate = 'hand <girl, baby, toy>']

$X_2 =$ [Definiteness = Indefinite
Number = Singular
Predicate = 'girl]

$X_4 =$ [Definiteness = Definite
Number = Singular
Predicate = 'baby]

$X_5 =$ [Definiteness = Definite
Number = Plural
Predicate = 'toy']

P → VP

(↑ Subject) = ↓ ; ↑ = ↓

Det Noun

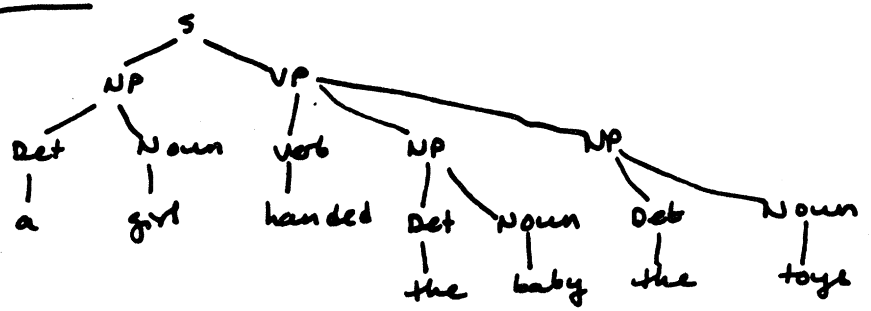
Verb (NP (↑ Object) = ↓ (NP (↑ Object-2) = ↓))

terminer (↑ Definiteness) = Indefinite
(↑ Number) = Singular

Noun (↑ Number) = Singular
(↑ Predicate) = 'girl'

Verb (↑ Tense) = Past
(↑ Predicate) = 'hand < (↑ Subject), (↑ Object), (↑ Object-2) >'

ture:



ns: (Instantiations)

$(x_1$ Subject) $= x_2$, $x_1 = x_3$

$(x_3$ Object) $= x_4$, $(x_3$ Object-2) $= x_5$

$(x_2$ Definiteness) = Indefinite, $(x_2$ Number) = Singular

$(x_2$ Number) = Singular, $(x_2$ Predicate) = 'girl'

$(x_3$ Tense) = Past

$(x_3$ Predicate) = 'hand < $(x_3$ Subject), $(x_3$ Object), $(x_3$ Object-2) >'

$(x_4$ Definiteness) = Definite

$(x_4$ Number) = Singular, $(x_4$ Predicate) = 'baby'

$(x_5$ Definiteness) = Definite

$(x_5$ Number) = Plural, $(x_5$ Predicate) = 'toy'

ple #3: ATM (a simple one)

Conditions & Actions:

S-1: $_{a}NP_{b}$
Action: Set Subject to * { * = thing just parsed }

S-2: $_{b}Verb_{a}$
Action: Set Main-Verb to *

S-3: $_{a}Verb_{a}$
Condition: Type of Main-Verb = Be, Do, Have, or Modal
Action: Append Main-Verb to Auxiliaries. Set Main-Verb to *

S-4: $_{c}Verb_{d}$
Condition: Form of * = Past-Participle & Type of Main-Verb =
Action: Set Voice to Passive. Append Main-Verb to Auxiliaries
Set Main-Verb to *. Set Direct-Object to Subject. Set S
to a dummy NP.

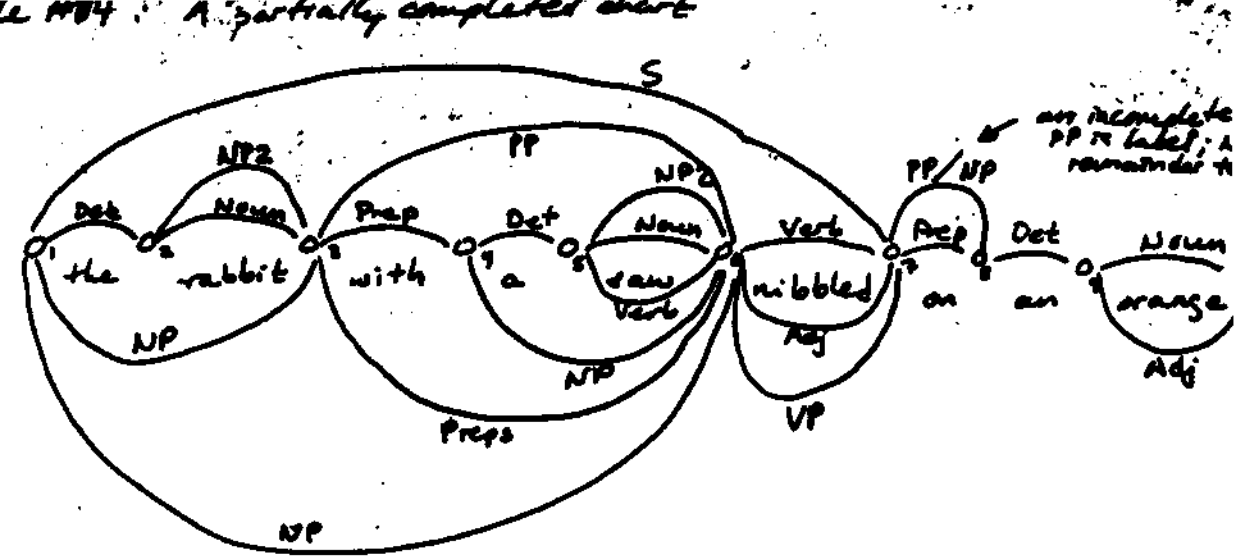
t-S: $_{c}NP_{d}$
Action: Set Direct-Object to *

S-6: $_{d}PP_{d}$
Action: Append * to Modifiers

S-7: $_{d}PP_{d}$
Condition: Voice = Passive & Subject = dummy NP and the word
Prep of * = 'V
Action: Set Subject to the Prep-Object of *.

S-8: SEND

To do number agreement we can compare Number of NP with
Number of Main-Verb. We can access the registers of the calling
with the notation ↑ }


S
NP2
PP
NP2
PP/NP
Det
Noun
Prep
Det
Noun
Verb
Prep
Det
Noun
the
rabbit
with
a
raw
nibbled
on
an
orange
Verb
Adj
Adj
NP
NP
PrepP
VP
NP

Natural Language Understanding — [illegible]

Wilks' System, 1973, Wilks
What: Translation from English to French
How: Decomposition into semantic primitives
Overview: Word meanings are expressed in terms of formulas, which express preferences for how the word is used in sentences. These preferences can be used to disambiguate. Then the disambiguated information can be used to generate the target language. There is no real "understanding" of what the program is processing.

LUNAR, 1972, Woods, BBN
What: Access to a Geology database
How: ATN parser
Overview: An ATN parser converts the NL into a query language which is a generalization of predicate calculus. Tha is run against the database. The system achieves goo performance through tuning to its narrow domain

SHRDLU, 1972, Winograd, MIT
What: Access to a blocks world
How: Procedural representation with multiple knowledge source
Overview: The parser is defined procedurally, modeled on a systemic grammar. It interacts with domain knowledge The processed sentence is fed to a PLANNER based theo prover that knows about the blocks world.

MARGIE, 1975, Schank, Stanford
What: Inference and paraphrasing of sentences
How: Translation to semantic primitives
Overview: A sort of demon driven parser builds a conceptual dependency graph of the sentence. This captures the meaning of the sentence in terms of semantic primitiv An inferencer then makes conclusions and assumptions based on the sentence. Finally, a generator paraphrases the input, or expresses the inferences reached.

SAM, 1975, Schank, Yale
What: Story understanding
How: Filling in scripts
Overview: A parser produces a conceptual dependency structure from a paragraph. This is then fit into one or more of several scripts. The scripts allow understanding of details that were left out of the paragraph.

PAM, 1977, Wilensky, Yale
What: Story understanding
How: Explaining a paragraph in terms of goals
Overview: A parser produces a concepptual dependency structur from a paragraph. The system tries to identify state goals of actors in the paragraph, or plans they are u

LIFER, 1977, Hendrix, SRI

What: General purpose database access

How: Context free grammar with large amounts of semantic informat
in the grammar

Overview: The system is a general architecture for building
Database interfaces. It takes a context free grammar
and builds a database query language output. The
grammatical catagories typically reflect the semantics
of the domain.

LADDER, 1977, Sacerdoti & Hendrix, SRI

What: Access to distributed database

How: Built on top of LIFER

Overview: There is a domain dependent grammar for LIFER,
The query that results is then analysed to determine
which databases must be accessed to return the answer.

DIALOGIC, 1982, Grosz, et al, SRI

What: Domain independent NL parsing

How: Augmented phrase structure grammar

Overview: The system contains a phrase structure grammar,
augmented with features, for a large fragment of
English. A parser builds a logical form that reflects
the syntactic structure of the input sentence. A specia
routine handles quantifier scoping, and another pragmat
routine handles things like prepositions and noun-noun
compounds. This system is the basis for TEAM and KLAUS

Chapter 21

Natural Language Understanding II

21.1 Outline of discussion

The blackboard outline for the session is included as a useful reference, although many of the topics listed could not covered in the discussion:

1. Continuation of discussion on various systems.
   For each system:
   a) who/when/where/why
   b) main ideas in system
   c) representations used
   d) parsing strategies and how was semantics handled?
   e) major limitations
   f) successor systems
2. John's summary of
   a) Winograd's paper
   b) compositional semantics
3. Barwise and Perry's "Situation Semantics"
4. Language Generation – issues

5- Views of NL
   a) Terry Winograd
   b) Barbara Grosz
   c) Roger Schank
   d) CSLI
   e) SRI
   f)MIT
6. Modern day NL systems (TEAM)
7. Connections between NLU and the rest of AI
8. Discourse
9. Open problems in NLU
10. Questions on Speech Understanding:
   a) Speech recognition vs speech understanding
   b) Similarities/differences between speech recognition and vision
   c) Isolated word recognition system - capabilities, problems with template matching approach, other approaches?
   d) Continuous speech recognition - what makes it so hard? What approaches have been used?
   e) ARPA SUR project - when/where/participants/projects/initial goals/ lessons and triumphs (both for CSR and the rest of AI; hint: BB architecture).
   f) Main ideas in HEARSAY, HARPY, HWIM, SRI-SDC; comparisons.

