

# Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

[Introduction](#)

[Topic 1: Cognition, adaptation, and computation](#)

[Topic 2: State-space search](#)

[Topic 3: Knowledge representation and rule-based inference](#)

[Topic 4: Uncertainty and probabilistic reasoning](#)

[Topic 5: Supervised learning](#)

[Topic 6: Reinforcement learning and adaptation](#)

[Topic 7: Multi-stream adaptive interaction](#)

[Topic 8: Future prospects and philosophical considerations](#)

[Summary](#)

[References](#)

# Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

## Introduction

1. What this course is about
2. The rational-agent approach to AI
3. Course plan and details

### 1. What this course is about

#### AI quantifies subjectivity

- Belief (Bayes)
- Probability
- *Plausibility* (Dempster-Shafer)
- *Possibility*: fuzzy sets, Zadeh, correspond to notions of “very,” “a little,” “slightly,” “almost completely,” “somewhat”
- Nervous system does this even unconsciously, whereas AI has to do it explicitly

#### Definitions of AI<sup>1</sup>

- “The collection of problems and methodologies studied by AI researchers”
- “The branch of computer science that is concerned with the automation of intelligent behavior”
- Two meanings to consider:
  - What is the research field of AI? What does AI research investigate?
  - What is intelligence? (Because “artificial” here just means machine based)

Note: for #2, some will say intelligence is what we *implement* artificially, others will say it's what we *mimic* with AI

- “AI is the study of the mechanisms underlying intelligent behavior through the construction and evaluation of artifacts designed to enact those mechanisms”<sup>2</sup> --less a theory than a methodology for testing theories

#### Early AI concepts

- *Physical symbol system hypothesis*: “The necessary and sufficient condition for a physical system to exhibit general intelligent action is that it be a physical symbol system” (Newell and Simon, 1976)<sup>3</sup>
- General intelligent action<sup>4</sup>: “The same scope of action seen in human action. Within physical limits, the system exhibits behavior appropriate to its ends and adaptive to the demands of its environment.” (Newell and Simon, 1976)

#### The Turing-test definition of AI<sup>5</sup>

- *Imitation game*: interrogator; human answerer; computer answerer
- Interrogator attempts to ask questions that enable detection of which answerer is the computer
- An objective test
- Bias is in favor of symbolic problem solving tasks, rather than perception and physical action
- Critique of the Turing test<sup>6</sup>
  - Depends on subjectivity
  - Questionable objective
  - Hopelessly culture-bound

<sup>2</sup> Luger05, p. 825

<sup>3</sup> Luger05, p. 826

<sup>4</sup> Luger05, p. 826

<sup>5</sup> Luger05, pp. 13ff. See Turing, 1950

<sup>6</sup> Ford and Hayes, 1998, in Fritz, Understanding AI, 2002, p. 7

<sup>1</sup> Luger05, pp. 1-2

## Knowledge representation and search<sup>7</sup>

- Identified as the two chief concerns of AI research
- *Knowledge representation*:
  - one tool: predicate calculus
  - fuzzy logic and natural language processing address ambiguities
- *Search*: Intelligent behavior requires search of a space of possible solutions

## AI “application” domains<sup>8</sup>

- Game playing
- Automated theorem proving
- Expert systems using inference
- Natural language processing
- Robotics and planning
- All these have huge defects as application areas – non interactive toy problems

---

<sup>7</sup> Luger05, p. 20

<sup>8</sup> Luger05, pp. 20-26

## 2. The rational-agent approach to AI

### Example: Chess

- Chess is fully observable, so model of environment is irrelevant
- Reflex agent looks up or calculates a move based on board position
- Goal-based agent will opt for checkmate, will avoid being checkmated if possible, using model
- Utility-based agent will evaluate and compare board positions for best one to move to

## 2. Course background

- Set theory
- Predicate calculus
- Probability
- Data structures
- Algorithms and complexity theory
  - Brute force
  - Greedy
  - Divide and conquer
  - Dynamic programming
  - Probabilistic and approximate
  - $O(1)$ ,  $O(\lg n)$ ,  $O(n)$ ,  $O(n^2)$ ,  $O(2^n)$  time
- Graph theory

### Algorithm

- **Definition:** A precise plan to solve a *functional problem* in a finite number of steps
- Program designs use algorithms that convert pre-specified input to output
- Algorithms compute functions from *input* to *output*
- In this classroom, “function” means “mapping,” not “subprogram”
- We may associate algorithms with the computation modeled by Turing machines and by programs that accept input, then process it, then produce output

### Undecidability and intractability

- Some informational problems can be solved *algorithmically*, in reasonable time. *Example:* Sorting or searching a list.
- Some algorithmic problems have no solution that takes less than time exponential in the amount of input: *intractable*
- Some problems lack an algorithmic solution: *undecidable* (for yes/no problems); *uncomputable* (for functions)

### Mathematical foundations<sup>9</sup>

- *Logic* (Aristotle, Boole, Frege)
- *Algorithms* (Euclid, Al-Kuwarizmi, Godel, Turing)
- *Computational complexity* (notions of intractability, i.e., problems solvable but too time-consuming for practical solutions)
- *Probability* (Cardano, Fermat, Pascal, Laplace, Bayes)

### Sciences<sup>10</sup>

- *Economics*
  - Is about decision making under uncertainty, with preferred outcomes (utility)
  - Seeks *satisficing* (good enough decisions)
- *Neuroscience:*
  - The brain is composed of neurons
  - Its parallelism makes it orders of magnitude more powerful than a PC
- *Psychology:* Behaviorism eclipsed cognitive research for a time

### Consciousness and intelligence<sup>11</sup>

- Consciousness is awareness
- Intelligence requires consciousness
- Intelligence cannot be implemented by algorithmic methods alone; requires a non-algorithmic component [DK]

<sup>9</sup> RN03, pp. 7-9

<sup>10</sup> RN03, pp. 9-12

<sup>11</sup> RN03, pp. 406-407

## Computer-related sciences<sup>12</sup>

- *Computer engineering* progressed from mechanical to vacuum tube to transistor, with performance doubling every 1.5 years
- *Control theory* studies artifacts that could control their own behavior using feedback
- *Linguistics*: born at the same time as AI, in mid-1950s

## Pre-history of AI<sup>13</sup>

- Aristotle:  
*Physics*: matter vs. form, precursor for *abstraction* in computer science  
*Logic*: inference
- *Renaissance science*:
  - Use of mathematical tools to understand nature
  - Study of thought itself
- *Mechanical calculation*: abacus; Napier's bones; Pascal's calculator; Leibniz's imagined machine
- Leibniz: formal logic applied by machine
- Babbage: programmable computer
- Boole: formalization of laws of logic
- Frege: predicate calculus
- Russell-Whitehead: math as a purely formal system
- Tarski: Theory of a reference semantics for applying formulas to real world

[See also slides, "Computer-science background to AI"]

---

<sup>12</sup> RN03, pp. 12-14

<sup>13</sup> Luger05, pp. 9-13

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 1: Cognition, adaptation, and computation

1. Cognition
2. Models of computation
3. Intractability

#### 1. Cognition

##### Foundations of cognitive science<sup>14</sup>

- Cognitive psychology
- Philosophy
- AI
- Linguistics
- Neuroscience
- Cognitive anthropology

