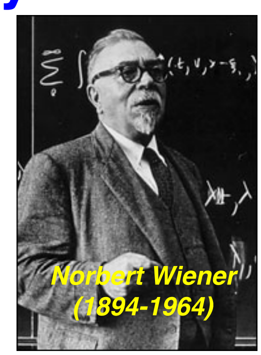
Communication, Information, and Machine Learning

Robert Stengel
Robotics and Intelligent Systems MAE 345
Princeton University, 2015

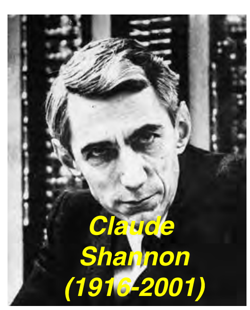
- Communication/Information Theory
  - Wiener vs. Shannon
- Finding Decision Rules in Data
- Markov Decision Processes
- Graph and Tree Search
- Expert Systems

"Communication Theory" or "Information Theory"?



- Prodigy at Harvard, professor at MIT
  - Cybernetics
  - Feedback control
  - Communication theory

Dark Hero Of The Information Age: In Search of Norbert Wiener, the Father of Cybernetics, Flo Conway and Jim Siegelman, 2005. Basic Books

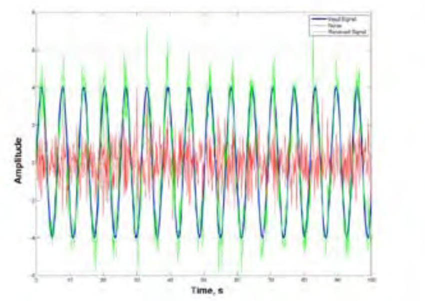


- University of Michigan, MIT (student), Bell Labs, MIT (professor)
  - Boolean algebra
  - Cryptography, telecommunications
  - Information theory

The Information: A History, A Theory, A Flood, James Gleick, 2011, Pantheon.

Communication: Separating Signals from Noise

Signal-to-Noise Ratio, *SNR*



$$SNR = \frac{Signal\ Power}{Noise\ Power} \triangleq \frac{S}{N} = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} (zero\text{-}mean),\ \text{e.g.,}\ \frac{watts}{watts}$$

SNR often expressed in decibels

$$SNR(dB) = 10\log_{10}\frac{Signal\ Power}{Noise\ Power} = 20\log_{10}\frac{Signal\ Amplitude}{Noise\ Amplitude} = S(dB) - N(dB)$$

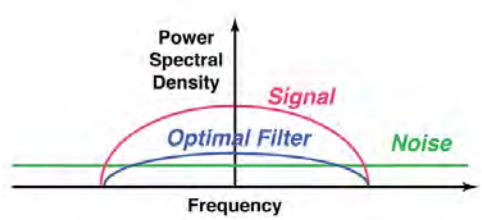
Communication: Separating Analog Signals from Noise

Signal-to-Noise Spectral Density Ratio, *SDR(f)*

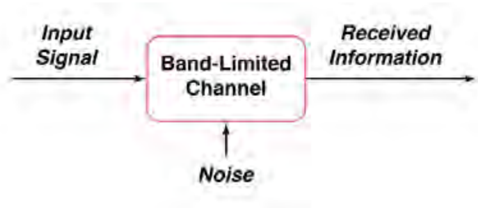
$$SDR\left(\frac{\omega}{2\pi}\right) = SDR(f) = \frac{Signal\ Power\ Spectral\ Density(f)}{Noise\ Power\ Spectral\ Density(f)} \triangleq \frac{PSD_{signal}(f)}{PSD_{noise}(f)}$$

Optimal (non-causal) Wiener Filter, *H(f)*

$$H(f) = \frac{PSD_{signal}(f)}{PSD_{signal}(f) + PSD_{noise}(f)} = \frac{SDR(f)}{SDR(f)+1}$$



Communication: Bit Rate Capacity of a Noisy Channel



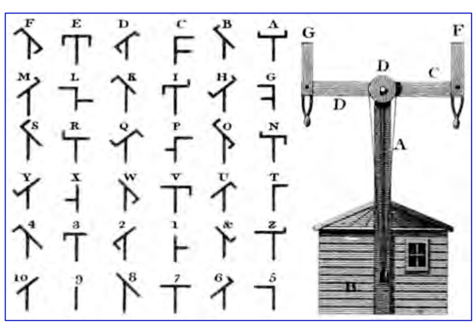
Shannon-Hartley Theorem, *C* bits/s

$$\boxed{C = B\log_2\left(\frac{S}{N}+1\right) = B\log_2\left(SNR+1\right)}$$

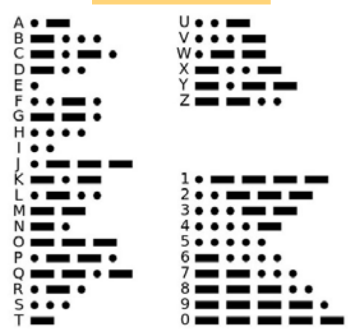
$S =$ Signal Power, e.g., watts
$N =$ Noise Power, e.g., watts
$B =$ Channel Bandwidth, Hz
$C =$ Channel Capacity, bits/s

Early Codes: How Many Bits?

Semaphore Line Code



Morse Code



- ~ (10 x 10) image = 100 pixels = 100 bits required to discern a character

ASCII encodes 128 characters in 7 bits (2 bytes – 1)

8th bit? | Parity check

- Dot = 1 bit
- Dash = 3 bits
- Dot-dash space = 1 bit
- Letter space = 2 bits
- 3 to 21 bits per character

Entropy Measures Information Content of a Signal

- ***H*** = Entropy of a signal encoding ***I*** distinct events

$$H = -\sum_{i=1}^{I} \Pr(i)\log_2 \Pr(i)$$

$$0 \le \Pr(.) \le 1$$
$$\log_2 \Pr(.) \le 0$$
$$0 \le H \le 1$$

- **Entropy is a measure of the signal's uncertainty**
  - ***High entropy connotes high uncertainty***
  - ***Low entropy portrays high information content***
- ***i*** **= Index identifying an event encoded by a signal**
- **Pr(*i*) = Probability of *i*th event**
- **$\log_2$Pr(*i*) = Number of bits required to characterize the probability that the *i*th event occurs**

Entropy of Two Events with Various Frequencies of Occurrence

- **–Pr(*i*) $\log_2$Pr(*i*) represents the channel capacity (i.e., average number of bits) required to portray the *i*th event**
- **Frequencies of occurrence estimate probabilities of each event (#1 and #2)**

$$\Pr(\#1) = \frac{n(\#1)}{N}$$
$$\Pr(\#2) = \frac{n(\#2)}{N} = 1 - \frac{n(\#1)}{N}$$

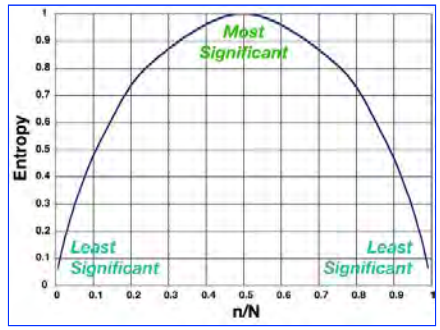
$$\log_2 \Pr(\#1 \text{ or } \#2) \le 0$$

- **Combined entropy**

$$H = H_{\#1} + H_{\#2}$$
$$= -\Pr(\#1)\log_2 \Pr(\#1) - \Pr(\#2)\log_2 \Pr(\#2)$$