21.2 Discussion on NL systems

Early systems (BASEBALL, STUDENT, etc.):
- Tended to focus more on pattern matching.
- Main approach: use a limited domain, focus on key words and use appropriate pattern matching to detect "understandable" sentences.

Winograd's three categories for natural language systems:
- Systems based on pure logic (example: QA3)
- Systems based on ad hoc methods ("hackery") (example: SIR?)

- Systems using non-logical methods (semantic nets, building data structures) (example: Quillian's work)

Under this categorization, early systems were generally ad hoc.

Useful reference: Phrasal Lexicon (author??)

Wilke's Translation System (1973)
- Interlingua used basic set of semantic ideas as primitives
- primitive elements used to build formula for each word meaning
- formula represents association preferences of a word
- Used preferences in matching - select match with best preference.
- Preferences similar to case frames.

For reference to Wilke's system see Shank and Colby "Computer Models of Thought."

LUNAR (Woods 1972)
- First real world domain (moon rock and soil composition)
- Done by BBN
- Used ATN grammar
- Translated natural language questions into Data Base query language, applied request to data base system for information retrieval
- Quantifiers were handled extremely well
- Query language a generalization of predicate calculus

SHRDLU (Winograd 1971 MIT)
- a breakthrough, because it integrated a variety of domains: natural language, question response, command acceptance, simulation of a simple block world, etc.
- used procedural representations for syntactic, semantic, and reasoning knowledge
- could remember context of discussion to disambiguate queries.
- "directed backtracking" used to recover from problems in parsing, using specialized routines.
- limitations:
- ad hoc design aspects, e.g. representation of speaker/hearer internal structure, resulting in limited extensibility
- procedural representation of words has inadequacies
- constrained to block world domain.

- the representation and reasoning operations could not be expanded to handle common sense knowledge.

*Digression...it was noted that the only general domains which have been attempted have been done using some type of semantic primitives as a basis.

MARGIE (Shank et al. 1973)
- Used to make inferences / paraphrase a text.
- First component converted text to conceptual dependency representation.
- Used demons that examined text to generate the CD representation.
- Middle component ("inferencer") deduced large number of facts from a given proposition, in the current context of the system's memory
- Inference knowledge represented in memory in a modified semantic net
- Third component generated text, using a discrimination net to distinguish between different word senses an an ATN to linearize the CD representation

SAM/PAM (Shank et al. 1975)
- Story understanders, using scripts, plans and CD representations.
- Can produce story summaries, answer questions.
- Scripts: a sequence of expectations for a series of events that describes some stereotypical human activity, e.g. going to a restaurant.
- Evoked by matching a key word/phrase to the script header.
- Scripts could be interrupted, then dropped or resumed according to the flow of the text.
- SAM fits stories into one or more of these scripts.
- Plans: a more general method. Two types: named plans and themes. A named plan is a set of actions and subgoals for accomplishing a main goal. Themes, e.g. LOVE, imply particular goals of the actors.
- PAM determines the goals that are to be achieved in the story and attempts to match story actions with goal-achieving methods.

LIFER (Hendrix, SRI, 1977)
- Designed to be an off-the shelf utility for building "natural language front ends" for applications in any domain.
- An interface builder uses language specification functions to define an application language (subset of English).

- This used to interpret NL inputs as commands for the application system.
- Language specified in terms of grammatical rewrite rules.

Current Systems (TEAM)

- NL system builders now: 1) learn linguistics; 2) specify a grammar with an abstract formal structure, e.g. Montague or LFG.
- hierarchical structure used for analysis: at bottom, syntax (with a lexical component "pulled out of semantics, e.g. preferences like case rules); in the middle, semantics; at top, pragmatics.

21.3 Summary of Winograd's paper

Winograd's paper emphasizes:

- a logical form for semantics is not feasible
- meaning depends upon context (e.g. "bachelor" and "water")

*Digression...a consideration of the problems caused by the metaphorical nature of language: "semantics is a notoriously slippery business" is the starting sentence of the book *Situation Semantics* by Barwise and Perry and is unanalyzable by the theory they present in that book!

21.4 Answers to Speech Understanding questions

Why does speech understanding constitute an application of AI?

Speech understanding is a signal to symbol transformation problem. Production of speech can be viewed as a long chain of successive transformations from intentions to semantic and syntactic structuring to the eventually resulting audible acoustic waves. Interpreting speech effectively amounts to inverting these transformations to recover from the speaker's intention from the sounds. At each step in this interpretive process ambiguity and uncertainty arise, (see Erman et al paper on Hearsay to get a sense of the levels of transformations and the kind of ambiguities that can occur at each step). The solution to this problem uses AI techniques for coordinating multiple sources of information and to handle uncertainty.

The difference between speech understanding and speech recognition.

Speech recognition requires correctly identifying each word; in speech understanding, partial recognition is acceptable if the meaning or intent of the sentence/phrase is identified.

Compare and contrast the speech recognition and the image understanding problem.

Similarities between speech recognition and vision:

-both processes are of the signal to symbol transformation type

-both need the coordination of multiple sources of information for interpretation,

-noise expectable and problematic

-both attempt to identify "deep structure" (objects can be seen from different perspectives, meanings can be conveyed in different ways)

The only difference between the two is that sound has a 1-D signal varying with time, while vision has a 2-D signal (which could vary on the time axis if we are studying motion or animation).

Problems with the template approach in isolated word recognition systems.

a) background noise

b) different pronunciations (different speakers)

c) the same word is not spoken the same way every time (same speaker)

d) instrument error

The template matching relies completely on a distance function that compares the input acoustic pattern with a prestored pattern. Hard to make allowances for the above problems in this framework. See the Baker paper for some interesting examples of these problems listed above.

Problems with continuous speech recognition.

Continuous speech recognition is much harder, because the way in which a word is pronounced changes according to its context in a sentence. Boundaries of individual words are hard to find. More knowledge is brought to bear upon this problem by humans who can understand speech in very noisy surroundings and as spoken by many people. Expectations about the form of the utterance and the content of the utterance are created based upon the hearer's knowledge of the syntax and semantics of English language utterances coupled with the the knowledge of the subject being discussed. The solution to the speech problem will require utilization of these constraints.

What was the ARPA SUR project?

The ARPA SUR project was carried out between the years 1971-76. The goals were to build a speech recognition system that should accept connected speech from multiple users, taken from a pre-set vocabulary of about 1000 words, in a highly constrained artificial syntax or a highly constrained task, with a tolerance of about lOreal time on a 100 MIPS machine, in a quiet room, with only slight tuning needed for individual speakers, the systems developed under the ARPA SUR project were Hearsay I, Dragon, Harpy and Hearsay II at CMU, Speechlis and HWIM at BBN and SRI/SDC at SRI.

What were the major lessons learnt from the ARPA SUR work?

HARPY met the performance goals set up for the project. The work on Hearsay II influenced system design in many areas notably vision, X-ray crystallography interpretation, common sense planning, signal interpretation and more recently protein structure determination. The success of HARPY represented a tremendous engineering success attributed mostly to the low level processing. Rules for expressing phonological and acoustic-phonetic regularities were built. New parsing strategies were developed (see the Woods paper in the Webber-Nilsson collection. The use of redundancy to cope with uncertainty was well demonstrated in the blackboard architecture.

Main features of Hearsay II

Main idea : independent knowledge sources cooperatively solving a problem by posting hypothesis on a global data base called the blackboard. Hearsay II used 12 knowledge sources and a hierarchical blackboard. Island driven control strategy. Data directed knowledge source invocation.

Main features of HARPY

Single compiled network incorporated knowledge at all levels

Beam search

Used heuristics to limit search time and size of network

Disadvantages of HARPY **Cannot be easily extended to include pragmatics. Sensitive to acoustical segments and missing words.**

Where the BB idea is good

Domains where the search space is large, there is a need to combine different kinds

of knowledge and where ambiguous or noisy data obtain. See Barbara Hayes-Ro
paper in the readings as well as her more recent papers on BB1.

Chapter 22

Guest session on Natural Language

Guest : Barbara Grosz, SRI

22.1 Outline of session

The discussion had five parts:

1. a brief history of NL research;
2. overview of different ways of classifying research on NL;
3. discussion of natural-language interfaces and how to evaluate them;
4. research on natural-language generation;
5. current *hot* research issues.

22.2 Brief overview of history of NL research

Until recently (about 1980) most research in natural-language processing was concerned with issues of *understanding* natural language; what little work was done on generation followed that on understanding. For example, the basic representa-

adapted from those used in parsing. The earliest work on generation is that of Simmons and Slocum (CACM, 1972) where English sentences were generated using an ATN and a case grammar (i.e an ATN which accepts NL was turned around to generate sentences). Now the need for 2-way communication with computers has motivated work on NL generation. But before going into the more recent work in generation, we will give a short history of work in NL.