##### Logic<sup>15</sup>

- *Propositional* (AND, OR, NOT, atoms)
- *Predicate* (universal and existential quantifiers)
- *Modal* (operations for necessity and possibility)
- *Epistemic* (operators for knowledge and belief)
- *Deontic* (operators for permissibility)
- *Abduction* (generalization from facts)
- *Bayesian reasoning* (explain using prior and conditional probability)

##### Rule-based systems<sup>16</sup>

- Distinguished from logic in that conclusions from rules often express goals
- Fundamental operation of thinking is *search*
- Facts represent short-term memory, rules represent long-term
- Rule-based model is learning-relevant, psychologically and neurologically plausible, and practically applicable

##### Problem solving and learning with concepts<sup>17</sup>

- Planning may consist of concept application, rather than deduction or search
- Concepts may be useful in explaining situations
- Some concepts may be innate; others learned by example
- Concepts are represented as *mental lexicon*
- Evidence of reality of concepts is chiefly by human experiment rather than simulation
- Learning a discipline may require deliberate change of concepts

##### Abstraction and knowledge<sup>18</sup>

- Abstraction enables knowledge via generalization
- The world may be organized as hierarchies of concepts
- “Most of our ideas are about other ideas”
- Two levels are identified by Newell: symbol, knowledge
- Symbol level is utilized as representation formalism

##### Images in cognition<sup>19</sup>

- Much computation is needed to process representation as images
- Some information is well represented by images, other not
- It is sometimes useful to reason or plan using visual information, e.g., about spatial relationships or visual appearance
- Imagery is often part of decision making, learning, explanation, and language

<sup>14</sup> Thagard05, pp. 8-10

<sup>15</sup> Thagard05, Ch. 2

<sup>16</sup> Thagard05, Ch. 3

<sup>17</sup> Thagard05, Ch. 4

<sup>18</sup> Luger05, pp. 636-

<sup>19</sup> Thagard05, Ch. 6

## A neurological theory of consciousness<sup>20</sup>

- Crick, 1994: consciousness involves
  - Attention
  - Short-term memory
- Consciousness of emotions is associated with sensing of bodily reactions
- Two kinds (Damasio, 1999):
  - Core (basic wakefulness and attention situated in old brain structures)
  - Extended (sense of self based on autobiographical memory)
- Consciousness can be understood as a computational-representational process

## Minds and their environments<sup>21</sup>

- Challenges to computational-representational understanding of mind (CRUM):
  - Thought is embodied
  - Mind, body are situated in physical world
  - Concepts often derive from bodily experience, e.g., “up”
- Heidegger challenged representational view, asserted that skills are nonrepresentational
  - Situations and contexts play large role in problem solving and learning

## Intelligence as knowledge generation

*Four ways to create knowledge:*

- Inference, reasoning
- Creativity
- Learning by interaction
- Learning in social environment

## Intelligence<sup>22</sup>

- Is it learned or innate?
- How does it relate to intuition, creativity, self-awareness?
- Does it require certain internal processes, or is it inferable from behavior?
- Does it require humanlike sensation and experience?
- Q: Is the human species the only one capable of exhibiting intelligent behavior? Can ants show intelligent behavior?

## Definitions of intelligence

- “Making wise choices of what to do next” (Newell and Simon, 1976 Turing lecture)
- Adaptive interaction
- An attribute of a social community
- An attribute of natural evolution
- A collection of behaviors including
  - Inference
  - Gaining, storing, and retrieving knowledge
  - Learning from environment
  - Goal seeking
  - Planning

## Communication among neurons<sup>23</sup>

- *Action potentials* (nerve impulses): pulses of a few thousandths of a second pass along axons at ~45 ft/sec
- At axon terminals (synapses), chemical signals (*neurotransmitters*) are triggered, capable of stimulating or inhibiting signals in destination neurons
- Transmitter molecules bind to *receptor* molecules on destination neurons, activating them
- Sum of all excitatory and inhibitory impulses in destination neuron determine its production of impulses, if a threshold voltage is reached
  - Hence neuron is both *analog* (sum of potentials in  $\mathbf{R}$ ) and *digital* (neuron either fires or not at time  $t \in \mathbf{R}$ )
  - Each neuron may have 1000 synapses
  - Number of neurons: 100 billion, hence  $10^{14}$  connections
  - [Pic, p. 30]

<sup>20</sup> Thagard05, Ch. 11, pp. 183-189

<sup>21</sup> Thagard05, Ch. 12

<sup>22</sup> Luger05, p. 1

<sup>23</sup> Luger05, pp. 28-29

## Glial cells

- 1 trillion in human brain
- Manufacture myelin, which sheaths neurons
- Have effect on synapse transmission
- Part of control system in brain

## Memory formation<sup>24</sup>

- *Second messengers* are internal chemical carriers of information within a neuron
- They may take information to nucleus with result of changing strengths of synapses
- Thus threshold values may be changed, changing responses of neurons

## Broadcast information<sup>25</sup>

- Freely-diffusing messenger molecules may take information to many parts of the brain
- Nitric oxide is one such molecule

## Parts of the brain<sup>26</sup>

- *Forebrain*: executive source of commands, decisions, judgments; larger in mammals than in fish and reptiles (cerebral cortex)
- *Mid and hindbrain* (brain stem): controls non-cognitive body functions like breathing and coordination
- *Cerebellum* (in hindbrain); coordination of motor commands; stores learned skills
- *Hypothalamus*: link of brain with hormonal system
- *Amygdala*: stores memories that trigger emotions, e.g., fear

## The cerebral cortex<sup>27</sup>

- Most recently evolved part of forebrain
- Location of planning, will, language, consciousness, imagination
- Exploded in size as primates evolved to humans: 300 cc (chimp) to 1400 cc (humans)
- Frontal cortex defines personality

## Left and right brain<sup>28</sup>

- Left brain (cortex) handles language, right handles geometrical thinking and music
- Some researchers view left, right brains as separate personalities

## Vision is interaction<sup>29</sup>

- The eye scans its field in an active saccadic feedback loop that determines where to look next
- Interpretation of information is a heavy part of vision
- Vision concentrates on what stands out from the background
- *Superior colliculus* participates in changing directions of gaze from one object of interest to another
- Brain also uses hearing to direct the eye

## Memory<sup>30</sup>

- Memory enables learning from experience
- Working memory is in cerebral cortex
- Long-term memory involves physical changes in synapses
- Commonplace facts (declarative knowledge) and procedural knowledge (how to do things) as well as episodic (event) memory are stored as long-term memory

## Adaptation and synapse change<sup>31</sup>

- Change in strength of synapses is the means of recording memory and learning
- This is done by chemical means
- Genes in neuron nuclei interact with synapses to produce long-term change

---

<sup>24</sup> Luger05, p. 40

<sup>25</sup> Luger05

<sup>26</sup> Luger05, pp. 53-57

<sup>27</sup> Luger05, p. 57

---

<sup>28</sup> Luger05, pp. 384-386

<sup>29</sup> Luger05, pp. 70-77

<sup>30</sup> Luger05, pp. 84ff

<sup>31</sup> Luger05, p. 98

## Consciousness<sup>32</sup>

- The cerebral cortex provides executive functions, such as planning and much of memory
- Spinal cord provides reflexes, which are unconscious
- Learned rote behavior is sometimes performed without conscious attention; this processing is in cerebellum, below and rear of cerebrum
- *Motivation and conscious intention* may be located in upper brain stem together with cortex
- Also, the *reticular formation* is related to awakesness; hippocampus, which is associated with longterm memory