Entropy of Two Events with Various Frequencies of Occurrence

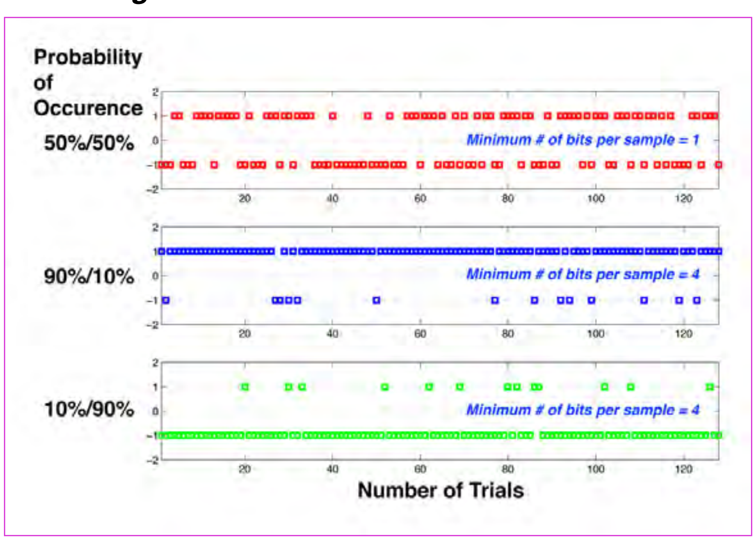
	Entropies for 128 Trials				
	Pr(#1)	- # of Bits(#1)	Pr(#2)	- # of Bits(#2)	Entropy
n	n/N	$\log_2(n/N)$	1 - n/N	$\log_2(1 - n/N)$	*H*
1	0.008	-7	0.992	-0.011	0.066
2	0.016	-6	0.984	-0.023	0.116
4	0.031	-5	0.969	-0.046	0.201
8	0.063	-4	0.938	-0.093	0.337
16	0.125	-3	0.875	-0.193	0.544
32	0.25	-2	0.75	-0.415	0.811
64	0.50	-1	0.50	-1	1
96	0.75	-0.415	0.25	-2	0.811
112	0.875	-0.193	0.125	-3	0.544
120	0.938	-0.093	0.063	-4	0.337
124	0.969	-0.046	0.031	-5	0.201
126	0.984	-0.023	0.016	-6	0.116
127	0.992	-0.011	0.008	-7	0.066



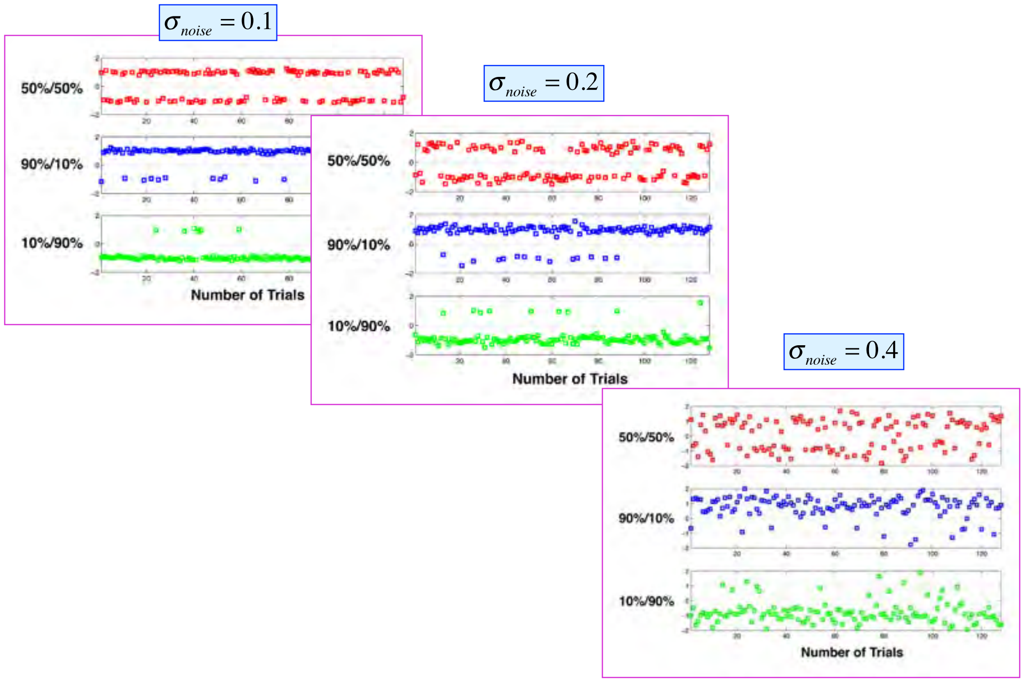
Entropy of a fair coin flip = 1

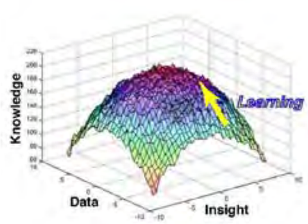
Accurate Detection of Events Depends on Their Probability of Occurence

Signals Rounded to Their Intended Values



Accurate Detection of Events Depends on Their Probability of Occurrence





Finding Efficient Decision Rules in Data

(Off-Line)

- **Choose most important attributes first**
- **Recognize when no result can be deduced**
- **Exclude irrelevant factors**

- **Iterative Dichotomizer*: the ID3 Algorithm**
  - **Build an efficient decision tree from a fixed set of examples** ***(supervised learning)***

***Dichotomy: Division into two (usually contradictory) parts or opinions**

Fuzzy Ball-Game Training Set

	Attributes				Decisions
Case #	**Forecast**	**Temperature**	**Humidity**	**Wind**	**Play Ball?**
1	Sunny	Hot	High	Weak	**No**
2	Sunny	Hot	High	Strong	**No**
3	Overcast	Hot	High	Weak	**Yes**
4	Rain	Mild	High	Weak	**Yes**
5	Rain	Cool	Low	Weak	**Yes**
6	Rain	Cool	Low	Strong	**No**
7	Overcast	Cool	Low	Strong	**Yes**
8	Sunny	Mild	High	Weak	**No**
9	Sunny	Cool	Low	Weak	**Yes**
10	Rain	Mild	Low	Weak	**Yes**
11	Sunny	Mild	Low	Strong	**Yes**
12	Overcast	Mild	High	Strong	**Yes**
13	Overcast	Hot	Low	Weak	**Yes**
14	Rain	Mild	High	Strong	**No**

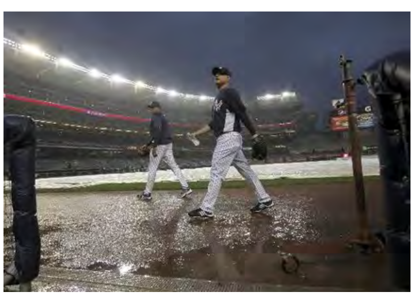
Parameters of the ID3 Algorithm

- **Decisions, e.g., Play ball or don't play ball**
  - ***D*** **= Number of possible decisions**
    - **Decision:** **Yes, no**

Parameters of the ID3 Algorithm

Doug Pensinger, Getty Images

- **Attributes, e.g., Temperature, humidity, wind, weather forecast**
  - ***M*** **= Number of attributes to be considered in making a decision**
  - **I_m = Number of values that the i^{th} attribute can take**
    - **Temperature:** Hot, mild, cool
    - **Humidity:** High, low
    - **Wind:** Strong, weak
    - **Forecast:** Sunny, overcast, rain



Parameters of the ID3 Algorithm

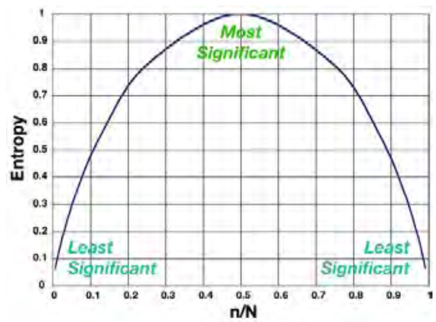
- **Training trials, e.g., all the games attempted last month**
  - ***N*** **= Number of training trials**
  - ***n(i)*** **= Number of examples with i^{th} attribute**

Best Decision is Related to Entropy and the Probability of Occurrence

- High entropy
  - Signal provides low coding precision of distinct events
  - Differences coded with few bits

$$H = -\sum_{i=1}^{I} \Pr(i)\log_2 \Pr(i)$$

- Low entropy
  - More complex signal structure
  - Detecting differences requires many bits



- Best classification of events when $H = 1$...
  - but that may not be achievable