The earliest work (pre-sixties) focused on machine translation. In the early sixties there was an attempt to build natural language understanding systems in a particular domain (e.g BASEBALL and SIR, see the Handbook for short descriptions of these systems, the original descriptions are in Minsky's Semantic Information processing). In the early 70's there was more ambitious work : Winograd built a very sophisticated natural language system that worked in the blocks world (SHRDLU) and Woods constructed a system for querying a database (LUNAR); this system used the ATN formalism as the basis for parsing; Woods paid special attention to problems of quantifier scoping. In both these systems, the information needed for constructing an internal representation of what a natural-language expression meant - the "semantic" components - was encoded proceduraily. The two systems emphasized different aspects of the problem: Winograd worked on a particular domain and handled several NL features (e.g anaphora) whereas Woods worked on handling quantification and the development of general parsing techniques. This dichotomy in approach is present in work in other sub-communities of AI. (see expert systems research vs. McCarthy's emphasis on non-monotonic reasoning).

In the early to mid-seventies, Eugene Charniak worked on story understanding (reading children's stories and answering questions about them). There are several problems in doing this: first, it requires an adequate treatment of a large range of discourse phenomena, e.g., pronominal references have to be handled correctly. The second kind of problem is illustrated with an example : the fox and the crow fable. To understand this story, we need to know the concept of flattery and a lot of common sense knowledge about birds and beaks. This is not part of the story and cannot be got out of a literal analysis of it. See Dreyfus's remarks on this particular example in his "What computers can't do" (2nd edition). Also see Winograd's subsequent work in "What does it mean to understand language". In any case, Charniak built a system which used demons and data structures that were precursors of the scripts and frames used

today. His main concern was with the combinatorial explosion in the number of inferences that one could perform in the system and not on semantic cleanliness. Contrast his work with Moore's "Notes on logical form" (SRI tech note). Charniak's system did not start with actual English sentences, but rather started with a hand-constructed representation of what the sentences meant. As a result, several important properties of the sentences in a story (as well as the pictures that accompanied them) were ignored; as a result Charniak lost information that was in the original story. This "tradition" of ignoring the surface form to concentrate on other problems continues in certain work of Schank and his students and their students.

In the mid-seventies, a lot of work went on in speech understanding research (DARPA funded a large project on this problem). Three major systems came out of this : HARPY (CMU) : this was the most successful and had the least AI (but had excellent low level pattern recognizers), DWIM (BBN) and the SRI/SDC systems tried to attack the problems at the higher end of the speech recognition system (parsing, semantics, discourse). In summary, these systems did very well considering that computers are deaf, blind, mute etc. The AI contributions from these systems included : the blackboard architecture developed for coordinating multiple sources of knowledge at CMU, integration of syntax and semantics in an ATN formalism (RUS at BBN), the DIAMOND/DIAGRAM parser at SRI which formed the background for subsequent work on TEAM (one of the most sophisticated NL front-end system builder we have today, development of partitioned semantic network formalism and systems for reasoning with them. The BBN and SRI systems also included discourse components that were among the first attempts to identify and solve individual problems in discourse – e.g., provide general treatments of referring expressions and speech acts.

Roger Schank did his work on MARGIE at Stanford in 197?. The emphasis was on controlling inferences. The attempt was to give a computational account of story understanding. See notes on the in-class discussion on NL for more details on Roger Schank's work.

In the late seventies, research in NL shifted to something quite practical: NL interfaces. Waltz's PLANES, Burton and Brown's SOPHIE, and LADDER were NL interfaces based on semantic grammars. Each was a special-purpose interface; it handled a wide-range of queries, but only for a limited domain and single task. The

subarea of NL that has an expert systemy flavor is the work on NL interfaces based on semantic grammars. Experience with LADDER led to more recent efforts to develop an interface that could be easily adapted to new domains; this research resulted in the TEAM system.

In theoretical NL, there was a shift towards problems at the discourse level; discourse content, language and action, speech act theory. Ideas from philosophy, AI, linguistics and mathematics were integrating to form new theories of language. This integration forms the basis for the program of research at CSLI, including work on situation semantics (Barwise and Perry), discourse and action, and various grammar formalisms (LFG, GPSG, PATR-II, FUG) as well as certain work in planning and action.

22.3 Classifying NL research

Research differs along a number of dimensions:

1. interpretation vs. generation

Fallacy: only interpretation really matters. This is a fallacy because to communicate with a person, a computer system has to respond reasonably. Although this may not seem to matter with simple database query, it does with almost any other interaction. Consider what happens if the DB system doesn't have an answer, or an expert system has to explain why one answer and not some other (when the response can't be known in advance and so canned).

Also, generation is an important test of any theory because without adequate constraints it is easy to generate incomprehensible discourses. This problem arises at the sentence level (grammars that overgenerate) as well as at the discourse level (saying something a user can't understand for semantic or pragmatic reasons).

2. concern with computational explosion or theoretical adequacy: compare Graeme Hirst's and Bob Moore's work on various issues in semantics.

3. goals: system qua system vs. system as an experimental tool

related issue: constrained domain (e.g., cooking, airplanes, mail system) vs. constrained task (e.g., database query, advice giving).

4. task - database query, interaction about task, translation, expert system interface, story understanding, newspaper reader. Questions to ask: what are the constraints of the task? its demands? what hidden information is there that depends on the particular task?

5. kind of interaction: text vs. dialogue

6. where does the generality lie? where is the power? syntax? commonsense reasoning (e.g., scripts, frames), discourse? etc.

22.4 Evaluating NL systems

Wherein lies the generality of a system? Suppose you are going to evaluate an NL interface as a consultant for some venture capitalist, what would you try to break it?

- **Quantifiers**
- **Conjunctions (...submarines in and ports on the Atlantic)**
- **Speech acts : show, list, what if, etc**
- **Modals**
- **Noun-noun combinations (Lafayette class submarines, French ships)**
- **Pragmatics (the crow and fox fable)**

Most systems on the market today are based on semantic grammars which do not handle pragmatics very well. If any of these systems is asked to answer a query like : Who is the commander of all ships?, it should point out the pragmatic ambiguity: Is the answer the President or is it the list of ship vs commander of that ship? Similarly the question : What is the height of all peaks? really requests a listing of peak vs height for every peak, and to interpret this request correctly the system should have knowledge of the functional relationship between peaks and their heights.

22*5 NL and expert systems

Most expert system researchers feel that the explanation problem will be adequately handled when NL is solved. As of today, expert system technology is totally inadequate for supporting NL generation. Most systems just used canned generation (cf

MYCIN). There is interesting research in this area, e.g., Swartout's at ISI.

22,6 Natural language generation

Contrast the NL generation/interpretation dichotomy with that of work in AI generally on plan generation and recognition.

Here now is the promised review of NL generation work. In the early 70's, there was the Simmons and Slocum work referred to before. After this, work proceeded along three distinct paths.

- surface form
  - how to say what's to be said in a single utterance; this is McDonald's work
- what to say and how to say it at the discourse level
  - this includes McKeown's work on rhetorical patterns and focus as well as Appelt's work on what the hearer believes;
- control issues - coordination of what to say with how to say it as well as when to make which decisions

Clear separation between the What and How boxes: McKeown's "how" component could not send information back to the "what" component.

Appelt's work coordinates the two decisions and there is cross talk between these two components. In particular he allows both the how and what to affect each other. Doug Appelt's generator : Language as action paradigm. Worked as follows: Determined the sort of illocutionary act that needs to be done (inform, request etc), the surface speech act was then determined (making a statement, etc), activation of concept, the utterance was then generated.

To get an idea of the subtleties in NL generation consider the two 'equivalent' sentences

: Your mother called / My mother-in-law called (to your spouse). Which convey more than just the surface meanings indicated. Winograd's example "We shall eat dinner after the children wash their hands" is another.

In NL generation, research in commonsense reasoning, planning, knowledge representation and reasoning with actions are unified.

22.7 Other questions

- What is the difference between planning in a communicative situation and planning in the blocks world?
  In traditional Blocks world planners, the domain is constrained and at any point we know what another player knows. In NL generation we have incomplete models of the world and worse, may have little knowledge about the specific hearer. The rules of the game are also unknown.
- Reasoning about time and events.
  Is very difficult. Consider "I went to LA last week. The trip was great." To understand this we need to know that 'the trip' refers to the 'going to LA last week'.
- Machine Translation
  Surface translation (without going through a inter-lingua can be fairly robust in technical domains, e.g manuals). This has actually been done under Slocum at the Univ. of Texas, Austin. The credo here is to pick the right problem to suit existing methods. No generality.
- Yale work.
  The right way to interpret their work is as the expert system approach to NL. They have spent an awful lot of time spelling out such things as how restaurants work, i.e encoding the facts of the matter to be used in NL systems. They have picked fascinating problems and produced interesting, but limited systems. They are ad-hoc and lack generality. They do not care about the surface form of the utterances they generate. Common sense reasoning and memory are their main research interests and language is a good vehicle for research in these areas.
- NL interfaces.

These do not form the right cut at the NL problem. The utterance and objective perspective is a better one, because it gets at the crucial issues in NL.

— User Models.

Work to date on "user models" can be summarized in one word "terrible". There is some work in natural-language processing on problems related to such models. This includes the work of James Allen at the university of Rochester; he and his students are looking at the plan recognition problem. Also, Martha Pollack at the University of Pennsylvania is looking at methods of generating appropriate answers to questions asked by new users of the mail system.

— On Terry Winograd's views on NL

Terry's premises are right and his concerns are serious but his conclusions are overly pessimistic. We need to be grounded in the world for understanding language, the connection to the world is very important. Hence we should start with domains where we have to deal with a reality that is inside a computer system and to which the machine has access to, to build successful NLU systems (at least until we have well-functioning robots).

— Geographical organization of work in NL.