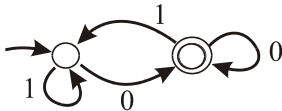
---

<sup>32</sup> Penrose, 1989, pp. 374-383

## 4. Models of computation

### Finite automata

- The automata below accepts strings that start with any number of 1's (possibly none), followed by a 0, followed by any number of (01)s, followed by any number of 0s



### Turing machines

- Based on human “computer” with paper and pencil
- Operations*: tape head reads a symbol at current location on paper tape, moves left or right, writes symbol at current location
- Next action is looked up in *transition table* based on current input and “state of mind” of computer
- Machine halts when it enters a “halting” (accept or reject) state

### Church-Turing thesis

- Thesis*: A problem is solvable algorithmically iff some Turing machine, for all inputs, solves the problem and halts.
- CTT Formalizes the intuitive notion of an algorithm
- Not a provable *theorem*
- Lack of counter examples so far lends strong support to CTT

### TMs and computability

- Turing’s model enabled the invention of general-purpose computers
- The model is equivalent to the notions of an algorithm, or of a program that terminates
- The model enables us to reason about intractable and undecidable problems, saving us the time we might spend trying to solve them
- Helps us capture essence and limits of algorithmic computing
- Equivalent models of algorithmic computation: random-access machines; recursively definable functions

## 5. Undecidability and intractability

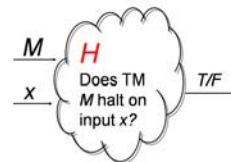
- sketching the *limits* of algorithmic computability by showing that certain problems have no algorithmic solution
- defining a class of problems whose solutions are so time consuming that they are not worth solving

### Undecidability

- Some languages are not decidable, e.g., the set of descriptions of TMs that halt
- Some undecidable languages are recognizable
- Goal*: to find the limits of algorithmic computation, i.e., the computable functions
- Main result*: *Nothing* about the language accepted by an arbitrary TM or program can be determined by looking at its code

### The halting problem<sup>33</sup>

- Consider the decision problem or language *HALT*, consisting of the set of pairs  $(M,x)$  s.t. the TM with description  $M$  halts on input  $x$
- Is there a TM that decides *HALT*?

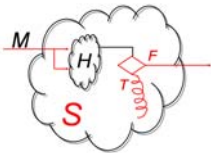


<sup>33</sup> D. Harel, *Algorithmics*

## Undecidability of HALT<sup>34</sup>

*Proof:*

- Suppose *HALT* is decidable.
- Then construct TM *S* from *HALT*-decider *H*, where *S* makes a copy of its input *M*, feeds *M* and *M* to *H*, loops forever if  $\phi_H(M,M) \downarrow$ , halts if  $\phi_H(M,M) \uparrow$
- Consider whether *S* halts on input *S*. If  $\phi_S(S) \downarrow$  then  $\phi_S(S) \uparrow$  and conversely -- a contradiction.
- Since *S* is clearly constructible except for component *H*, therefore *H* cannot be constructed
- Hence *HALT* is undecidable.



## Computation complexity

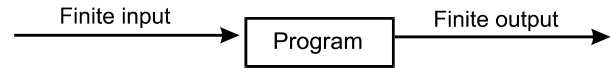
- We analyze the complexity of *problems*, as opposed to *algorithms*
- The complexity of a *problem* is the complexity of the most efficient algorithm that solves it
- We prove complexity by *reducing* a problem of known complexity to a problem of unknown complexity
- Problems may be categorized as *tractable* (*P*, polynomial-time) or *intractable* (NP-hard)

## Satisfiability of logical formulas

- Let  $SAT = \{ \phi \mid (\exists \rho) \phi_\rho = \text{true} \}$  where  $\phi$  is a formula in logic and  $\rho$  is a set of truth assignments
- *Example:*  $\rho = \{ (p, \text{true}), (q, \text{true}), (r, \text{false}) \}$
- Then *SAT* is the set of formulas that are *satisfiable*
- Thus *SAT* is a language
- Equivalently, *SAT* is the *decision problem* of telling whether a formula is satisfiable (whether a formula is in the language *SAT*)
- No algorithm is known that solves the *SAT* problem in time less than  $O(2^n)$ ; hence *SAT* is considered intractable

## Algorithms vs. interaction

- **Algorithmic computation (D. Knuth):**  
The effective transformation of a finite, pre-specified input, to a finite output, in a finite number of steps.



- Algorithms compute functions
- A system that executes an algorithm is *closed*
- Algorithms are equivalent to Turing-machine computation

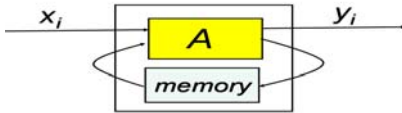
## Kinds of concurrency

- Sequential interactive computation alternates input with output
- Multitasking and threads may run on one processor
- *Multi-stream interaction* may bring together more than two computing entities and may entail *indirect interaction*
- *Parallel computation* may be algorithmic but involves interaction and sharing of memory among processors
- *Distributed computing* involves communication at a distance
- *Proposition:* Turing machines model none of these adequately

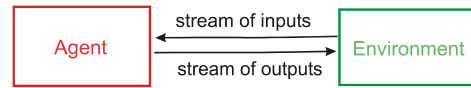
<sup>34</sup> D. Harel, *Algorithmics*

## Sequential interaction

- Feature of computing today: Computation as an ongoing *service*, not assumed to terminate
- Dynamic input and output *during* computation
- *Persistence of state* (memory) between interaction steps



- 



- Characterized by a single interaction stream of input alternating with output
- If one participant is an agent, the other is its *environment*
- Interaction may involve changes of state

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 2: State-space search

1. State-space search
2. Hard problems in constraint and optimization
3. Heuristics

#### 1. State-space search

##### State-space problem representation<sup>35</sup>

- Nodes of a graph correspond to state in solution space
- *Initial states* represent problem information
- *Goal conditions* are solution states or properties of paths through space (graph)
- *State-space search* seeks a *solution path* from initial state to a goal state
- Searches of *trees* have unique path to any state; other state-space searches may loop infinitely

##### State-space search and heuristics<sup>36</sup>

- *State space*: a set of possible situations; e.g., game board configurations, or known facts about a sick person
- *Search*: traversal of a path in state space toward a solution to a problem
- *Exhaustive search*: search of all of state space; usually space is too large for exhaustive search in reasonable time
- *Heuristic*: a strategy or rule of thumb for searching a state space

##### Backtracking<sup>37</sup>

- Begins at start state, continues until either a goal or a dead end
- If a dead end
  - back up to most recent node with unvisited siblings
  - continue searching that branch
- Uses stack or recursion to store states to backtrack to
- Implements either data- or goal-driven search

##### Breadth-first and depth-first search<sup>38</sup>

- *Breadth-first*: visits each node adjacent to start node, advances slowly outward
- *Depth-first*: traverses one path, then backtracks one step, then two, etc.
- Breadth-first search finds shortest path, but uses space exponential in path length
- *Variant*: depth-first search with bound on depth (“iterative deepening”)