Case #	Forecast	Temperature	Humidity	Wind	Play Ball?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Low	Weak	Yes
6	Rain	Cool	Low	Strong	No
7	Overcast	Cool	Low	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Low	Weak	Yes
10	Rain	Mild	Low	Weak	Yes
11	Sunny	Mild	Low	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Low	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision-Making Parameters for ID3

H_D = Entropy of all possible decisions

$$H_D = -\sum_{d=1}^{D} \Pr(d)\log_2 \Pr(d)$$

G_i = Information "gain" of i^{th} attribute

$$G_i = S_D + \sum_{i=1}^{I_m} \Pr(i) \sum_{d=1}^{D} \left[\Pr(i_d)\log_2 \Pr(i_d)\right]$$

$\Pr(i_d) = n(i_d)/N(d)$: Probability that i^{th} attribute depends on d^{th} decision

$\sum_{i=1}^{I_m} \Pr(i) \sum_{d=1}^{D} \left[\Pr(i_d)\log_2 \Pr(i_d)\right]$: **Mutual information** of i and d

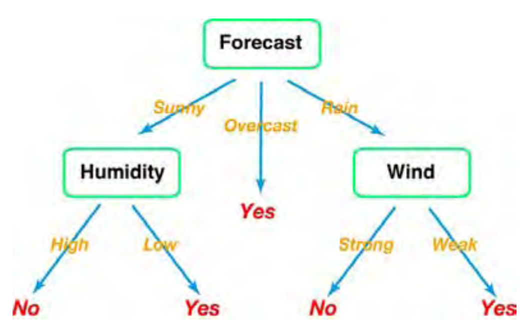
Decision Tree Produced by ID3 Algorithm

- Root Attribute gains, G_i
  - Forecast: 0.246
  - Temperature: 0.029
  - Humidity: 0.151
  - Wind: 0.048

- Therefore
  - Choose *Forecast* as root
  - Ignore *Temperature*
  - Choose *Humidity* and *Wind* as branches

Decision Tree Produced by ID3 Algorithm

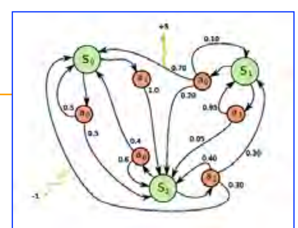
- Evaluating remaining gains,
  - *Sunny* branches to *Humidity*
  - *Overcast = Yes*
  - *Rain* branches to *Wind*



Markov Processes

Markov Decision Process

- **Model for decision making under uncertainty contains following elements**



$$\text{where} \qquad \boxed{\left[\mathbf{X},\mathbf{A},P_{a_m}\left(\mathbf{x}_i,\mathbf{x}'\right),L_{a_m}\left(\mathbf{x}_i,\mathbf{x}'\right)\right]}$$

$\mathbf{X}$: **Finite set of states**, $\mathbf{x}_1,\ \mathbf{x}_2,\ldots\mathbf{x}_i,\ldots,\mathbf{x}_I$

$\mathbf{A}$: **Finite set of actions**, $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_j,\ldots,\mathbf{a}_J$

$$P_{\mathbf{a}_j}\left(\mathbf{x}_k,\mathbf{x}'\right)=\Pr\left\{\left[\mathbf{x}\left(t_{k+1}\right)=\mathbf{x}'\right]\middle|\left[\mathbf{x}\left(t_k\right)=\mathbf{x}_k \text{ and } \mathbf{a}\left(t_k\right)=\mathbf{a}_j\right]\right\}$$

$$= \text{Probability that } \mathbf{a}_j \text{ will cause } \mathbf{x}_i\left(t_k\right) \text{ to transition to } \mathbf{x}'$$

$$L_{\mathbf{a}_j}\left(\mathbf{x}_k,\mathbf{x}'\right)= \textbf{Expected immediate reward} \text{ for transition from } \mathbf{x}_k \text{ to } \mathbf{x}'$$

- **Optimal decision maximizes (minimizes) <u>expected total reward (cost)</u> by choosing best set of actions (control policy)**
  - **Linear-quadratic-Gaussian (LQG) control**
  - **Dynamic programming -> HJB equation ~> A* search**
  - **Reinforcement learning ~> Heuristic search**

Maximizing the Utility Function of a Markov Process

Utility function: $J = \lim_{k_f \to \infty} \sum_{k=0}^{k_f} \gamma(t_k) L_{\mathbf{a}}\left[\mathbf{x}(t_k), \mathbf{x}(t_{k+1})\right]$

$\gamma(t_k)$: **Discount rate**, $0 < \gamma(t_k) < 1$

Utility function to go = Value function:

$$V = \lim_{k_f \to \infty} \sum_{k=k_{current}}^{k_f} \gamma(t_k) L_{\mathbf{a}}\left[\mathbf{x}(t_k), \mathbf{x}\left(t_{k+1}\right)\right]$$

Maximizing the Utility Function of a Markov Process

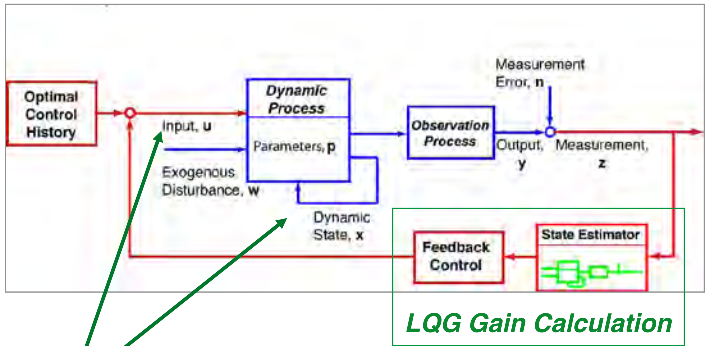
Optimal control at *t*

$$\mathbf{u}_{opt}\left(t_k\right) = \arg\max_{\mathbf{a}} \left\{ L_{\mathbf{a}}\left[\mathbf{x}(t_k), \mathbf{x}(t_{k+1})\right] + \gamma(t_k) \sum_{k=k_{current}}^{\infty} P_{\mathbf{a}}\left[\mathbf{x}(t_k), \mathbf{x}(t_{k+1}))\right] V\left[\mathbf{x}(t_{k+1})\right] \right\}$$

Optimized value function

$$V^*\left(t_k\right) = L_{\mathbf{u}_{opt}(t_k)}\left[\mathbf{x}^*(t_k)\right] + \gamma(t_k) \sum_{k=k_{current}}^{\infty} P_{\mathbf{u}_{opt}(t_k)}\left[\mathbf{x}^*(t_k), \mathbf{x}_{est}^*(t_{k+1})\right] V\left[\mathbf{x}^*_{est}(t_{k+1})\right]$$

LQG Control Optimizes Discrete-Time LTI Markov Process



$$\left[\mathbf{X},\mathbf{A},P_{a_m}\left(\mathbf{x}_i,\mathbf{x}'\right),L_{a_m}\left(\mathbf{x}_i,\mathbf{x}'\right)\right]$$

where

$\mathbf{X}$: **Finite set of states**, $\mathbf{x}_1,\ \mathbf{x}_2,\ldots\mathbf{x}_i,\ldots,\mathbf{x}_I$

$\mathbf{A}$: **Finite set of actions**, $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_j,\ldots,\mathbf{a}_J$

$$P_{\mathbf{a}_j}\left(\mathbf{x}_k,\mathbf{x}'\right)=\Pr\left\{\left[\mathbf{x}\left(t_{k+1}\right)=\mathbf{x}'\right]\left[\mathbf{x}\left(t_k\right)=\mathbf{x}_k \text{ and } \mathbf{a}\left(t_k\right)=\mathbf{a}_j\right]\right\}$$