* Stanford : Anne Gardner, CSLI
* MIT : Robert Berwick (syntax and language acquisition)
* BBN : Database interface project, Sidner's work on discourse structure and plan recognition.
* Rochester : James Allen (plan recognition problem, language as action paradigm)
* Yale : Birnbaum (arguments), Schank
* Columbia : McKeown (student advisor program), Leibowitz
* Bell labs : Intonation, speech work, Mitch Marcus' parsing work
* Penn. : Interfaces, explanation, scalar implicatures
* CMU : Jaime Carbonell (this is not really a center for NL research, SUR mostly)
* USC/SCI: Explanation (Swartout), Systemic grammar, NL generation(Mann).
* Berkeley : Wilensky (continues the Yale tradition)

* UCLA : Mike Dyer and Margot Flowers (also the Yale tradition)
* University of Massachusetts: Lehnert, McDonald
* Brown: Charniak
* SRI: parsing, grammars, semantics, discourse, generation, interfaces : a too many people to list

Chapter 23

Expert System Applications I

AI in Medicine

- *Inexact Reasoning*

  Both Mycin and Prospector use inexact reasoning. Dendral does not. When do we need it? Is it related to how experts in the domain reason? Is it because of the (limited) extent of our understanding of the domain?

- *Expert vs. Emycin*

  Both of these systems use inexact reasoning and production rules.

  Emycin is top-down. It has a fixed way of combining uncertainties. Also, it has an explanation facility built in.

  Expert is bottom-up. The Findings are input in a questionnaire format. The user can specify the combining function without digging into the depths of the code. It separates the rules into three types: FF FH HH. FORTRAN.

- *Expert vs. IRIS*

  Both of these systems use forward chaining. Both have inputs by simple-minded questionnaires.

  IRIS is a framework that allows experimentation in many aspects of your expert system. It does not use pure production rules; it also has semantic nets. It was possible to use this system to encode Mycin's therapy selection algorithm (this couldn't really be done with Emycin). The reasoning scheme could be

Why couldn't Emycin encode Mycin's therapy selection algorithm? The therapy selection algorithm is an optimization problem that seeks to provide coverage for organisms while minimizing the number of drugs prescribed. MYCIN's inferential rule representation was inadequate because of the inherent algorithmic (procedural?) nature of the problem (i.e iteration and complex data structures).

- *PIP*

  Questioning can be erratic. A physician concentrates more on one active hypothesis. It had a poor way to decide when to stop the questioning.

- *Internist-I*

  It's goals were to capture the knowledge needed for the whole of internal medicine. It allowed multiple diseases. It was an attempt to model clinicians. First, the user entered lots of symptoms. Diseases from a disease tree were then invoked. (Internist-II did this more efficiently, by evoking classes of diseases.) Everything after that centered around the disease tree. It used multiple questioning strategies, such as: rule out a disease, get more evidence for a disease, distinguish between two sets of diseases, ...

- *The Mycin Family*

  Teiresias [Davis '76]: explanation and knowledge acquisition. Guidon [Clancey '79] $\Rightarrow$ NeoMycin: tutoring and explanation. Puff [Fagan '79]: a use of Emycin. Emycin [Van Melle '80]: Essential Mycin – framework for building expert systems. Centaur [Aikens '80]: frame-based implementation of expert system.

- *Digitalis therapy advisor*

  This combines a qualitative and a quantitative model.

- *AI/MM*

  By Kunz in '82, this combined causal, shallow and mathematical methods. It could generate coherent multiple-paragraph explanations.

- *CASNET*

  The causal model in CASNET associated diseases to findings. The causal model allowed better explanation possibilities. It could also capture temporal aspects of diseases, and give a prognosis for the disease. It was a more understandable representation that (probably) allowed less rules to be used; this would result from a better structuring of the rules.

- *Knowledge Acquisition*

  Mycin had simple and stylized representation that facilitated interactive transfer of expertise. What KA activities went on in the context of the other AIM systems?

- *Explanation vs. Tutoring*

  Explanation on most Expert Systems is added as an after-thought. Wallis & Shortliffe tried to have a system where the "user-model" was just a number from 1 to 5, reflecting the expertise of the user. Also this work used the assumption that providing increasing detail of the inference process enhances explanation. See Clancey's paper on "The epistemology of explanation", to get another viewpoint. John Seeley Brown contends that CAI systems can not just be "expert systems warmed-over". Tutoring needs an explanation component that is tailored to the user, in addition it needs a critiquing module (for diagnosing student errors).

- *The Dendral Family*

  How did Dendral (it used a Plan-Generate-Test cycle) get away without having any form of inexact reasoning? How was it validated? – just like any other researcher. Meta-Dendral learned half-order theories for classes of molecules. Congen was the GENERATE part of Dendral, spiffed up a bit; it is the part that is actually used by researchers who use mass spectrometers.

  Why were later research issues taken up in the context of Mycin instead of in the context of Dendral?

- *Random Questions*

  * Why aren't these computer-based systems used in the real world?
  * What are the limitations on statistical approaches to these systems?
  * What are the limitations of Teiresias?
  * Is it good or bad to have Meta-rules have the same format as normal rules?
  * How is/should uncertain information be captured?

This is a list of questions on the material in Chapter 7 and 8 of the AI handbook. These questions are in no particular order. They range from being factual to being thought-type in nature.

- What is the need for inexact reasoning? Answer this w.r.t MYCIN and PROSPECTOR. For what sorts of domains and problems would you anticipate that inexact reasoning would be used?
- More association questions – who, what, when ,where, why SACON, PUFF,R1
- Name some VLSI circuit design aids developed using expert system technology. What aspects of the design process (in any domain) can be automated?
- What are the kinds of explanations explored in the context of expert systems? There are five kinds of explanations that can be offered for a given phenomenon.
  * *Deductive/nomological* : Laws of nature are cited to explain an event. e.g the height to which a ball rises in the air when thrown can be explained using Newton's first and second laws.
  * *Morphological* : Here the structure of an event/device is used to explain it – gene replication explanations are of this sort, so is the explanation of how a fibre glass bundle of fibres conduct light.
  * *Systematic* : This is best understood with the help of an example. The explanation of the operation of a car is in terms of the various subsystems that constitute it and the interactions between these subcomponents (which themselves may be explained by any of these methods).
  * *Reduction* : Attempt to find first principles. Newton's derivation of Kepler's Laws and the reduction of thermodynamics via statistical mechanics are examples.
  * *Procedural* : For want of a better word, we use this. This is the sort of explanation that MYCIN offers, namely the problem solving steps undertaken form an explanation of MYCIN's behavior. What are the disadvantages of this form of explanation if it is to be used in another context : e.g tutoring, addition of new knowledge (e.g control knowledge).
- What aspects of MYCIN are hard to explain?
- What is Swartout's approach to explanations?
  See his paper "XPLAIN: A system for creating and explaining expert consultation systems", AI Journal, Vol 21, No.3, 1983. Is good explanation at odds with good performance?
- How does your favorite tutoring system give explanations?

- For each system mentioned in these chapters, get a sense of the following
  - * a description of the problem attempted
  - * Kinds of knowledge represented and the rep. formalisms used
  - * Reasoning methods used. How uncertainty is handled
  - * How easy is it to add learning to the system.
- What is a rule model in TEIRESIAS? How is it constructed? How is it used?
- How are meta-rules used in TEIRESIAS? Meta-rules in TEIRESIAS were expressed in the same way as the base level rules. What are the trade-offs associated with this design decision?
- How did they get away without inexact reasoning in DENDRAL (or did they?)
- How does DENDRAL do structure elucidation? What knowledge of mass spectrometry in encoded in it?
- How was DENDRAL validated?
- What aids have been developed to make the use of CONGEN easier? MACSYMA?
- How did META-DENDRAL represent its evolving knowledge of mass spectrometry? How does it constrain search in the space of all rules?
- Can Meta-dendral undo rules? What additional capabilities would it need to do that?
- What are the two major approaches in the systems which do chemical synthesis? How do they differ from AI blocks world planners?
- What were the reasons for exploring basic research issues in explanation, tutoring and interactive transfer of expertise in the context of MYCIN, as opposed to DENDRAL?
- What are the limitations of current-day computer based medical consultation systems which limit their acceptability in the real world? What are the efforts to improve them along this dimension? (cf ONCOCIN)
- Limitations of the statistical approach to doing medical diagnosis.
- What do VM and RX do?

- **How is uncertain information and uncertainty in reasoning captured in each of the medical expert systems in Chapter 8.**
- **How is evidence combined in these medical expert systems?**
- **AI in medical decision-making has been a test bed for what basic issues in AI?**
- **Why do we separate categorical and probabilistic reasoning in MYCIN?**
- **Knowledge rep. formalism in MYCIN. Advantages and disadvantages.**
- **Why are CF's used as opposed to standard prob./statistical measures.**
- **Reasoning mechanism in MYCIN. Which other medical AI system does exhaustive search?**
- **What is the unity path hack in MYCIN? partial evaluation?**
- **How is therapy selection done in MYCIN?**
- **What sorts of knowledge can TEIRESIAS not acquire?**
- **What is the nature of the causal model in CASNET?**
- **How is diagnosis done using the causal network?**
- **What can CASNET do that MYCIN cannot and vice-versa?**
- **Would it make sense to compile a shallow model of glaucoma diagnosis from the 'deeper' model that CASNET has? What are the advantages and disadvantages?**
- **How is time captured in CASNET - it can reason about the progression of a disease.**
- **How does CASNET gather evidence? In what manner does its questioning strategy differ from MYCIN? Why?**
- **What are the major goals of the INTERNIST project?**
- **What is the difference between INTERNIST-1 and INTERNIST-2?**
- **Compare INTERNIST-l's style of diagnosis with the way MYCIN works? AQ11?**
- **What is the disease tree in INTERNIST? What is its counterpart in MYCIN?**
- **What is INTERNIST'S questioning strategy?**
- **How does INTERNIST'S maintain focus in its diagnostic reasoning?**
- **What is the major focus of the IRIS project?**

- What is special about IRIS's KR scheme? What epistemological claims do th IRIS designers make about their framework?
- What is the questioning and reasoning strategy used in IRIS?
- What efficiency hacks are programmed into IRIS?
- What is the major focus of the EXPERT project?
- Compare EXPERT with another framework system like EMYCIN.
- How do EXPERT and IRIS compare?
- What is the KR scheme used?
- Reasoning strategy used and the evidence gathering mechanism.
- What are the goals of the Digitalis Therapy advisor?
- Chief features of the advisor?
- Validation of the advisor and explanation research in the context of the adviso
- Goals of PIP.
- Representation schemes used.
- Reasoning methods.
- Problems with this approach to medical diagnosis.