#### 2. Hard problems in constraint and optimization

##### Fitness and function optimization

- *Example*: Suppose  $f$  is viewed as a *fitness* function and  $x$  is the set of attributes of individuals of a population
- Then finding  $x$  s.t.  $f(x)$  is maximal is finding the fittest possible individual of the species, i.e., those with the best attributes to assure survival
- Evolution by natural selection tends to optimize fitness, over many generations

<sup>35</sup> Luger05, p. 87

<sup>36</sup> Luger05, pp. 43-44

<sup>37</sup> Luger05, pp. 96-98

<sup>38</sup> Luger05, pp. 99-106

### 3. Heuristics<sup>39</sup>

#### Simulated annealing and local beam search<sup>40</sup>

- *Simulated annealing*: random choice is used to “shake up” the state transitions out of local optima
- *The intensity of shaking-up* is progressively reduced (analogous to lowering of temperature in metal annealing)
- *Beam search* uses a population of states, choosing the best performers for re-generation

#### Transitions in state space<sup>41</sup>

- Application of inference rule is one type of state transition from a less-knowledge state to a greater-knowledge one
- Other possible state transitions:
  - exploring game-board states
  - policy search
- See *graph theory* for framework of state transitions

#### Hill climbing<sup>42</sup>

- *Heuristic strategy*: Choose “best” child of current state
- *Drawback*: tendency to become stuck at local maxima that are not goal states, where best child nodes are always inferior to current node, but better states do exist beyond immediate child nodes

#### Best-first search<sup>43</sup>

- Heuristics are used to evaluate states adjacent to current one
- The one evaluated as best is selected
- This process continues until goal state is reached

#### Features of heuristics<sup>44</sup>

- *Admissibility*: finding shortest path to a goal if it exists
- *Informedness*: superiority of heuristic
- *Monotonicity*: local admissibility

#### A\* algorithms<sup>45</sup>

- Search algorithms whose evaluation function is sum of  $g(n)$ ,  $h(n)$ , where  $g$  is cost of state  $n$  from start, and  $h$  is estimated cost of going from  $n$  to a goal, are called A\*
- *Theorem*: All A\* algorithms are admissible, i.e., find shortest path

#### Monotonicity of heuristics<sup>46</sup>

- This is local admissibility, i.e., ability to consistently find shortest path to each state encountered
- Monotone heuristic functions enable avoidance of cases where goal states are first reached via paths that are not minimal

#### Informedness of heuristics<sup>47</sup>

- A heuristic  $h_2$  is more informed than heuristic  $h_1$  if for every state  $n$ ,  $h_1(n) \leq h_2(n)$
- A more informed heuristic expands a smaller part of the state space

#### Broader considerations on games<sup>48</sup>

- Games of chance greatly complicate minimax
- Belief states are quite important in games involving choice
- Other approaches exist than standard minimax, alpha-beta, and evaluation functions
- General questions of decision making and meta reasoning are not illuminated by the standard approach

#### Minimax<sup>49</sup>

- Heuristic algorithm for two-player game playing
- Assumes opponent will make optimal moves, i.e., moves that favor 0 outcome
- Each leaf node is either 0 (loss) or 1 (win)
- Algorithm works backward from leaves toward root
- If parent state is at level corresponding to player’s move, assign it maximum among children values; otherwise assign min.

<sup>39</sup> Luger05, pp. 123-124

<sup>40</sup> RN03, pp.115-116

<sup>41</sup> RN03, pp. 115-116

<sup>42</sup> Luger05, pp. 127-128

<sup>43</sup> Luger05, pp. 133-136

<sup>44</sup> Luger05, pp. 145ff

<sup>45</sup> Luger05, p. 146

<sup>46</sup> Luger05, pp. 147ff

<sup>47</sup> Luger05, pp. 148-149

<sup>48</sup> RN03, pp. 173-185

<sup>49</sup> Luger05, pp. 150-152

## 4. Rule-based production systems<sup>50</sup>

- A pattern-directed procedure for problem solving
- *Production rules*: condition-action pairs denoting an item of knowledge
- *Working memory*: describes current state of world, is updated by actions in condition-action pairs
- *Recognize-act cycle*:
  - Patterns in working memory are matched with production rules and problem description
  - One element of set of rules satisfied is chosen to fire

### Production system example: text<sup>51</sup>

- Rules (for alphabetizing):
  1.  $ba \rightarrow ab$
  2.  $ca \rightarrow ac$
  3.  $cb \rightarrow bc$
- Working memory:
 

|       |        |
|-------|--------|
| acbac |        |
| acabc | rule 3 |
| aacbc | rule 2 |
| aabcc | rule 3 |
- Production system here transforms “input” *acbac* to “output” *aabcc*

### Production rules and working memory<sup>52</sup>

- Rules represent long-term memory
- Working memory represents a state in problem-solving process
- Production systems are a model of human problem solving and of computation equivalent to Turing machines or algorithms

### Production system example: inference<sup>53</sup>

- 1)  $p \wedge q \rightarrow \text{goal}$
  - 2)  $r \wedge s \rightarrow p$
  - 3)  $w \wedge r \rightarrow q$
  - 4)  $t \wedge u \rightarrow q$
  - 5)  $v \rightarrow s$
  - 6)  $\text{start} \rightarrow v \wedge r \wedge q$
- Execution (data driven, using modus ponens):
    - $v \wedge r \wedge q$  (r6)
    - $v \wedge r \wedge q \wedge s$  (r5)
    - $v \wedge r \wedge q \wedge s \wedge p$  (r2)
    - $v \wedge r \wedge q \wedge s \wedge p \wedge \text{goal}$  (r1)

### Advantages of production systems for AI<sup>54</sup>

- Separation of knowledge from control: control is recognize-act cycle, knowledge is production rules
- Natural mapping to state-space search:
  - states of working memory are equivalent to nodes of state-space graph
  - production rules are equivalent to transition rules
- modularity of production rules

### Blackboard problem-solving architecture<sup>55</sup>

- Blackboards coordinate multiple problem-solving “knowledge sources”
- Production rules are grouped in modules
- Knowledge sources obtain data from blackboard and update its contents with new knowledge

## 5. Evolutionary computation

### Evolutionary computation

- State space is explored using and comparing a *population* of states rather than one at a time
- Evolutionary algorithm repeatedly modifies and selects from a population of solutions
- Selection is driven by fitness of each member of the population
- Randomization is used to explore state space

<sup>50</sup> Luger05, pp. 200ff

<sup>51</sup> Luger05, p. 202

<sup>52</sup> Luger05, pp. 202-203

<sup>53</sup> Luger05, pp. 211

<sup>54</sup> Luger05, p. 215

<sup>55</sup> Luger05, p. 217

## The evolutionary algorithm

```

t ← 0
Initialize (P0)
V ← Evaluate (P0)
While not Terminate (V, t) do
    t ← t + 1
    Pt ← Select (Pt-1, V)
    Pt ← Alter (Pt)
    V ← Evaluate (Pt)
Return Pt

```

## Genetic algorithms<sup>56</sup>

- A form of learning inspired by natural selection
- State-space search is accomplished by setting a population of solutions to compete based on fitness
- At each step, the population is selected and regenerated using genetic operators, e.g., mutation and crossover
- *Parameters*: % of population to retain at each generation, which mate and produce offspring, which probability measures, if any, to use