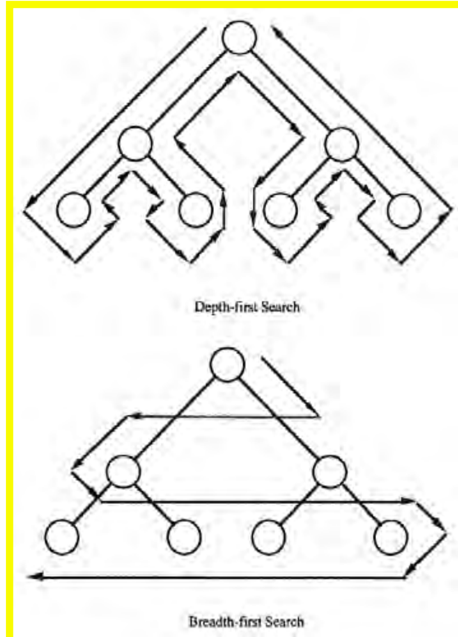
$=$ Probability that $\mathbf{a}_j$ will cause $\mathbf{x}_i\left(t_k\right)$ to transition to $\mathbf{x}'$

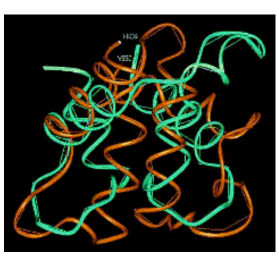
$L_{\mathbf{a}_j}\left(\mathbf{x}_k,\mathbf{x}'\right)=$ **Expected immediate reward** for transition from $\mathbf{x}_k$ to $\mathbf{x}'$

Graph and Tree Search

Search

- Typical AI textbook problems
  - Prove a theorem
  - Solve a puzzle *(e.g., Tower of Hanoi)*
  - Find a sequence of moves to win a game *(e.g., chess)*
  - Find the shortest path between points *(e.g., Traveling salesman problem)*
  - Find a sequence of symbolic transformations that solve a problem *(e.g., Mathematica)*
- The common thread: search
  - Structures for search
  - Strategies for search



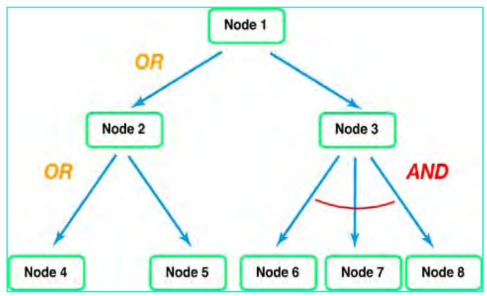


Curse of Dimensionality

- Feasible search paths may grow without bound
  - Possible combinatorial explosion
  - Checkers: 5 x 10^{20} possible moves
  - Chess: 10^{120} moves
  - Protein folding: ?
- Limiting search complexity
  - Redefine search space
  - Employ heuristic (i.e., pragmatic) rules
  - Establish restricted search range
  - Invoke decision models that have worked in the past

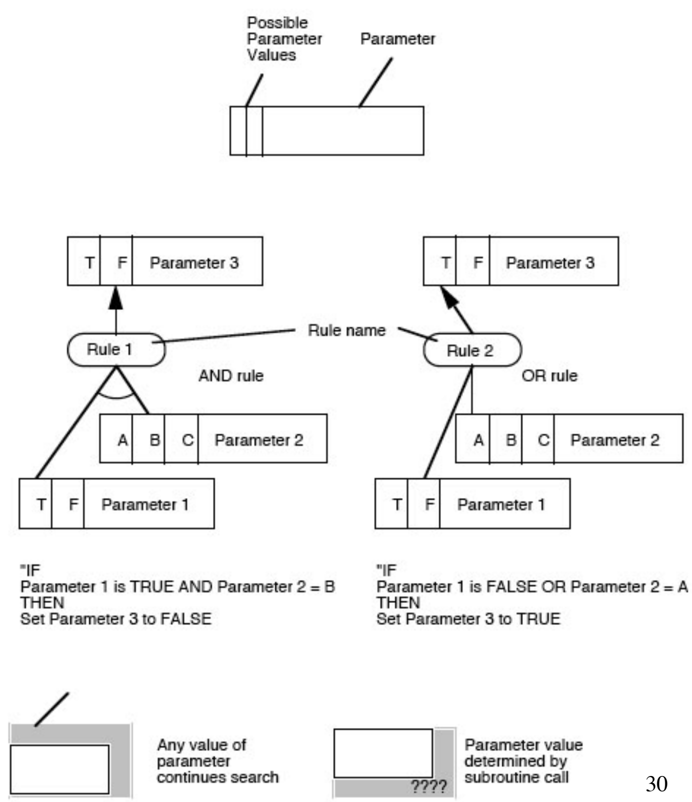
Structures for Search

- Trees
  - Single path between root and any node
  - Path between adjacent nodes = arc
  - Root node
    - no precursors
  - Leaf node
    - no successors
    - possible terminator



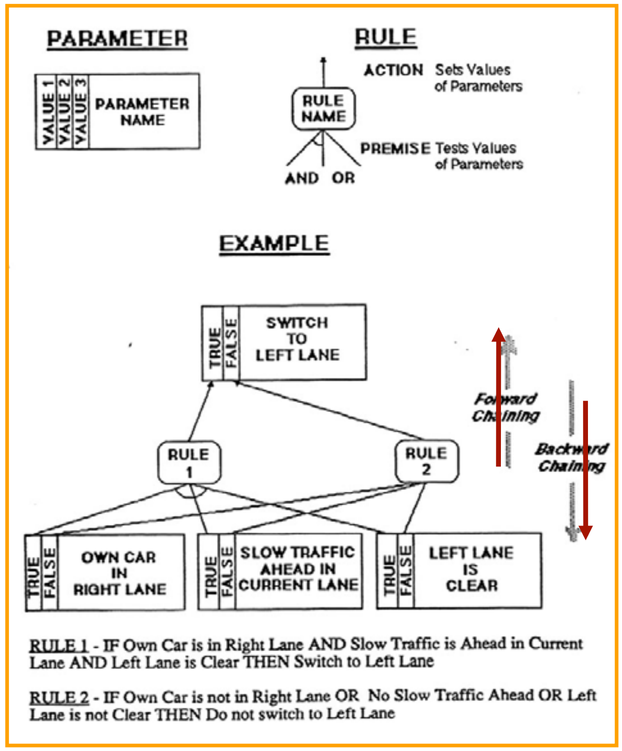
Expert System Symbology

- Parameters
  - Values
- Rules
  - Name
  - Logic
- And/Or



Structures for Search

- **Graphs**
  - **Multiple paths between root and some nodes**
  - **Trees are subsets of graphs**



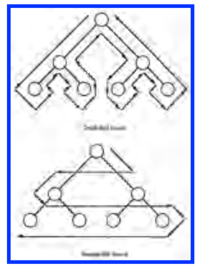
31

Directions of Search

- **Forward chaining**
  - **Reason from premises to actions**
  - **Data-driven: draw conclusions from facts**
- **Backward chaining**
  - **Reason from actions to premises**
  - **Goal-driven: find facts that support hypotheses**

Strategies for Search

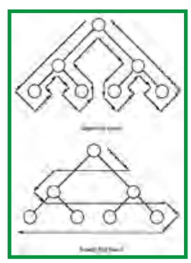
- Realistic assessment
  - Not necessary to consider all 10^{120} possible moves to play good chess
  - Forward and backward chaining, but not 10^{120} evaluations



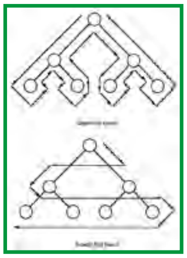
- Search categories
  - Blind search
  - Heuristic search
  - Probabilistic search
  - Optimization

• Search forward from opening? • Search backward from end game? • Both?