Mycin; 1976; Buchanan and Shortliffe; Stanford

What: Diagnosis of and therapy for bacterial infections.

How: Exhaustive depth-first backward chaining of production rules.

Distinctions: Use of "certainty factors"; extensive formal evaluations which showed expert performance. One of the early "real-world" systems.

Casnet [Causal ASsociational NETwork]; 1977; Weiss, Kulikowki and Safir; Rutgers

What: Medical diagnosis - major application to glaucoma. Treatments prescribed. Course of disease with and without treatment predicted.

How: Three planes of knowledge: disease states, patient observations, classifications of diseases based on presence and absence of states in first two planes.

Reasoning propagates confidence factors through the net from observations via causal links between nodes in the planes. Bottom-up approach.

Distinctions: Evaluating opthamologists called it expert Representation makes expansion of disease model easy.

Internist: 1977; Pople and Myers; Univ of Pittsburgh

What: Disease diagnosis in internal medicine from symptoms, lab data, patient history. Several diseases may be present simultaneously.

How: Knowledge stored in a disease taxonomy. Diseases associated with their manifestations. Observed manifestations "evoke" related diseases which arc ranked by comparing observed and expected manifestations. Basically matches obscvations to categories. First version was exhaustive, second used certain key manifestations to initially limit the search to certain disease classes.

Distinctions: Large knowledge base, expert performance.

PIP (Present Illness Program), 1976, Pauker ct al, MIT

What: Diagnosis of renal disease from relatively limited patient data.

: KR using frames of diseases, physiological states. Disease descriptions include associated patient findings categorized for ranked matching to actual data (e.g., sufficient, crucial, necessary, and incompatible findings). Frame hierarchy groups related diseases. Simple control structure of: observe, modify hypodicses, formulate question.

inctions: Categorical and probabalistic reasoning.

Therapy Advisor, 1977, Swartout, MIT

it: Treatment regimen for a complex drug - patient management History, body weight, renal function, symptoms, toxicity.

r: Combines mathematical models of digitalis concentration changes in die patient with qualitative models of patient needs.

r8, Trigoboff and Kulikowski, Rutgers

it: A tool for medical decision making. [Designed for experimentation with modes of KR, clinical strategies, modes of user interaction. Actually used for glaucoma diagnosis.

r: Combines semantic nets with production rules. Seperate representations of patients and medical knowledge. Symptom nodes linked to disease nodes linked to treatments. Rules associtaed with links produce inferences.

979, Weiss + Kulikowski, Rutgers

it: General tool for design and testing of consultation models. Classificaton problems are best suited to is approach.

t: Program compiles a rules language enterred by the user. A consultation model consists of hypotheses, findings, decision rules. Uncertainty is represented. Rules take findings to hypotheses to move hypotheses (forward chainging). Hypotheses are ranked in various ways. A number of different systems were being built with this tool.

ICAI

1970, Carbonell, BBN

What: Mixed-initiative (student or system can ask questions) tutoring system for South American geography. They want to deal with representing real-world knowledge and domain-independent ways of making plausible inferences from incomplete databases.

How: KR is semantic nets. Natural Language based on a case grammar system. Inferences done by tracing semantic net links. It also reasons about the extent of its own knowledge to some degree to answer questions, which is rare in AI.

WHY, 1978, Stevens + Collins, BBN

What: A tutor on the causes of rainfall. Emphasizes a domain where causal and temporal relations exist among many concepts instead of just the facts.

How: Focus on: (a) qualities of a good tutor's questions, statements, examples, (b) student misconceptions, (c) abstractions used to explain physical processes. They analyzed actual student-tutor dialogues. Came up with heuristics. KR is scripts. They attempt to model a students view of a model of a process the system possesses.

SOPHIE; 1976; Brown, Rubenstein, Burton; BBN

What: Explore bbroad student initiative in context of an electronics labboratory. Student diagnoses malfunctioning equipment by making measurements.

How: System has knowledge of the domain plus rules for answering questions, criticiaing hypotheses, suggesting alternate explanations. Student asks questions to find faults and system answers and monitors progress, probing students actions with questions. Has a NL system too. Tasks: hypothesis generation, evaluation. Accomplished by circuit simulation.

WEST; 1976; Burton and Brown, BBN

What: Computer coach for a child's game - it looks over the student's shoulder and gives suggestions/criticism.

How: Concentrate on diagnostic strategies and tutoring strategies. Students may discover they have misconceptions; system should help students disambiguate the causes of their misconceptions so these bugs become constructive. Compare student's actions to an expert's actions. Analyze how a

US; 1977; Carr, Goldstein; MIT

at: Another coach for a silly game.

w: Four modules: Expert tells Psychologist how student's move is non-optimal. Psychologist decides hos student's knowledge is lacking and modifies Student Model, which guides Tutor's interaction with the student. Each module is rule-based.

tinctions: A single system has several distinct areas of expertise.

)N; 1979; Clancey; Stanford

at: Mixed-initiative dialogue with prolonged structure. Conversation about an infected patient based on Mycin's rules, which are the skills to be taught.

w: Seperate tutorial rules for guiding dialogue, presenting diagnostic rules, constructing student model. Q: Are Mycin's problem solving rules useful for teaching as well? Student model assumes student's knowledge is always a subset of Mycin's. A weakness is that Guidon sees the student only as lacking Mycin's rules, as opposed to having other rules, correct or incorrect.

l: Students find Mycin's rules hard to remember. Much strategy is implicit in the rules but is not communicable by Guidon.

Y; 1978; Brown and Burton; BBN

at: Determine bugs in a student's arithmetic skills.

w: Skills are modeled by porocedural nets in which sub-procedures can be made faulty to simulate mis-conceptions. An exhaustive search of the space of possible mis-conceptions allows the student to find mis-conceptions compatible with a given student's behavior.

Chapter 24

Guest session on AI applications

Guest: Bruce Buchanan, Stanford University

The guest for this session was Prof. Bruce Buchanan, the faculty sponsor for this course. We discussed AI applications and answered some questions that Prof. Buchanan raised. he discussion closed with Prof. Buchanan's comments on the AI qual.

[Note taker's caveat: I opted to preserve the temporal order of the discussion in this transcription]

Prof. Buchanan mentioned that he was uncomfortable with the split : Expert System Principles and AI applications that the class has adopted. The experimental methodology of AI research constituted using an application to drive research in fundamental issues in AI (cf DENDRAL, MYCIN, and more recently PROTEAN).

There are a lot of application areas and it is not clear what a qual taker should know. This point was clarified with the help of the following analogy : How much should a student taking an advanced physics qual know about bridges? He should certainly know about very important bridges (part of his common sense database, presumably !) and should be able to derive the principles upon which a bridge is built. This is the extent to which an AI qual taker should know about AI applications. Most of these are still straightforward application of 1960s-70s technology and are important from the commercial perspective. More important are those applications where the existing methodology is challenged and which have lead to research in basic issues in

Thus, qual takers should be familiar with MACSYMA, DENDRAL and MYCIN which were the 'original' expert systems. Programs covered in the AI handbook in chapter 7 through 9 are also important from the perspective of the research insights they gave us. Contrast this with the AI research on games that went on in the sixties that have found their place in the commercial environment - the checkers program of Samuel, the backgammon program of Berliner and Greenblatt's chess player are now coded into PC's.

This lead to the question : When is applications research construed of as constituting research in AI? Prof. Buchanan cited an IFIP paper (1971) by Feigenbaum - AI suffers from the phenomenon that once a problem is successful, it moves out of the realm of AI.

There are two research methodologies in AI - top-down, issue-driven and bottom-up, application-driven. In the latter the hope is that the specifics of the application will guide research. Research in the latter style has been extremely successful in the last decade. An example of research of the former type is nonmonotonic reasoning that is pursued independent of a particular application.

Why is it that expert systems are identified with science, medicine and education and not any others? This is probably a historical accident.

Why would people want expert systems in the industrial context? What applications of expert systems could there be besides medicine, science and education? Business (financial management), Operations research (non-linear optimization), civil engineering (SACON is a consultant built using EMYCIN which advises people on the use of a FEM package), space (let your imagination soar here).

Why is automatic programming not an application? I.e why does it merit a separate chapter in the Handbook (and not be clumped with 'expert systems')? This question led to renewed discussion on what an application was. One viewpoint was that an application of AI was whatever part of AI has found its way into practical use in industry. Another view was that an application area was one that formed a test bed for experimenting with fundamental issues in AI (knowledge representation, learning, explanation, uncertainty, common sense etc.). It was noted that the AI ideas that are in regular use are those that have been tested and tried. For instance, industrial vision uses the simplest of pattern matching using structured light to eliminate shadows, for

chip inspection (look up the Chin paper for details on this). Industry uses only those techniques for which it can be demonstrated beyond reasonable doubt that the potential benefits outweigh the potential risks. The situation is not very different from the use of experimental therapy and time-tested medication in the field of medicine.