## CNF satisfiability<sup>57</sup>

- CNF: Conjunctive Normal Form in propositional logic, where a CNF formula is one that ANDs a series of OR clauses
- SAT: Satisfiability decision problem to tell, given a CNF formula, whether a set of Boolean variable assignments exists such that the formula evaluates to true
- Examples:  $(p \vee q) \wedge (\neg p)$  is not satisfiable;  
 $(p \vee q) \wedge (\neg p \vee \neg q)$  is satisfiable

## GA solution to CNF-SAT<sup>58</sup>

- Representation: for  $n$  variables,  $n$  bits representing truth assignment
- Example: 101 may mean  $p = \text{true}$ ,  $q = \text{false}$ ,  $r = \text{true}$
- Appropriate genetic operators: crossover, mutation, inversion
- Fitness: number of disjunctive clauses in CNF formula that return true under truth assignment

## Classifier systems<sup>59</sup>

- An extension to GAs and a formula of reinforcement learning
- Components:
  - Inputs from environment, feedback from environment
  - Output via effectors
  - Working memory to integrate rule firing with input information
  - Set of production rules (classifiers) each with condition, action on memory, fitness measure
  - Set of genetic operators to modify production rules
  - System to give credit to successful rules
- Learning features: reward system, rule modification

## Classifier system general example<sup>60</sup>

- Suppose
  - Objects to be classified have six attributes ( $c_1, \dots, c_6$ )
  - Each may have any of five values (1 .. 5)
  - Objects are classified in four classes ( $A_1, \dots, A_4$ )
- Form of classifier rule:  
 $(c_1 c_2 c_3 c_4 c_5 c_6) \rightarrow A_i$   
where  $c_i \in \{1, \dots, 5, \#\}$  ( $\# = \text{don't care}$ )
- Genetic operators adjust the rules to classify objects as desired according to teacher
- In case of matches with two rules, each rule puts a competitive "bid"

<sup>56</sup> Luger05, pp. 509-510

<sup>57</sup> Luger05, pp. 511-512

<sup>58</sup> Luger05, pp. 511-512

<sup>59</sup> Luger05, pp. 519-521

<sup>60</sup> Luger05, p. 521

### Particular example<sup>61</sup>

- Rules, using one classification only:
  - (1 # # # 1 #) → 1     S = 0.6 (fitness)
  - (# # 3 # # 5) → 0     S = 0.5
  - (2 1 # # # 1) → 1     S = 0.4
  - (2 # # 3 # #) → A1    S = 0.2
  - etc.
- Crossover of first two rules yields
  - (# # 3 # 1 #) → 0     S = 0.0
  - (# # # 2 # 5) → 1     S = .57
- Fitness of offspring is weighted function of fitness of parents

### Genetic programming<sup>62</sup>

- Koza (1992) evolved structures that were elements of computer programs in LISP
- Components:*
  - Structures denoting programs
  - Initial structures
  - Fitness measure to evaluate structures
  - Genetic operators
  - Descriptions of members of each generation
  - Termination conditions

### GP example<sup>63</sup>

- Evolving program code from Kepler's Third Law ( $P^2 = cA^3$ ), where  $P$  is planet's orbital period,  $A$  is average distance from sun
- LISP expression is  $P = (\text{sqrt}(*A(*A A)))$
- Sample data:*  $A, P$  for Venus, Earth, Mars, Jupiter, Saturn, Uranus
- Sample data is used to create fitness measures
- [Pix of target program and members of evolving population: Luger05, p. 530]

### The function-optimization problem<sup>64</sup>

- Let  $f: \mathbf{N}^k \rightarrow \mathbf{R}$  for some  $k$  (the *arity* of  $f$ )
- Problem:* Find some  $x \in \mathbf{N}^k$  s.t.  $f(x)$  is maximal
- Example:* Suppose  $x$  is the set of proportions of ingredients in a fuel mixture,  $f(x)$  is fuel efficiency under this mixture
- Optimizing  $f(x)$*  means finding the most efficient mixture
- For an *algorithm* to optimize a function we must have  $f: X \rightarrow Y$  with  $X, Y$  finite

### Fitness and function optimization<sup>65</sup>

- Example:* Suppose  $f$  is viewed as a *fitness* function and  $x$  is the set of attributes of individuals of a population
- Then finding  $x$  s.t.  $f(x)$  is maximal is finding the fittest possible individual of the species, i.e., those with the best attributes to assure survival
- Evolution by natural selection tends to optimize fitness, over many generations

### Example: Checkers (Samuel, 1950s)<sup>66</sup>

- Let fitness function  $f: \mathbf{N}^k \rightarrow \mathbf{R}$  be an evaluator of checkers board positions from a black or red point of view
- Let  $x \in \mathbf{N}^k$  be a  $k$ -tuple of *weights* for each of  $k$  different criteria for evaluating a checkers position
- Example:* let  $x_1$  be relative importance of number of kings,  $x_2$  be relative importance of number of opponent checkers threatened, etc.
- Then writing a good checkers-playing program reduces to finding a good set of relative weights  $x_1, \dots, x_k$  for these criteria

<sup>61</sup> Luger05

<sup>62</sup> Luger05, pp. 524-527

<sup>63</sup> Luger05, pp. 528-529

<sup>64</sup> Keil, EC in DPEs, 2004

<sup>65</sup> Idem.

<sup>66</sup> Idem.

### Checkers heuristics<sup>67</sup>

- A set of checkers-playing heuristics (weights of attributes of a board layout), is evolved (Samuels '59)
- Fitness: rate of wins that a set of heuristics obtains
- Population  $P$  consists of sets of weights (values) of different attributes of a *checkers board configuration*
- E.g., opportunity to jump is of weight 5, opportunity to king is 3, etc.
- EC here refines heuristics that help compute a function from board configurations to (good) moves
- Fitness function is applied by putting heuristics in competition with other heuristics

---

<sup>67</sup> D. Keil, dissertation proposal, 2006

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 3. Knowledge representation and rule-based inference

1. Knowledge and beliefs
2. Logical inference
3. Expert systems and resolution proof
4. Concepts and planning
5. The LISP language

#### 1. Knowledge, beliefs, and reasoning

##### Wumpus world example<sup>68</sup>

- World is  $4 \times 4$  grid, with one wumpus, one heap of gold, and deadly pits
- Agent dies if enters cell containing wumpus or pit
- Agent wins if enters cell containing heap of gold
- Agent has one arrow that it may use to kill wumpus
- Agent receives percepts: *stench* (Wumpus), *breeze* (pit), *bump* (wall), *glitter* (gold), *scream* (wumpus dies)
- Agent may move, turn, or shoot

##### Procedural vs. declarative approach<sup>69</sup>

- To operate on data (facts), users of *procedural languages* must write ad hoc domain-specific code
- *Declarative languages* express knowledge and inference separately and perform domain-independent inference
- Propositional logic has semantics that is *declarative*, context independent, unambiguous, and compositional

##### Diagnostic vs. model-based reasoning<sup>70</sup>

- *Diagnostic rules* enable inference from observations to hidden causes of the observed facts
- *Causal rules* for model-based reasoning form a model of the system

##### Fuzzy logic and sets<sup>71</sup>

- Fuzziness is not uncertainty; *truth*, not *belief*, is quantified at between 0 and 1
- Membership in a fuzzy set such as *Tall* may be *partial*
- *Application*: fuzzy control
- Probability that a set  $S$  is equal to a fuzzy set, such as *Tall*, may be considered