Blind Search



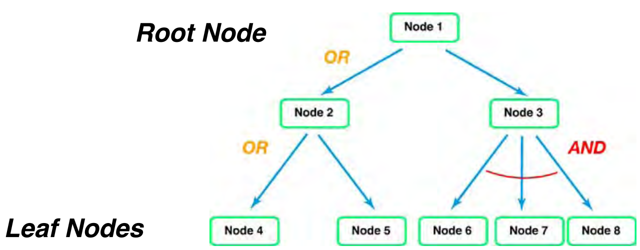
- Node expansion
  - Find all successors to that node
- Depth-first forward search
  - Expand nodes descended from most recently expanded node
  - Consider other paths only after reaching node with no successors
- Breadth-first forward search
  - Expand nodes in order of proximity to start node
  - Consider all sequences of arc number n (from root node) before considering any of number $(n + 1)$
  - Exhaustive, but guaranteed to find the shortest path to a terminator



Blind Search

- **Bidirectional search**
  - Search forward from root node and backward from one or more leaf nodes
  - Terminate when search nodes coincide
- **Minimal-cost forward search**
  - Each arc is assigned a cost
  - Expand nodes in order of minimum cost

AND/OR Graph Search

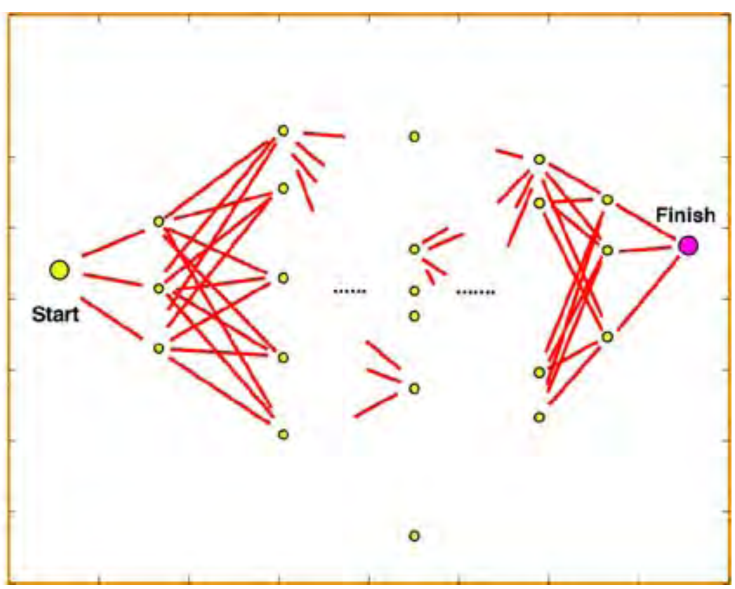


- **A node is "solved" if**
  - It is a leaf node with a satisfactory goal state
  - It has solved AND nodes as successors
  - It has OR nodes as successors, at least one of which is solved.
- **Goal: Solve the root node**

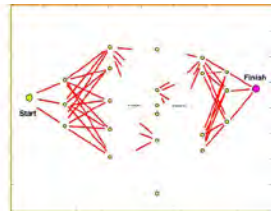
Heuristic Search

- For large problems, blind search typically leads to combinatorial explosion
- Employ heuristic knowledge about quality of possible paths
  - Decide which node to expand next
  - Discard (or *prune*) nodes that are unlikely to be fruitful
- Search for feasible (approximately optimal) rather than optimal solutions
- Ordered or best-first search
  - Always expand "most promising" node

Heuristic Optimal Search



Heuristic Dynamic Programming: A* Search



$$\hat{J}_{k_f} = \sum_{i=1}^{k} J_i + \sum_{i=k+1}^{k_f} \hat{J}_i(arc_i)$$

- Each arc bears an incremental cost
- Cost, *J*, estimated at k^{th} instant =
  - Cost accrued to *k*
  - Remaining cost to reach final point, k_f
- Goal: minimize estimated cost by choice of remaining arcs
- Choose arc_{k+1}, arc_{k+2} accordingly
- Use heuristics to estimate remaining cost

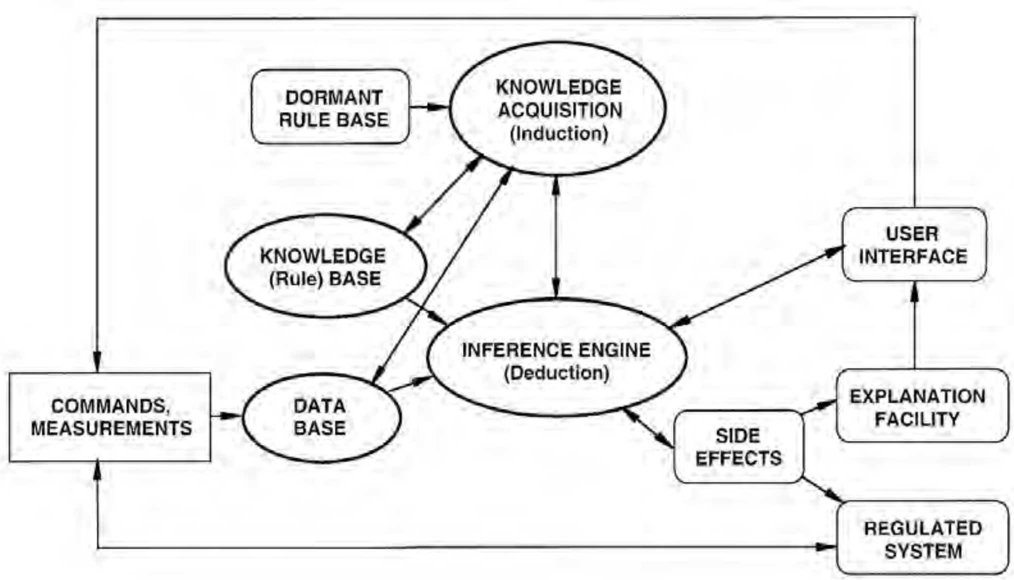
Expert Systems: Using Signals to Make Decisions

- Program that exhibits intelligent behavior
- Program that uses rules to evaluate information
- Program meant to emulate an expert or group of experts making decisions in a specific domain of knowledge (or *universe of discourse*)
- Program that chains algorithms to derive conclusions from evidence

Functions of Expert Systems

- Design
  - Conceive the form and substance of a new device, object, system, or procedure
- Diagnosis
  - Determine the nature or cause of an observed condition
- Instruction
  - Impart knowledge or skill
- Interpretation
  - Explain or analyze observations
- Monitoring
  - Observe a process, compare actual with expected observations, and indicate system status
- Negotiation
  - Propose, assess, and prioritize agreements between parties
- Planning
  - Devise actions to achieve goals
- Prediction
  - Reason about time, forecast the future
- Reconfiguration
  - Alter system structure to maintain or improve performance
- Regulation
  - Respond to commands and adjust control parameters to maintain stability and performance

Principal Elements of a Rule-Based Expert System



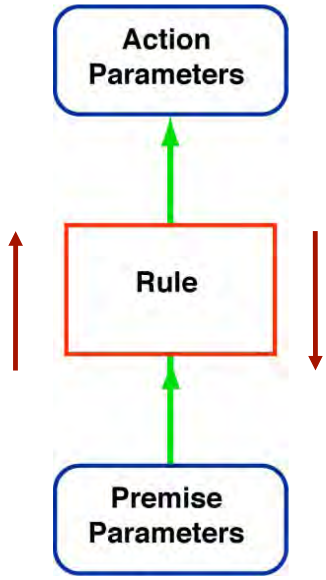
Critical Issues for Expert System Development

- System architecture
- Inference or reasoning method *(Deduction)*
- Knowledge acquisition *(Induction)*
- Explanation *(Abduction)*
- User interface

Representation of Knowledge for Inference

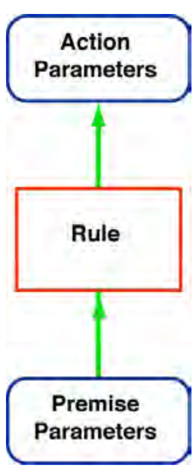
- Logic
  - Predicate calculus, 1st-order logic
  - Fuzzy logic, Bayesian belief network, …
- Search
  - Given one state, examine all possible alternative states
- Procedures
  - Function-specific routines executed within a rigid structure (e.g., flow chart)
- Semantic (propositional) networks
  - Model of associative memory
  - Tree or graph structure
  - Nodes: objects, concepts, and events
  - Links: interrelations between nodes
- Production (rule-based) systems
  - Rules
  - Data
  - Inference engine