Is Natural language an application of AI? This was to be interpreted as : is there a research component to work in NL? Research in discourse understanding, speech act theory, plan recognition, modeling of beliefs of speakers and hearers, NL generation involve extending and contributing to research in knowledge representation, common sense reasoning and planning. If the NL problem is viewed in the narrower context of building front-ends to data bases and expert systems, we could call it an application.

Progress in natural language understanding amounts to progress in AI. This was an instantiation of the claim that if progress in X amounts to progress in AI then X constitutes an application for bottom-up AI researchers. Understanding any science does not necessarily tell us anything about intelligence, but understanding a complex cognitive ability like language does help in the process of understanding human intelligence. But what of NMR interpretation? Humans do not do it routinely. What can it tell us about intelligence? Or are we modeling a certain kind of problem solving behavior?

Prof. Buchanan pointed out that it was better to make a division based on problem classes as opposed to making one on the aspect of human cognition being modeled (as was suggested in the previous paragraph). Bennett made a start on this in his ROGET system. Simon talks about the need to determine a taxonomy of problem types in one of his essays in "The sciences of the Artificial". What expert system companies are engaged in, is matching problem types to problem solving methods (or else fitting applications to expert system building frameworks). This constitutes valuable contribution (data points) to our understanding of problem solving. Read Clancey's paper which gives a knowledge level analysis of a particular type of problem solving, called classification problem solving. Open problem : What other categories are there? For instance, where does the monitoring problem fit? Prof. Buchanan wrote up an essay on problem types which appeared in the proceedings of the Philosophy of Science Association. It also appears as an HPP memo (HPP-84-??).

Are there any successful applications of AI to business? Yes, there are some commercial systems that provide advice in financial management. They have been used to ensure uniformity of lending policies across different branches of a bank. There are some other systems that are used for advising insurance policy buyers.

What are the two main issues in AI since the 1950's. Representation and reasoning. In the context of applications, more issues were unearthed - explanation and tutoring, interactive transfer of expertise, user modeling and user interfaces.

How do you validate an expert system? There is no consensus on how an expert system should be validated. Basically, we need to devise a Turing's test for that system. Why do we need to evaluate expert systems? In the first place, if one were to buy one, one would need some proof of the fact that the expert system does satisfy the stated specifications. Also there are a number of legal, moral and ethical issues that have to be resolved before expert systems can directly or indirectly take 'critical[1] actions. The issues involved here are exactly analogous to the tests that drug manufacturers have to undertake before releasing a new drug on the market.

What sorts of uncertainties need to be captured in expert systems? Uncertainties in data (data may be incomplete or noisy), uncertainties in the rules (rules may be missing or wrong), uncertainties in the conceptual framework. The last one is the hardest to detect and recover from. Dendral handles uncertainties in data (noisy peaks) by simple thresholding. This is the approach used in most edge-finders in vision systems. In MYCIN, we can tolerate incompleteness and inaccuracy in data as long as there is enough evidence for correct interpretation (cf the 0.2 threshold in CFs). The basic approach that AI has devised to cope with uncertainty is the exploitation of redundancy. This is well illustrated in the Hearsay architecture (Erman et. al).

How does learning proceed in a noisy environment? Most learning systems make the assumption that the data is classified correctly (Winston's arch learner and Mitchell's version space method). In fact, all data driven learners have very poor noise immunity and a single noisy instance may throw them way off course. Model driven learners have strong models of their domain and can have reasons for believing why an instance was classified positive or negative. Thus they can weed out mis-classified instances by invoking a theory of the domain. Meta-Dendral deals with noise in this manner (has a half order theory of the domain). Li-min Fu's thesis explores the question of

learning both base level and meta level rules in the presence of noise.

What were meta-rules in TEIRESIAS used for? They were used primarily for conflict resolution. Why did the MYCIN systems end up with few meta-rules? This is because the meta information is hard-wired into the rules in the form of ordered conjuncts and ordered rules. Clancey attempted to separate control information in the rules and place them in the meta level and his system NEOMYCIN has a larger number of meta rules.

What are the advantages of having the same representation for the base and meta levels? There is no need to rewrite the interpreter, the one that works for the base level, works for the meta level too. We only need to introduce symbols at the meta level that can talk about sets of rules and their properties. The same knowledge acquisition methods, explanation routines can be mapped over to the meta level too. If it was cumbersome to express control (or other meta level information) in the restricted IF-THEN syntax of the base level rules, there might be reason to have a different representation at the meta level.

How does Dave Smith's work differ from Randy Davis's work on TEIRESIAS? We did not answer this question. Prof. Buchanan remarked that this is a question that a qual taker will not have to know for now, because Dave's thesis is yet unpublished !

On the subject of the qual, Prof. Buchanan emphasized that the qual will be an oral exam (there was some discussion on the prospect of a written version, but in Prof. Buchanan's opinion an oral exam was a better means for testing the breadth and depth of a student's knowledge in the field). The candidate will be tested on how well he has grasped the fundamental issues of the field. Knowledge of the history of work in the various sub-fields, comparison questions, basic stuff like writing a Lisp program or doing a proof by resolution, and some depth in the sub-field the candidate wishes to do a thesis in will be tested also. This is a 1 hour exam administered by a committee of 3 faculty members.

Chapter 25

Guest session on Advanced Topics

Guest : Nils Nilsson,5*an/oncf *University*

This is a transcription of the session with Prof. Nilsson. I apologize in advance for all the errors and misquotes in the following.

25.1 Characterizing AI

It is very hard to give a definition of AI - any attempt either includes too much or too little of the work that gets called AI. In broad terms AI overlaps with systems (read 'fancy programming'). We might try to characterize an AI program by defining the attributes that it should satisfy : e.g

- ***If search techniques are used, it is AI.***
  This is not adequate however,
  because a lot of work in Operations Research which we do not wish to consider as AI, gets included too.
- ***If it uses symbolic computation, it is AI.***
  This does not suffice either, because work done in traditional compiler design

- *If it is knowledge-based, it is AI*

  This does not work very well, either. What is *knowledge-based* anyway? An operating system for a large time-sharing computer has a lot of knowledge. So has a program doing weather simulation. It has a lot of knowledge about atmospheric physics. This does not seem to be the kind of knowledge that AI folks talk about. AI people are concerned with knowledge that can be expressed sententially or declaratively. (cf. all the expert systems). Unfortunately, this is not a very sharp statement. The boundary line between procedural and declarative knowledge is hard to draw.

So, we arrive at the following very narrow definition of AI : representation and use of knowledge expressed as sentences in some logical formalism. This, in Prof. Nilsson's opinion, will be the definition that will survive.

25.2 Behaviorist theories of AI

Stan Rosenschein and Fernando Pereira have developed models of situated automata. An automaton knows something if it acts as if it knew it, irrespective of how explicitly the knowledge is encoded. A robot programmed to avoid a wall, does not need to have a declarative representation of a wall. A deep sea creature (viewed as a finite state automaton) avoids larger sea creatures if it finds itself under their shadow (a light meter will suffice for detection of a larger creature) without ever having an explicit representation for a 'larger sea creature'.

Nilsson feels that this view will help in the design of intelligent systems and that it will also help clarify further the declarative/ procedural distinction.

25.3 Role of logic in AI

Given the definition of AI above, we would look for accumulated wisdom in the representation and manipulation of sentences. This is to be found in logic. An AI scientist without knowledge of logic can be compared to an electrical engineer without knowledge of differential equations and Laplace transforms ! Logic is the foundation of AI if

you are committed to the notion of representing sentential knowledge. Logic provides the lingua franca for communication of technical results in the field. [1]

25.4 Alternative viewpoints

Minsky objects to the use of logic as a KR language for very good reasons. He points out the inadequacies of classical first order logic in capturing human thought. Current work on non-monotonic reasoning is an attempt to capture assumption based reasoning that humans use all the time. Reasoning with time is also an extension over classical first order logic that has been explored by several people. Minsky also believes that some aspects of human thought cannot be codified in logic. "Intelligence is a kludge", he says, and he may well be right !

Nilsson feels that Minsky has not come up with a counter-proposal to logic. The frame idea (1975) was shown to be a variant of first order logic which elevated indexing into the syntax of the language (as Pat Hayes showed in the "Logic of Frames"). Frames spurred a flurry of activity in knowledge representation and led to several useful formalisms (KLONE, KRL, UNITS etc.). However, logic was needed to understand what a frame was and what the operations on it meant. Logic has the virtue of having a nicely specified semantics in spite of its spartan syntax. Issues in default reasoning in frames have been better understood by recasting them in logic.

The society of minds theory of Minsky's will hopefully inspire the same amount of work in AI that his frames proposal did. DAI could be construed as a test bed for that idea.

25.5 What is intelligence?

Nilsson reiterated the Intelligence as computation' view. He clarified the 'computation' notion as – computing with sentential knowledge. Thus, according to his definition a fly (with its excellent vision) would not qualify as 'intelligent'. Obviously, some notion of 'consciousness' is involved in this distinction. The discussion

[1] Why we cannot dispense with logic is explained beautifully in Chapter 12 of the AI Handbook by Bob Moore.

turned to the question : how do you demarcate consciousness/unconsciousness and intelligence/non-intelligence. Nilsson explained that a system or a process can be viewed at several levels and an explanation of a phenomenon can (theoretically) be given at all these levels (we bottom out when we get to atomic physics, now). The question is, how useful is an explanation at a given level. How does our understanding of an octopus' vision improve by postulating mental states for the octopus (or, what does attributing beliefs to our toaster contribute to our understanding of how it works?).

More on the above issues can be found in Daniel Dennett's 'Brainstorms'.

25.6 Grand Tactic for AI

Get at the knowledge needed for intelligent action –maybe first in English (or some other natural language) and then represent it in logic. The expert system community has come up with a method for extracting knowledge (interviewing experts) which will come in handy here. Once, we get the knowledge, we need to explore ways of using it to achieve intelligent behavior(which in turn needs more knowledge !).