##### Fuzzy sets and fuzzy logic<sup>72</sup>

- L. Zadeh, 1983, proposed *possibility* theory denoting vagueness
- Rejects excluded middle:  $x \in S \rightarrow x \notin \text{complement}(S)$
- Set characteristic function maps to  $0 \leq y \leq 1$ , denotes possibility or confidence measure
- *Example*: *Small-integers* = {1 (1.0), 2(1.0), 3 (0.9, 4 (0.8), ...}
- *Example*: *Tall-persons*  $\cap$  *Med-size-persons*  $\neq \emptyset$

#### 2. Propositional and predicate logic<sup>73</sup>

##### SAT as state-space search

- Satisfiability is reducible to exhaustive search of all variable assignments and finding one that satisfies formula or finding none
- *Goal state*: a variable assignment s.t. formula holds

<sup>68</sup> RN03, pp.197-200

<sup>69</sup> RN03, pp. 240-242

<sup>70</sup> RN03, pp. 259-260

<sup>71</sup> RN03, pp. 526-527

<sup>72</sup> Luger05, pp. 353-357

<sup>73</sup> Luger05

## Interpretations<sup>74</sup>

- An interpretation of a set of formulas is an assignment of truth values to the symbols
- *Example:* An interpretation of  $(p \wedge \neg(p \vee q))$  is  $p = \text{true}, q = \text{false}$ . Under this interpretation,  $(p \wedge \neg(p \vee q))$  is false
- A formula is *satisfiable* if it has an interpretation under which it is true

## Variables in the predicate calculus<sup>75</sup>

- Suppose  $x$  is a variable in the domain of natural numbers
- In expressions  $\text{sum}(x, 5)$ ,  $\text{odd}(x)$ ,  $\text{prime}(x)$ ,  $x$  is a *placeholder* that stands for an unknown *constant*
- “ $\text{sum}(x, 5) = 8$ ”, “ $\text{odd}(x) = \text{true}$ ” are meaningless without having a value for  $x$
- “ $x = 3 \rightarrow \text{sum}(x, 5) = 8$ ”, “ $x = 1 \rightarrow \text{odd}(x) = \text{true}$ ” are meaningful and true statements

## Quantifiers<sup>76</sup>

- Meaningful:  
 $(\exists x) x + 5 = 8$   
 $(\forall x) x + 1 = x$
- Quantifiers *bind* variables so that they can be used meaningfully in sentences

## Semantics of predicate calculus<sup>77</sup>

- In a real-world *domain*, there are relations:  
 $\text{female}(h\_clinton)$  unary  
 $\text{married}(b\_obama, m\_obama)$  binary
- Predicates *female*, *married* here express these relations
- For a domain  $D$ , an *interpretation* over  $D$  is an assignment of entities in  $D$  to constants, assignment of non-empty subsets of  $D$  to variables, with each predicate mapping  $D^n \rightarrow \{t, f\}$

## Truth value (semantics) of predicate-calculus expressions<sup>78</sup>

- Given expression  $E$ , and interpretation  $I$  for  $E$  over domain  $D$ , truth value is as in propositional calculus, in addition to:
- For sentence  $S$ ,
  - $(\exists x) S$  is true iff some assignment under  $I$  has an assignment to  $x$  s.t.  $S$  is true
  - $(\forall x) S$  is true iff  $S$  is true for all assignments of values to  $x$  under  $I$

## Satisfiability, validity, consistency<sup>79</sup>

- An interpretation *satisfies* a sentence if it makes the sentence true.
- If  $I$  satisfies sentence  $S$  for every set of user assignments, then  $I$  is a *model* of  $S$
- $S$  or a set of sentences is *satisfiable* if some  $I$  satisfies it
- A set of sentences not satisfiable is *unsatisfiable*
- If  $S$  is true for all interpretations, then  $S$  is *valid*
- $(\forall x) (P(x) \vee \neg P(x))$  is valid
- A sentence that is not true under any interpretation is *inconsistent*
- $(\forall x) (P(x) \wedge \neg P(x))$  is inconsistent

## Inference<sup>80</sup>

- Expression  $x$  *logically follows* from a set  $S$  of sentences iff every interpretation that satisfies  $S$  also satisfies  $x$
- *Inference rule:* A validity-maintaining procedure for deriving sentences from other sentences
- *Proof procedure:* An inference rule and an algorithm for applying it to a set of sentences to yield a new sentence

## Soundness and completeness<sup>81</sup>

- *Sound:* an inference rule that produces from set  $S$  only sentences that logically follow from  $S$
- *Complete:* a rule that can infer from  $S$  every sentence that logically follows from  $S$
- *Examples:*
  - Modus Ponens:  $(p \wedge (p \rightarrow q)) \rightarrow q$
  - Modus Tollens:  $((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$
  - *And* elimination, *and* introduction, universal instantiation

<sup>74</sup> Luger05

<sup>75</sup> Luger05

<sup>76</sup> Luger05

<sup>77</sup> Luger05, p. 57

<sup>78</sup> Luger05, p. 58

<sup>79</sup> Luger05, p. 62

<sup>80</sup> Luger05

<sup>81</sup> Luger05, pp. 64-65

## Predicate calculus and knowledge representation<sup>82</sup>

- Predicate calculus is a way to represent knowledge of
  - Objects
  - Relations
- Inference rules enable derivation of new knowledge

### Inference as state-space search<sup>83</sup>

- Assertions may be represented as nodes
- Implications may be represented as transitions
- Inference is finding a path from starting assertions to goal assertion
- *Example:*  $((p \Rightarrow q) \wedge (q \Rightarrow r) \wedge p) \Rightarrow r$   
[pic]  
Boxing  $p$  means it is starting data

### Predicate logic<sup>84</sup>

- Elements of natural language include objects (nouns), relations (adjectives and verbs), functions (relations with unique right-hand values)
- Predicate logic (FOL) represents these and notions of *some* or *all* objects
- Each logic has *ontological* commitment (to represent facts about the world) and *epistemological* commitment (to represent beliefs about facts)
- *Properties* are unary relations
- Objects, relations, and functions are represented by *symbols*

### Backward and forward chaining with Horn clauses<sup>85</sup>

- Deciding entailment with Horn clauses by forward chaining is  $O(n)$  in size of knowledge base
- Forward chaining applies Modus Ponens; is sound and complete
- Forward chaining generates all possible entailments
- Forward chaining is data driven
- Backward chaining is goal driven, works backward from query

## 3. Expert systems, resolution proof, and the Prolog language<sup>86</sup>

### Candidate domains for expert systems<sup>87</sup>

- Potential for large cost savings through use of expert knowledge: mineral exploration, medicine, military
- Problem susceptible to symbolic reasoning
- Inaccessibility by humans
- Common sense reasoning not required
- Appropriate problem size and scope

### Roles in development of expert systems<sup>88</sup>

- Domain expert
- Knowledge engineer
  - Familiar with implementation language, knowledge base development
  - May know good questions to ask expert
  - Prepares early prototypes, develops KB iteratively
  - Uses exploratory development
  - Develops system throughout its life cycle
- End user

### Explanation<sup>89</sup>

*Types of query:*

- *Why* (why did system ask for certain info?)  
*Answer:* Current rule that system is trying to fire
- *How* (How did system get a result?)  
*Answer:* sequence of rules used to conclude a result