Basic Rule Structure



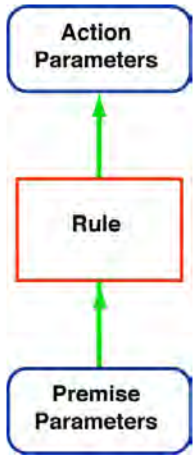
- Rule sets values of action parameters
- Rule tests values of premise parameters
- Forward chaining
  - Reasoning from premises to actions
  - Data-driven: facts to conclusions
- Backward chaining
  - Reasoning from actions to premises
  - Goal-driven: find facts that support a hypothesis
  - Analogous to numerical inversion

Elements of a Parameter



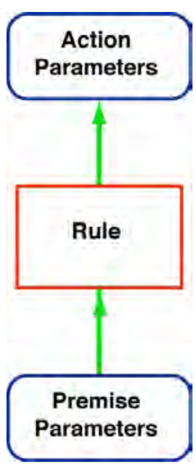
- Type
- Name
- Current value
- Rules that test the parameter
- Rules that set the parameter
- Allowable values of the parameter
- Description of parameter (for explanation)

Elements of a Rule

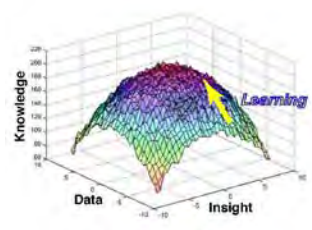


- Type
- Name
- Status
  - 0: Has not been tested
  - 1: Being tested
  - T: Premise is true
  - F: Premise is false
  - U: Premise is unknown
- Parameters tested by rule
- Parameters set by rule
- Premise: Logical statement of proposition or predicates
- Action: Logical consequence of premise being true
- Description of premise and action (for explanation)

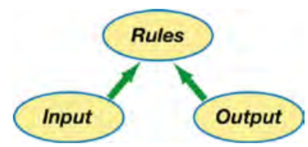
The Basic Rule: IF-THEN-ELSE



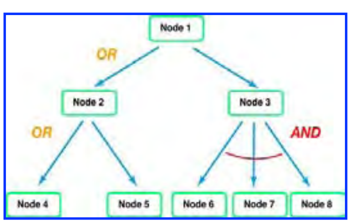
- If A = TRUE, then B, else C
- Material equivalence of propositional calculus, extended to predicate calculus and 1st-order logic, i.e., applied to logical statements
- Methods of inference lead to plans of action
- Compound rule: Logic embedded in The Basic Rule, e.g.,
  - Rule 1: If (A = B and C = D), then perform action E, else ....
  - Rule 2: If (A ≠ B or C = D), then E = F, else ....
- Nested (pre-formed compound) rule: Rule embedded in The Basic Rule, e.g.,
  - Rule 3: If (A = B), then [If (C = D), then E = F, else ...], else ....



Finding Decision Rules in Data



- Identification of key attributes and outcomes
- Taxonomies developed by experts
- First principles of science and mathematics
- Trial and error
- Probability theory and fuzzy logic
- Simulation and empirical results



Example of On-Line Code Modification

- Execute a decision tree
  - Get wrong answer
- Add logic to distinguish between right and wrong cases
  - If Comfort Zone = Water,
    - then Animal = Hippo,
    - else Animal = Rhino
  - True, but Animal is Dinosaur, not Hippo
  - Ask user for right answer
  - Ask user for a rule that distinguishes between right and wrong answer: If Animal is extinct, …

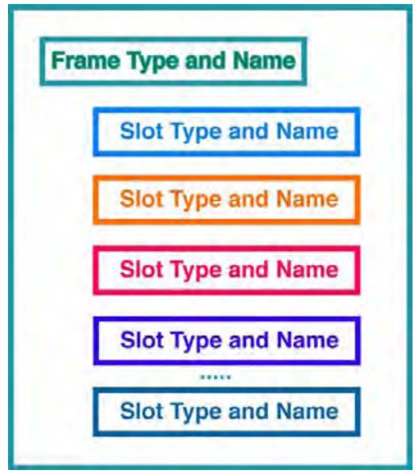
Decision Rules

Representation of Data

- **Set**
  - Crisp sets
  - Fuzzy sets
- **Schema**
  - Diagrammatic representation
  - A pattern that represents elements (or objects), their attributes (or properties), and relationships between different elements
- **Frame**
  - Hierarchical data structure, with inheritance
  - Slots: Function-specific cells for data
  - Scripts: frame-like structures that represent a sequence of events
- **Database**
  - Spreadsheets/tables/graphs
  - Linked spreadsheets

Structure of a Frame (or Object)

- Structure array in MATLAB
- Structure or property list in LISP
- Object in C++
- Ordered set of computer words that characterize a parameter or rule
- An archetype or prototype
- Object-oriented programming: Express Rules and Parameters as Frames

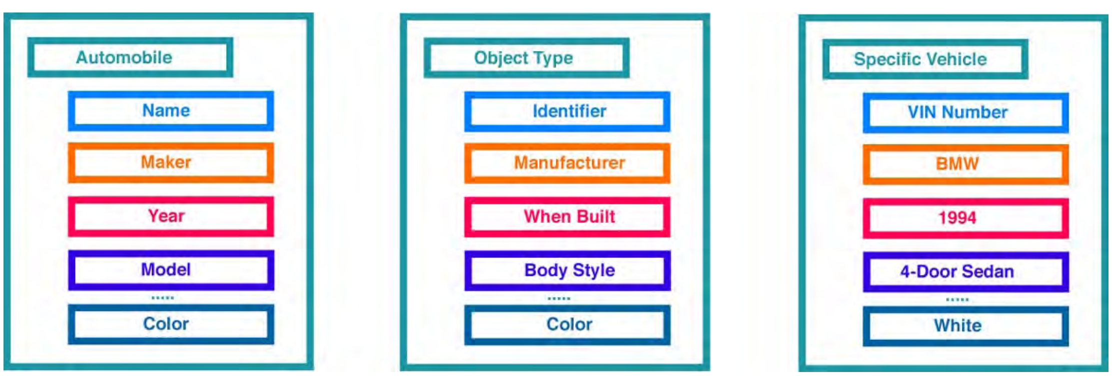


Example, Fillers, and Instance of a Frame

Application-Specific Frame

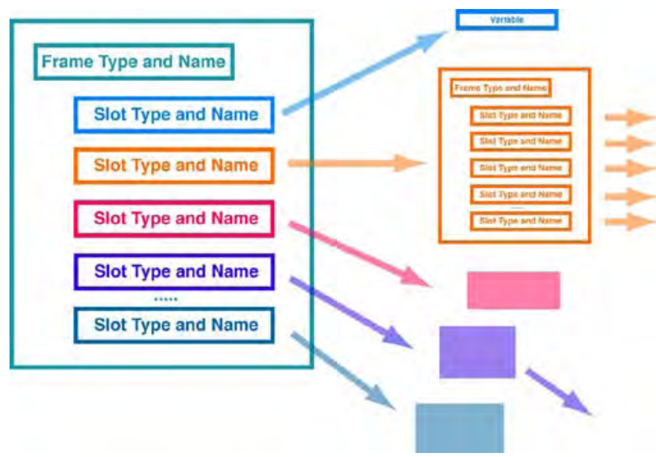
Generic Fillers

Instantiation



Inheritance and Hierarchy of Frame Attributes

- Legal fillers: Can be specified by
  - Data type
  - Function
  - Range
- Inheritance property
  - All instances of a specific frame may share certain properties or classes of properties
- Hierarchical property
  - Frames of frames may be legal
- Inference engine
  - Decodes frames
  - Establishes inheritance and hierarchy
  - Executes logical statements



Animal Decision Tree:
Forward Chaining

- What animal is it?