25.7 Forthcoming book in AI

Prof. Nilsson and Prof. Genesereth are collaborating in the production of a new AI primer which puts forth the above philosophy of AI. This book consists of two parts. Part 1 is a thoroughly rewritten 'Principles of AI' by Nilsson. It covers material on representation using first order logic, reasoning methods in logic and mechanization of these methods. The second part covers advanced material (a first order correction to the simple theories advanced in Part 1) notably reasoning with uncertainty, non-monotonic reasoning and learning. Their view of uncertainty differs from that of Shortliffe in that : Shortliffe gives an exposition of uncertainty in MYCIN at the 'implementation' level whereas Nilsson and Genesereth aim to be more general and give an account of it at the meta-level : as a layering over standard first order logic. Contrast this with the Halpern/Rabin approach in their 'Likelihood logics'.

25.8 Non-monotonic reasoning

Nilsson characterized two basic approaches to non-monotonic reasoning - ones which augment the base level language with a 'consistency' operator *a la* McDermott and Doyle, and the ones which step outside of the logic to incorporate non-monotonidty at the 'meta-level'. McCarthy's circumscription belongs in the latter category. More details can be found in the chapter on Non-monotonic reasoning from the forthcoming book mentioned above. *NJN's remark : Comments on the chapter are very welcome*

25.9 The robot with continued existence project

This is an AI application that will be a test **bed** for the following research issues.

- Updating beliefs
- Short-term memory, forgetting
- Learning and experimentation
- User modeling
- Reasoning about time and space
- Synthesis of work in perception, robotics, planning, learning, representation

Interesting issues in DAI can be explored if there are more than one of these critters in existence.

25.10 What can we expect from AI in the next ten years?

The most 'impressive' work (work which will be appreciated by society at large) will be by people who are outside of AI (with respect to the definition given above). The most interesting 'theoretical' challenge in AI is common-sense knowledge and reasoning. The research plan to achieve this consists of one half of what Ed Feigenbaum also believes in - namely we need to get at the knowledge first. The research plans advocated by Feigenbaum and Nilsson deviate after this - Nilsson does not believe that a

pure scale-up of expert system technology will suffice, conceptual advances (like circumscription) are needed. Nilsson feels that most work in expert systems subsequent to Shortliffe's have been minor twiddles on the MYCIN paradigm.

- **What are the research issues in knowledge representation?**
  Alternatively, what constitutes a good thesis in KR today or two years from now? Codification of the knowledge in some area (say, naive physics), representation of propositional attitudes, reformulation or reconceptualization (principles and automation methods) of concepts in a non-trivial domain.
- **Wehyrauch's ideas**
  Richard Wehyrauch has contributed two fundamental ideas to AI – semantic attachments to partial models and the use of reflection principles to generate statements in a meta-theory. Both of these are explained with examples in the 'Prolegomena' paper in the Webber-Nilsson collection.

Chapter 26

Advanced Topics I

This discussion took the form of asking and answering questions on Chapter 5 of the Webber and Nilsson collection. The questions and some of the answers are listed below.

26.1 Discussion on Amarel's Paper

What is needed to automate Amarel's architecture for finding a reasonable problem representation? We need to make explicit our theory of representation and also the assumptions we make about problem solving. This is because a shift to another formulation is done to gain space/time efficiency (usually) and this pre-supposes a model of computation. To understand the generation problem (generation of representations) we need to have a microtheory of representation. Amarel shows a very interesting example of representation shifts and hints at the sources of knowledge needed to automate this. Amarel does not explicitly deal with the issue of the relationship between the various formulations. It is an interesting exercise to prove that the formulations are equivalent modulo the goal.

26.2 Learning by Taking Advice

You need some kind of interpreter to take advice from the user and convert it into

specification languages to make advice-giving and interpretation easy, i.e we need a vocabulary to state and formulate advice. Next, we need a mechanism to make the advice operational. This is nontrivial. This operation is called compilation. An example from Mostow's game of Hearts strategy "learning" program was discussed.

Some discussion on meta-learning: or somehow learning about how to learn.

Question raised: is learning, as currently used in AI, a hack? Counter response: Rosenbloom has gained a lot of ground using a single learning strategy.

Some discussion on learning in very young children. Seems to be quite different from the "intellectual" learning that does on in adults: more oriented towards spatial/temporal/continuity relations in the world.

26.3 Discussion on McCarthy's papers

— What are the epistemological problems of AI?

See page 432 for the definition of the epistemological problem and page 460 for an inventory of these problems in AI. In particular note,

* Cooperative problem solving by independent agents. See Jeff Rosenschein's recent PhD thesis from Stanford.
* Acquisition of knowledge. How does McCarthy's perspective on this differ from those in Expert Systems (or does it?)?.
* Concurrent events and actions. What exactly are the problems here? How useful is the work that people in systems are doing wrt this problem?
* Spatial and temporal reasoning. What are the problems in doing this and what sorts of solutions have been proposed in literature?
* The vision problem. What has been accomplished and what are the limitations of current solutions. Would it be fair to say that Brooks' system incorporates the scheme that McCarthy suggest on page 461?
* Modal concepts like 'causality'. What is hard about representing and reasoning with these concepts (Hint: look at Moore's paper in the Webber/Nilsson collection)
* The frame problem and the qualification problem. What solution does McCarthy propose? (Hint : Circumscription. You need to have an intuitive

idea of what this is all about. The formalism presented in the article in the Webber/Nilsson collection has since been replaced by a better one (see a recent SAIL memo by McCarthy)).

- What is the distinction between metaphysical and epistemological adequacy, and of what value is the distinction? Is it related to the performance/competence distinction made by Chomsky?

  Metaphysical adequacy: facts about the world are represented in a form which is internally consistent; a consistent model of the world for facts that are of interest to the observer. Generally good for constructing theories about the world.

  Epistemological adequacy: facts are useful for expressing practical concerns about things and relations in the world. The facts satisfactorily explain things of interest to the observer.

  Question raised: Is McCarthy's paper philosophically adequate? (ie. does it explain anything of interest to the observer?)

  Discussion on 2 puzzles: the black ravens and the grue puzzle.

  The raven puzzle: the statement "All ravens are black" is logically equivalent to the statement "Any non-black object is not a raven." Hence any non-black object is effectively evidence in support of the statement "All ravens are black." But this seems counter-intuitive to most people.

  The grue puzzle: Let the color grue mean "green until time t sometime in the future, afterwards blue." Then any object which we now call green could really be grue. It would be more correct to say "green or grue."

- What is McCarthy's complaint with Modal Operators and Logics? How does he propose to do away with them?

  They make using the system much more unwieldy. Like higher order logics they make solving problems within them much more difficult, even if they are more descriptive.

  Digression to mini-tutorial into Modal Logic: Discussion of predicates for Necessity and Possibility ("square" and "diamond") operators. Discussion on possible worlds semantics.

- What is reification? What is the utility of this notion?

  See McCarthy's paper "First order logic propositions as individuals". This way

we can get the power of second order logic within a first order system.

- What does philosophy have to do with AI in McCarthy's opinion? How can research in philosophical issues feed into the building of AI theories?
- Why does McCarthy emphasize research on common sense reasoning?
- why have attempts to achieve AI by simulated evolution not succeeded?
- What is the relationship between MP and RP (page 433)? Why have an MP at all?

26.4 Discussion on Moore's paper

- What are the main ideas in this paper?
- What are the problems in reasoning with knowledge?
- How does reformulating the problem in possible worlds semantics solve some of the above problems?
- What problems in reasoning with actions that Moore not touch upon? Why?
- What are the disadvantages of this approach?
- Logic of Knowledge:
  1. Kp implies p property of knowledge
  2. Kp and K(p implies q) implies Kq closure
  3. Kp implies KKp positive introspection
  4. not Kp implies K(not Kp) negative introspection

  Persons doing work in logic of knowledge: Kurt Konolige, SRI
  Hector Levesque, Toronto(?)
  Joe Halpern, IBM San Jose

26.5 Doyle's TMS

- Why is belief revision important in AI?
- Difference between nonmonotonic reasoning and fuzzy reasoning?
- How does Doyle justify the overhead of recording justifications for beliefs?
- Give a short description of how the TMS works (with an example).

Chapter 27

Advanced Topics II

27.1 Outline of discussion

- **CS229c**
- **Assessment of course.**
- **Discussion and bank of Qual questions**

27.2 CS229c

The class members suggested that a sequel to this course be designed where discussion would be oriented solely for the qual and where the 'classics' reading list and the reading list for the 'advanced topics' would be covered in depth. The idea was voted down since there is quite a bit of bureaucratic machinery to move in order to create a new course. The standard practice was for students to sign up for CS390 credit (3 or 6) under their advisor and form their own study group for the AI qual. Typically, senior graduate students are invited to these sessions for help with issues and for giving mock quals.

27.3 What has the course generated?

Other than enthusiasm for research in AI among the participants(l), the following

- An extensive annotated reading list for the AI qual.
- Transcriptions of the discussion on the reading list (2 per week) edited and extended by the author.
- Transcriptions of the discussion sessions with invited guests (1 per week).
- A bank of qual questions.
- A list of sources of AI literature.
- A proposal for an automatic qual question generator.

27.4 Qual Question List

After a brief debate on whether we wanted to discuss some of the papers in Chapter 5 of the Webber-Nilsson collection or go through an informal qual, the class opted for the latter. The questions below were prepared by the author in 1984.(Thanks to Jock Mackinlay, Russ Greiner, Dave Smith, Prof. Genesereth). All the answers are not provided, but hints and pointers are given, where necessary.