### Rule-based breadth-first and opportunistic search<sup>90</sup>

- Data-driven search seeks to draw conclusion from premises met
- A refinement is opportunistic search, in which the effort is to use conclusions drawn from previous firing as premise for another firing

<sup>82</sup> Luger05, p. 79

<sup>83</sup> Luger05, pp. 107-109

<sup>84</sup> RN03, pp. 242-246

<sup>85</sup> RN03, pp. 218-220

<sup>86</sup> Luger05, p. 199

<sup>87</sup> Luger05, p. 281

<sup>88</sup> Luger05, pp. 281-282

<sup>89</sup> Luger05, pp. 291ff

<sup>90</sup> Luger05, pp. 293ff

### Efficiency of rule-based search<sup>91</sup>

- Rule order and order of application of tests may affect efficiency
- These choices, like rules, are of heuristic nature

### Strong vs. weak reasoning methods

*Strong:*

- Expert systems
- Rely on domain specific knowledge

*Weak:*

- General problem solvers
- Automated reasoning

### Resolution theorem proving<sup>92</sup>

- A method that adds the theorem's negation to the database of axioms and shows that this leads to contradiction
- Resolution refutation method begins by putting premises or axioms in *clause form*, a normal form using disjunctions of literals (atomic expressions or their negations)

### Example of resolution<sup>93</sup>

- *Goal:* To prove that Fido will die
- *Predicate form*  
 $(\forall x) \text{dog}(x) \rightarrow \text{animal}(x)$   
 $\text{dog}(\text{fido})$   
 $(\forall y) \text{animal}(y) \rightarrow \text{die}(y)$   
 $\text{die}(\text{fido})$  by *modus ponens*
- *Clause form:*  
 $\neg \text{dog}(x) \vee \text{animal}(x)$   
 $\text{dog}(\text{fido})$   
 $\neg \text{animal}(y) \vee \text{die}(y)$   
 Negate the assertion to be proven:  
 $\neg \text{die}(\text{fido})$   
 This leads to a clash after substitutions lead to  $\text{die}(\text{fido})$

### Resolution in propositional logic<sup>94</sup>

- Let  $(p \vee \neg q) \wedge (q \vee r)$
- Observe that either  $\neg q$  or  $q$  is true, hence one operand of the disjunction holds
- $\therefore p \vee \neg q \wedge q \vee r$
- $\therefore p \vee r$
- This can be done for longer disjunctions, as long as they have terms that are negations of each other
- Resolution shortens clauses

### Resolution proof example<sup>95</sup>

- Intuition: smart and not-poor people are happy
- Happy people live exciting lives
- Readers are smart
- Q: John reads and is wealthy. Does anyone have an exciting life?
- *Axioms:*  
 $(\forall x) (\neg \text{poor}(x) \wedge \text{smart}(x) \rightarrow \text{happy}(x))$   
 $(\forall y) \text{read}(y) \rightarrow \text{smart}(y)$   
 $\text{reads}(\text{john})$   
 $\neg \text{poor}(\text{john})$   
 $(\forall z) (\text{happy}(z) \rightarrow \text{exciting-life}(z))$
- *Disjunctive form:*  
 $\text{poor}(x) \vee \neg \text{smart}(x) \vee \text{happy}(x)$   
 $\neg \text{read}(y) \vee \text{smart}(y)$   
 $\text{read}(\text{john})$   
 $\neg \text{poor}(\text{john})$   
 $\neg \text{happy}(z) \vee \text{exciting-life}(z)$
- *Negation of theorem:*  
 $\neg \text{exciting-life}(w)$
- Hence theorem holds
- Resolution refutation proof tree (breadth-first search)
- [pic]
- Since we are led to a contradiction, hypothesis  $\neg e(w)$  must be rejected

<sup>91</sup> Luger05, pp. 296ff

<sup>92</sup> Luger05, pp. 554-556

<sup>93</sup> Luger05

<sup>94</sup> Luger05, p. 561

<sup>95</sup> Luger05, pp. 564-565

## Complexity of resolution proof<sup>96</sup>

- For  $n$  clauses, at first level of proof tree there are  $n^2$  possible ways to combine them
- Problem is exponential-time
- Heuristics are important to solve problem
- *Strategies:*
  - BFS: guarantees shortest solution path
  - Set of support
  - Unit preference
  - Linear input form

## Horn clauses and logic programs<sup>97</sup>

- *Definition:* a disjunction in predicate logic with at most one positive literal, e.g.,  $a \vee \neg b \vee \neg c \vee \neg d$
- Horn clauses are usually expressed as implications:  
 $b \wedge c \wedge d \rightarrow a$
- *Logic program:* a set of universally quantified Horn clauses: goals, facts, and rules
- Logic programs prove *goal clauses*  $\leftarrow b \wedge c \wedge d \wedge \dots$ , and proceed by *goal reduction*

## Goal reduction<sup>98</sup>

- Logic program chooses an arbitrary  $a_i$  from  $\leftarrow a_1 \wedge a_2 \wedge a_3 \wedge \dots \wedge a_n$
- Then chooses a clause to unify with  $a_i$ , reducing that clause
- Proof of goal clauses uses resolution refutation
- Backtracking in clause set implements depth-first search of proof space

## Prolog and resolution proof<sup>99</sup>

- Prolog interpreter executes search for proof of goal clauses
- Proof is guaranteed if goal clauses are valid, unless *cut* truncates search
- Proof provides answer (interpretation) for which goal is true
- Closed world assumption governs Prolog: does not try all interpretations, but only those in program's database
- Assumes all unprovable assertions to be false (note this contradicts Gödel's Theorem)

## Prolog programs<sup>100</sup>

- A program is a set of sentences in predicate logic about a domain
- All implications are in the form of Horn clauses ( $p \leftarrow q$ , where  $p$  is not negated)
- *Syntax of logic*      *Syntax of Prolog*  
 $\neg$                       not  
 $\vee$                       ,  
 $\wedge$                      ;  
 $\leftarrow$                   :-
- *Example:*  
like(fido, john), eat(x, steak)  $\leftarrow$  dog(x)  
Fido likes John and all dogs eat steak

## Prolog queries<sup>101</sup>

- Interpreter responds to user queries at prompt “?-“
- Queries with unbound variables are answered with multiple values if user types “;”
- Query                      response  
?- likes (fido, x)      john  
;                              spot

## Closed-world assumption<sup>102</sup>

- Prolog assumption (aka “negation on failure”) is that all knowledge of domain is contained in database
- Hence anything not provably true is considered false

## Definitions of relationships in Prolog<sup>103</sup>

- Implicit universal quantification holds
- *Example:* friends( $x, y$ ) :- likes ( $x, y$ ), likes ( $y, x$ )  
(Friends are people who like each other)

## The Prolog environment<sup>104</sup>

- To add a fact: *assert* (clause)
- *Example:* assert(likes(fido, Mary))
- To subtract a fact: *retract* (clause)
- To add facts stored in a file: *consult* (filename)
- Lists may be defined [a, b, c, d]
- And queried: ?- member(a, [a, b, c]) returns *yes*