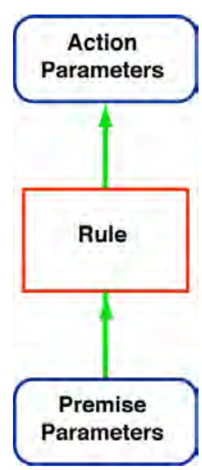
```
Premise  Parameter:  Size
Rule 1:       If  'Small',   test  'Sound'
               Else,   test   'Neck'
Action Parameter: None

Premise  Parameter:  Sound
Rule 2:       If  'Squeak',  Animal  =  Mouse
[END]
          Else, Animal   =   Squirrel     [END]
Action Parameter: Animal

Premise  Parameter:  Neck
Rule 3:      If  'Long',  Animal  =  Giraffe
[END]
               Else,   test   'Trunk'
Action Parameter: Animal

Premise  Parameter:  Trunk
Rule 4:      If  'True',  Animal  =  Elephant
[END]
              Else,   test   'Comfort   Zone'
Action Parameter: Animal

Premise  Parameter:  Comfort  Zone
Rule 5:       If  'Water',  Animal  =  Hippo
[END]
          Else,  Animal   =  Rhino      [END]
Action Parameter: Animal
```



Animal Decision Tree: Backward Chaining

- What are an animal's attributes?

```
Animal = Hippo
  From  Rule  5,  Comfort  Zone  =  Water
  From  Rule  4,  Trunk  =  False
  From  Rule  3,  Neck  =  Short
  From  Rule  1,  Size  =  Large
```


Animal Decision Tree: Parameters

```
Type:        Object  Attribute
Name:        Animal
Current  Value:  Variable
Rules  that  Test:  None
Rules  that  Set:  2,  3,  4,  5
Allowable Values: Mouse, Squirrel, Giraffe, Elephant,
Hippo, Rhino
Description:    Type  of  Animal

Type:        Object  Attribute
Name:        Size
Current  Value:  Variable
Rules  that  Test:  1
Rules  that  Set:  None
Allowable Values: Large, Small
Description:    Size  of  Animal

Type:        Object  Attribute
Name:        Sound
Current  Value:  Variable
Rules  that  Test:  2
Rules  that  Set:  None
Allowable Values: Squeak, No Squeak
Description:    Sound  made  by  Animal
```

```
Type:        Object  Attribute
Name:        Neck
Current  Value:  Variable
Rules that Test: 3
Rules  that  Set:  None
Allowable Values: Long, Short
Description:    Neck  of  Animal

Type:        Object  Attribute
Name:        Trunk
Current  Value:  Variable
Rules that Test: 4
Rules  that  Set:  None
Allowable Values: True, False
Description:    Snout  of  Animal

Type:        Object  Attribute
Name:        Comfort  Zone
Current  Value:  Variable
Rules that Test: 5
Rules  that  Set:  None
Allowable Values: Water, Dry Land
Description:    Habitat  of  Animal
```

Animal Decision Tree: Rules

```
Type:        If-Then-Else
Name:        Rule   1
Status:      Variable  (e.g.,  untested,  being
tested,
      tested  and  premise  =  T/F/unknown)
Parameters  Tested:  Size
Parameters  Set:  None
Premise:    Size  =  Large  or  Small
Action:       Test  'Sound'  OR  Test  'Neck'
Description:    Depending  on  value  of  'Size',
       test   'Sound'   or   'Neck'

Type:        If-Then-Else
Name:        Rule   2
Status:           Variable
Parameters  Tested:  Sound
Parameters  Set:  Animal
Premise:    Size  =  Large  or  Small
Action:      Set  value  of  'Animal'  AND  END
Description:    Depending  on  value  of  'Sound',
      identify  'Animal'  as  'Mouse'  or
'Squirrel'

Type:        If-Then-Else
Name:        Rule   3
Status:           Variable
Parameters  Tested:  Neck
Parameters  Set:  Animal
Premise:    Neck  =  Long  or  Short
Action:      Set  value  of  'Animal  AND  END
        OR  Test  'Trunk'
Description:    Depending  on  value  of  'Neck',
      identify  'Animal'  as  'Giraffe'  or
       test  'Comfort  Zone'

Type:        If-Then-Else
Name:        Rule   4
Status:           Variable
Parameters  Tested:  Trunk
Parameters  Set:  Animal
Premise:    Trunk  =  True  or  False
Action:       Set  value  of  'Animal'  AND  END
        OR  Test  'Comfort  Zone'
Description:  Depending  on  value  of  'Trunk',
      identify  'Animal'  as  'Elephant'  or
        test  'Comfort  Zone'

Type:        If-Then-Else
Name:        Rule   5
Status:           Variable
Parameters  Tested:  Comfort  Zone
Parameters  Set:  Animal
Premise:   Comfort  Zone  =  Water  or  Dry  Land
Action:       Set  value  of  'Animal'  AND  END
Description:   Depending  on  value  of  'Comfort
Zone',
               identify 'Animal' as 'Hippo' or
               'Rhino'
```


Animal Decision Tree: Programs

Procedural Sequence of Rules

```
Rule1(Size, Rule2, Rule3)
Rule2(Sound, Animal, Animal)
Rule3(Neck, Animal, Rule4)
Rule4(Trunk, Animal, Rule5)
Rule5(Comfort Zone, Animal, Animal)
```

Declarative Sequence of Rules

```
BasicRule(Size, Sound, Neck)
BasicRule(Sound, Animal, Animal)
BasicRule(Neck, Animal, Trunk)
BasicRule(Trunk, Animal, Comfort Zone)
BasicRule(Comfort Zone, Animal, Animal)
```

Animal Decision Tree:
Rule-Based Approach

- Well suited to simple graphical user interface (GUI)

Frame Type	Parameter
Name	
Current Value	
Rules That Test	
Rules That Set	
Allowable Values	
Description	

Frame Type	Rule
Name	
Status	
Parameters Tested	
Parameters Set	
Premise	
Action	
Description	

Rule-Based Approach:
Training and Chaining

- GUIs for training and operations

Training	
Decisions	
Attributes	
Training Trials	

Chaining	
Input Parameter	
Output Parameter	

Animal Decision Tree: Procedural Logic

Simple exposition of decision-making
Rigid description of solution

```
If Size = Big
  Then If Sound = Squeak
    Then Animal = Mouse
    Else Animal = Squirrel
      EndIf
  Else If Neck = Long
      Then Animal = Giraffe
       Else If Trunk = True
          Then Animal = Elephant
           Else If Comfort Zone = Water
               Then Animal = Hippo
               Else Animal = Rhino
                   EndIf
            EndIf
      EndIf
EndIf
```


Next Time: State Estimation

Supplementary Material

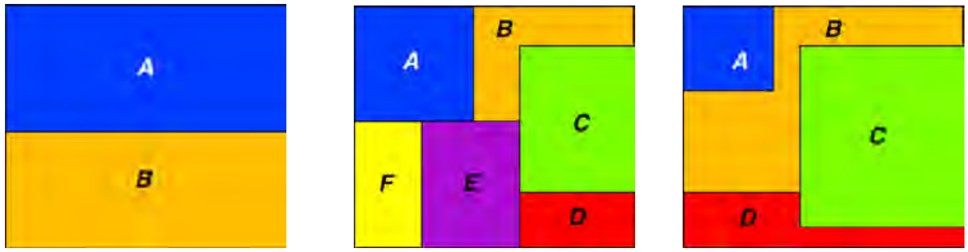
Example: Probability Spaces for Three Attributes