- If you wrote an AI primer, how would you organize its contents?
  This is a hard question. The intent is to get at the student's understanding of AI as a whole (whether it is greater than the sum of its parts or not, at any rate whether it is greater than the sum of the 14 chapters in the handbook !). Identify core areas and applications with substantiating reasons for the split. Where does learning fit? Robotics? CAI? NL? What is the order in which these areas would be covered? Another way of phrasing this question is : How would you organize an AI course?
- Major areas in which AI has to make progress.
  The idea is to get at a sense of what you perceive to be the key open problems in AI and why? Extra points for answering with confidence and for tying up work in the different sub-fields of AI (this shows a comprehensive understanding !).
- What is AI?
  In itself, a bad question. But probable intent is to get at your orientation toward the field which might explain your biases in the above two questions.

- What is the thrust of expert system research?
  It is a bad idea to begin listing in a random way all the current work that is going on. A structured answer (top-down !) is what is expected. An answer to this question is in Prof. Buchanan's paper 'New research in expert systems' (HPP-81-1).
- What areas in AI are going to benefit from parallelism?
  Low level computations in signal to symbol transformation problems (i.e speech, edge-finding in vision), complex simulations (weather prediction), certain types of inference mechanisms (semantic net marker propagation [Fahlman]). The blackboard architecture might be particularly suited for investigation of the power of parallelism, because of its decomposability.
- What problems in AI will not be solved by parallelism?
  See the notes from the Zadeh article
- Who are the people working on parallel AI?
  See the reading list on advanced topics. There is a group at Stanford called PAI (Parallel AI) – contact Vineet Singh (VSINGH@SUMEX) if you want to get on their mailing list. They have a library with important publications in this field. They meet weekly to discuss recent work.
- Why is work in qualitative physics (a la deKleer) part of AI?
- What is relaxation?
  See the notes for the Vision week to get at the answer to this question.
- What is goal regression?
  Read the article "Achieving several goals simultaneously" by Richard Waldinger in the Webber-Nilsson collection.
- What is the procedural/declarative controversy?
  This famous (non)controversy is explained well in Terry Winograd's article "Frame representations and the procedural/declarative controversy". See also the notes from the guest session with Prof. Nilsson for more on this.
- What is the frame problem? The qualification problem?
  Crisp definitions are necessary. Both problems arise in the representation of actions. The frame problem arises in describing the effects of an action. The qualification problem arises in describing the pre-conditions of an action.

- **What solutions have been proposed for the frame problem?**
  Pat Hayes's paper " The Frame problem and related problems in Artificial Intelligence' in the Webber-Nilsson collection outlines in a unified framework all the major solutions proposed.

- **What work goes on at the Robotics Institute in CMU?**
  See the 'Research in Progress' articles on this institute handed out in class.

- **What is the cutest title for an AI publication?**
  "How to tell your computer to recognize speech - not wreck a nice beach" - by Janet Maclver Baker. This article was handed out to all the members of the class. Please keep this author posted on any competing titles !

- **What is chunking?**
  This is explained in the AAAI-84 paper by Rosenbloom entitled "Chunking as a general learning mechanism".

- **Who/what are the following which occur in AI folklore - Simon's ant and grandfather's axe?**
  Simon's ant occurs in the opening paragraph of one of Simon's most beautiful essays 'On the architecture of complexity'. The point of the example is that the behavior of an ant walking on a beach seems extremely complex, but the principle underlying its motion is (could be) extremely simple. Grandfather's axe is a famous example borrowed from philosophical literature. What characterizes this axe : what if the handle and the cutting edge were both replaced? Would it still qualify as grandfather's axe? Leads into the question of what constitutes 'being'. Thought question : Why should we worry about this in AI?

- **Do a resolution proof.**
  Conversion to CNF and application of the resolution rule.

- **Contrast Prolog with Lisp.**
  Refer to article which compares them . Bewarned that it is written by Prolog folks and might be biased.

- **What is fuzzy logic? What is it trying to capture? Where will it be useful for AI? See interview with Zadeh (the founder of Fuzzy logic) to get a feel for this issue.**

- What efforts are directed to improving man-machine communication? Machine-machine communication? Why do these issues come under the purview of AI?
- Why do we want computers to understand NL?
  This is taken from Bobrow's paper in SIP where he describes his STUDENT system. NL will improve the bandwidth of communication between computers and humans. It is easier to state problems in English than in any known computer language. Programming languages are typically process-oriented whereas English is a rich vehicle for description. First order logic is also a rich medium for description, but it is not well suited for the description of pictures etc. Also, given the Chomskian view that language is a complex cognitive ability, the understanding of NL will lead to better understanding of the human mind. Thus creating a NLU machine would constitute a big step toward the construction of an artificially intelligent agent. These goals enunciated in 1968 hold even today.
- What work has been done on learning by analogy?
  Starting from Thomas Evans work on geometrical analogies in the mid sixties (reported in SIP), to Kling's work at Stanford on using analogical reasoning in a theorem prover(in the early seventies), to Jaime Carbonell's work (reported in Machine Learning, 1983), to the Binford-Winston-Lowry-Katz effort on learning physical descriptions from functional descriptions, examples and precedents (reported in AAAI83), to Russ Greiner's work on analogies based on common abstractions(Stanford, 1984), to current work at Rutgers by Smadar Kedar-Cabelli on analogy with purpose in legal reasoning and Stuart Russell's work on a semantic characterization of analogical inference at Stanford.

Short answer questions

- Circumscription.
- The relation between circumscription and default reasoning.
- The B* algorithm.
- Admissibility of a search algorithm.
- Procedural nets.
- Generalized cones.

- ATN's and RTN's.
- Difference between best-first and A* algorithms.
- Difference between backward chaining and depth-first search.
- Represent the transitivity axiom in frames.
- Godel's incompleteness result.
- Godel's completeness theorem.
- Compare Winston's arch learning algorithm and Mitchell's VSA.
- Compare least commitment and dependency-directed backtracking.
- Features that help measure distance.
- Semantic nets – adv. and disadv.
- Epistemological issues in AI.
- What is R1?
- What is KRL?
- What is SOPHIE? What AI issues were addressed in that project?
- What are BUGGY and WEST?
- What is HARPY? What is the major lesson learnt from it?
- Which expert system is a geological prospector? How was it validated?
- Compare how MYCIN and PROSPECTOR handled uncertainty.
- What is referential opacity?
- What is possible world semantics?
- What system did Pat Suppes build?
- What are the special features of the BMTP?
- What are the main ideas in the Advice Taker paper?
- Compare natural deduction and resolution.
- What is truth maintenance.
- What is side-tracking?
- What is the role of the cut operator in PROLOG?
- Significance of the 1956 summer conference at Dartmouth.

- Transformational grammars - what are they?
- Compare the processing in the human visual cortex with the computation that is done in low level vision.
- Methods for representing visual scenes.
- Verification vision.
- Hans Moravec's contribution to the stereo problem.
- AI issues that can be explored in the context of robotics.
- Discrimination net.
- What did EPAM do?
- What are the main ideas in MARGIE?
- Importance of counterfactual conditionals to AI.
- Tarskian semantics.
- Who are the recipients of the CT award?
- For what kinds of problems is the BB architecture suited?
- Why is minimaxing analysis not enough for playing chess.
- Why do heuristic programs solve much harder problems than self organizing systems?
- Why did the early attempts at machine translation fail?
- How does Rl differ from MYCIN?
- How do the inference mechanisms in MYCIN and PROSPECTOR differ?
- How do the KR schemes in MYCIN and PROSPECTOR differ?
- What are the advantages of abstracting control knowledge from rule bases?
- What are the uses of meta-knowledge? name some systems that use meta-knowledge.
- What is the 'knowledge-cliff' problem?
- Mention some design criteria for the building of consultation systems. Does VM meet these criteria?
- What is default reasoning? Give examples. How is it handled in today's systems?

- Why is representation a key issue in AI?
- What is AP? What is your description of an ideal AP system?
- What are the main ideas in Programmer's Apprentice?
- What is a constraint? Name a few systems that use constraints. When are constraints appropriate to use? When are they inappropriate?
- How is conflict resolution handled in rule based systems?
- Name five problems that the tasks of speech understanding and image understanding share with each other.
- What are the advantages of a production system architecture.
- What kinds of inferences can be done in a frame based system.
- What are the advantages, if any, of a frame based system, over a pure first order logic system.
- How is causality represented in AI systems? Give examples.
- How is time represented and reasoned with in AI systems?
- What is the closed world assumption.
- Compare Wilkins' chess planner PARADISE to a robot planner like STRIPS.
- Explain Cordell Green's answer extraction method.
- Why does EURISKO have a simple control structure? Would a hierarchical control structure like MOLGEN's extend EURISKO's abilities?
- How is a skeletal plan chosen in Friedland's MOLGEN?
- What the current problems in NL generation?
- What methods in AI are used for limiting search?
- Name three systems that use FC, BC, means-end analysis, constraint prop.
- When is FC superior to BC?
- What is NM logic?
- Difference between modal logic and meta-level.
- Difference between second order and meta-level.
- What representational schemes have been devised to handle the meaning of English sentences.

- Why is NLU an AI problem?
- Limitations of a script based theory of understanding.
- What do the experiments on AM and EURISKO tell us about the nature of heuristics.
- What are the main contributions of HACKER to AI research.
- In what domains are Expert systems appropriate?
- How are expert systems evaluated?
- What the design issues in the building of expert systems.
- What features make a programming language an AI programming language?
- What is the Turing test?
- What are the objectives of machine learning research?
- What are the basic approaches to machine learning?
- What is credit assignment and what are the methods proposed to handle it?
- What is analytical learning?
- What are the limitations of the VSA?
- Main ideas in Mostow's FOO.