- Probability of an attribute value represented by area in diagram

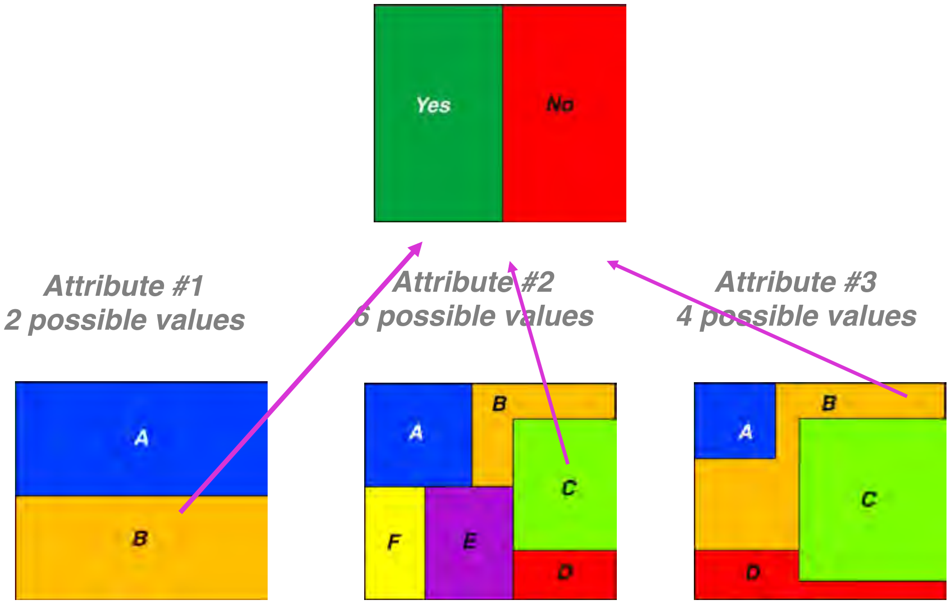
Attribute #1
2 possible values

Attribute #2
6 possible values

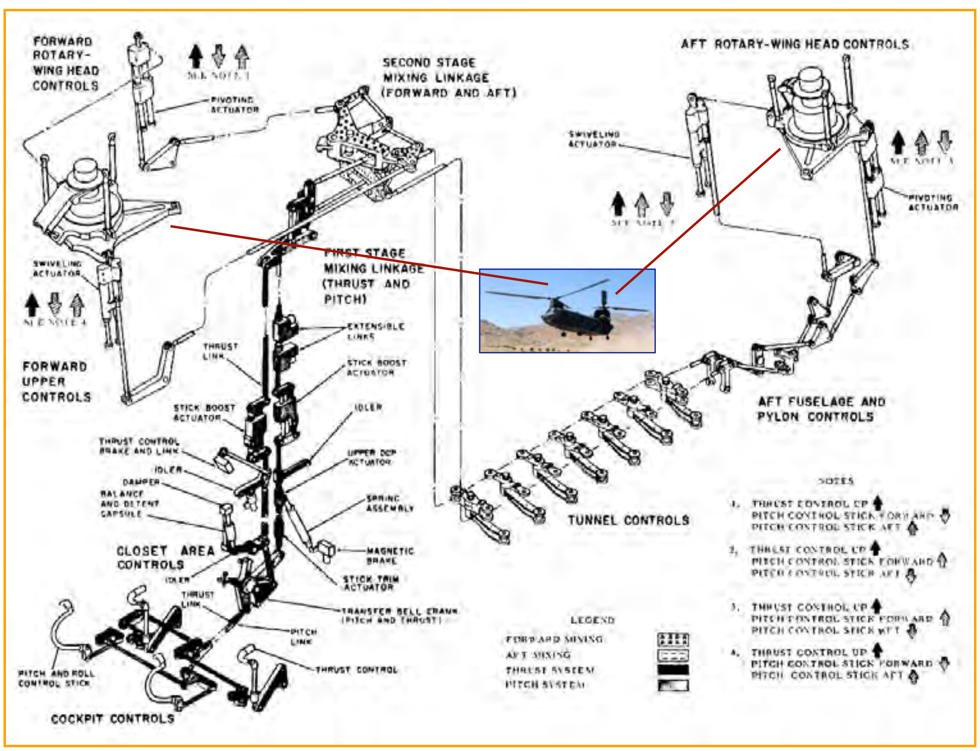
Attribute #3
4 possible values



Example: Decision, given Values of Three Attributes

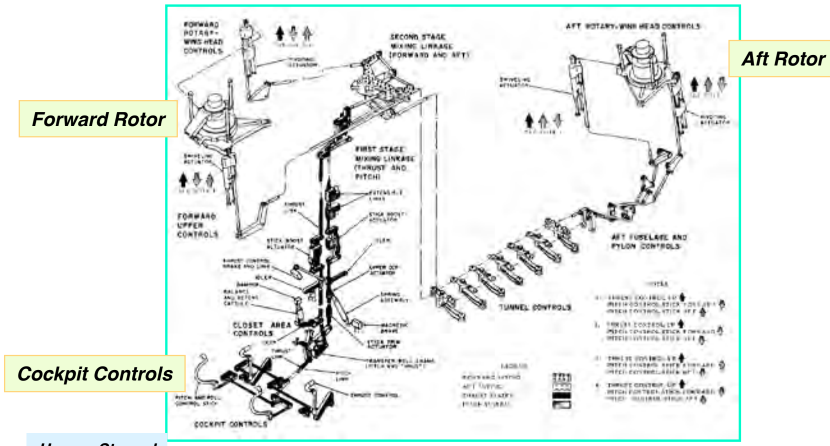


Mechanical Control System



Inferential Fault Analyzer for Helicopter Control System

- Local failure analysis
  - Set of hypothetical models of specific failure
- Global failure analysis
  - Forward reasoning assesses failure impact
  - Backward reasoning deduces possible causes



Huang, Stengel

Heuristic Search

- Local failure analysis
  - Determination based on aggregate of local models
- Global failure analysis
  - Determination based on aggregate of local failure analyses
- Heuristic score based on
  - Criticality of failure
  - Reliability of component
  - Extensiveness of failure
  - Implicated devices
  - Level of backtracking
  - Severity of failure
  - Net probability of failure model

Local Failure Analysis

- **Frames** store facts and facilitate search and inference
  - Components and up-/downstream linkages of control system
  - Failure model parameters
  - Rule base for failure analysis *(LISP)*

Local Failure Model #1
The cause of Nodes 9-2 (1.0) & 17-2 (1.0) being down
MAY be that Node 8-2 (1.0) is down

==

Local Failure Model #2
The cause of Nodes 9-3 (1.0) & 17-3 (1.0) being down
MAY be that Node 8-3 (1.0) is down

==

Local Failure Model #3a
The cause of Nodes 17-2 (1.0), 9-2 (1.0) & 18-2 (1.0) being down
MAY be that Node 7-2 (0.67) is down
This **IMPLICATES** Nodes 8-2, 15, 3, & 11-2

==

Local Failure Model #4
The cause of Nodes 5 (1.0) & 16 (1.0) being down
MAY be that Node 2 (1.0) is down

==

Global Failure Analysis

Global Failure Model #1

Formed from local model(s): 1,2 Score: 32.5
Flagged Devices: Torquemeter-1, Torquemeter-2, Eng-oil-temp-1, Eng-oil-temp-2, Eng-oil-press-1, Eng-oil-press-2, Flow-meter-1
Probable Cause: Engine-#1 (0,75), Engine-#2(0.75)
Implicated End-Devices: Pump-press-sensor-1, Pump-press-sensor-2, Actr-press-sensor-1,-2,-3,-4, Aft-yaw-&-roll, Aft-pitch-&-heave, Eng-chip-detector-1, Eng-chip-detector-2.

==

Global Failure Model #2

Formed from local model(s): 1,3 Score: 30.875
Flagged Devices: Torquemeter-1, Torquemeter-2, Eng-oil-temp-1, Eng-oil-temp-2, Eng-oil-press-1, Eng-oil-press-2, Flow-meter-1
Probable Cause: Engine-#2 (0,75), Fuel-System-#1(0.675)
Implicated End-Devices: Pump-press-sensor-1, Pump-press-sensor-2, Actr-press-sensor-1,-2,-3,-4, Aft-yaw-&-roll, Aft-pitch-&-heave, Eng-chip-detector-1, Eng-chip-detector-2.

==

Global Failure Model #3

Formed from local model(s): 2 Score: 19.25
Flagged Devices: Torquemeter-1, Eng-oil-temp-1, Eng-oil-press-1
Probable Cause: Engine-#1 (0,75)
Implicated End-Devices: Pump-press-sensor-1, Eng-chip-detector-1, Actr-press-sensor-1.

==