<sup>96</sup> Luger05, pp. 566-567

<sup>97</sup> Luger05, pp. 576-577

<sup>98</sup> Luger05, pp. 577-579

<sup>99</sup> Luger05, pp. 580-581

<sup>100</sup> Luger05, pp. 642- 644

<sup>101</sup> Luger05, pp. 642-644

<sup>102</sup> Luger05, p. 644

<sup>103</sup> Luger05, pp. 644-645

<sup>104</sup> Luger05, p. 646

### Pruning searches with cut<sup>105</sup>

- Prolog *cut* (denoted “!”) terminates goal seeking or backtracking
- Enables saving of memory or time
- Cut may be used to give flavor of subroutine-calling

### Example: knights-tour problem<sup>106</sup>

- *Problem*: to move a knight on part of a chessboard so as to touch each square
- *Solution*: recursive search, using database of valid moves, recursive definition of *path* predicate

### Data structures in Prolog<sup>107</sup>

- Stack, queue, priority queue, set may be implemented using list structure
- Graph search algorithms are supported by these structures
- Farmer-wolf-goat-cabbage river crossing problem may be solved using production system in Prolog

### Search strategies in Prolog<sup>108</sup>

- Default is depth-first search with backtracking
- Prolog techniques exist for
  - more efficient DFS ordered by heuristic means
  - breadth-first search
- Best-first search

### Unification<sup>109</sup>

- Variables are bound by unification rather than evaluation
- *Example*:  $y = x + 1$  attempts to unify “y” with “x + 1” syntactically

### Prolog applications for AI<sup>110</sup>

- Rule-based expert system shell (e.g., car diagnostics)
- Semantic net with inheritance (e.g., birds)
- Frame-structured semantic net
- Version space search
- Explanation-based learning

### Natural language processing in Prolog<sup>111</sup>

- Semantic networks (conceptual graphs, case frames) may be implemented in Prolog
- Syntax can be represented in Prolog
- Prolog supports recursive-descent parsing; context-free parsing; context-sensitive parsing

## 4. Concepts and planning

### Semantic networks<sup>112</sup>

- An alternative to the predicate calculus
- Words have long been defined in terms of other words
- Showing intersection of two word networks can help disambiguate word uses
- *Example*: The following shows how to disambiguate  $Cry_1$  from  $Cry_2$  in a context. When *comfort* is used near “cry,” the meaning is  $Cry_1$ .
  - $Cry_1$  = make *sad* sound
  - $Cry_2$  = make *loud* sound
  - *Comfort* = make less *sad*

### Reasoning with default information<sup>113</sup>

- *Closed-world assumption*: what is not asserted as true is false
- *Unique names assumption*: different names refer to different objects
- FOL does not use these assumptions
- *Negation as failure*: a negative may be considered to hold if proof of affirmative fails
- Circumscriptive reasoning uses *Abnormal* predicate, so that *Flies(Tweety)* holds unless *Abnormal(Tweety)*

<sup>105</sup> Luger05, pp. 652-654

<sup>106</sup> Luger05, pp. 650-652

<sup>107</sup> Luger05, pp. 654-663

<sup>108</sup> Luger05, pp. 663-671

<sup>109</sup> Luger05, p. 675

<sup>110</sup> Luger05, pp. 682-704

<sup>111</sup> Luger05, pp. 704-716

<sup>112</sup> Luger05, pp. 231-234

<sup>113</sup> RN03, pp. 354-358

## Planning algorithms<sup>114</sup>

- *Forward state-space search*:
  - From initial state to result state
  - Applying actions from among those whose preconditions are met
  - Considered too inefficient to be practical, suffering from irrelevant-action problem
- *Backward state-space search* (regression planning):
  - Requires description of predecessor states
  - Considers only relevant actions (those that achieve at least one conjunct of the goal)
- Both forward and backward search require heuristics for efficiency (cost estimates)

## Partial-order planning and planning graphs<sup>115</sup>

- *Least-commitment strategy* delays decisions, leading to partially ordered plans, which are dags rather than sequences
- A *planning graph* is a tool for generating heuristics that relaxes the planning problem
- *Graphplan* algorithm generates a detailed plan from a graph
- Planning-graph construction is in *PTIME*
- *Partial-order planning* is amenable to divide and conquer

## Scheduling<sup>116</sup>

- *Job-shop scheduling problem*: to set a schedule that minimizes time while satisfying resource constraints
- *Critical path method*: eliminate delay between steps on critical (longest) path of operations
- *Complexity*:  $O(nb)$ , where  $n$  is number of actions,  $b$  is branching factor in or out of action
- *Aggregation* of resources of which there are multiple instances simplifies schedule search
- Resource-constrained scheduling is NP-hard

### Approaches:

- *Early*: semantic networks, conceptual dependencies, scripts, frames
- *Conceptual graphs* used in natural-language systems
- *Decentralized non-explicit* representation: subsumption architecture
- *Agents*

## Problem-solving approaches<sup>117</sup>

- Syntactic general-purpose (weak) problem solving (Newell and Simon, 1956)
- Domain-specific heuristic based (strong) problem solving (Feigenbaum and others)
- Distributed and embodied approaches with robotics (Brooks, 1989)

## Conceptual dependency theory<sup>118</sup>

- Roger Schank, 1974
- Four primitive notions: action, object, action modifiers, object modifiers
- 12 components of actions, e.g., transfer location (go), transfer relationship (give)
- Rules enable representation of meaning of sentence

## Pros and cons of conceptual dependency theory<sup>119</sup>

- *Advantages*:
  - Reduces ambiguity
  - Provides canonical form for meaning so that two sentences with same meaning have same representation
- *Disadvantages*:
  - Reduction of English to canonical form is uncomputable
  - Fails to capture subtleties

## Scripts<sup>120</sup>

- A script is a structured description of a typical situation
- *Example*: In restaurant, you are taken to a table or find a table, you obtain a menu, you order, eat, pay, and leave.
- *Scripts enable filling-in* of ambiguous or unstated events or facts
- Typical activities correspond to *roles* such as customer, waitress

<sup>114</sup> RN03, pp. 382-386

<sup>115</sup> RN03, pp. 387-402

<sup>116</sup> RN03, pp. 417-422

<sup>117</sup> Luger05, pp. 223-225

<sup>118</sup> Luger05, pp. 236-239

<sup>119</sup> Luger05

<sup>120</sup> Luger05, pp. 241-243

## Conceptual graphs<sup>121</sup>

- Finite connected graphs in which vertices are concepts or relations between concepts
- Graphs are bipartite, so that concepts only point to relations, which only point to concepts
- Dog → color → brown [pic]
- Concepts, instances, and names are distinguished
- Canonical formation rules preserve meaningfulness, e.g., ideas don't have length or mass.

## Conceptual graphs and logic<sup>122</sup>

- CGs may express modal-logic concepts such as knowledge, belief
- CGs have predicate-logic equivalent:  
[pic]  
 $(\exists x, y) \text{ dog}(x) \wedge \text{color}(x, y) \wedge \text{brown}(y)$
- CGs unlike predicate calculus have *join*, *restrict* inferencing features

## Model-based reasoning<sup>123</sup>

- *Heuristic* rule-based systems sometimes fail to take into consideration deeper causal factors such as the structure of an entity in the problem domain
- *Model-based* reasoning system: relies directly on specifications and functionality of a physical system
- *Example*: diagnosing failure of a circuit by use of knowledge of the components
- *Drawback*: models may be incomplete

## Case-based reasoning<sup>124</sup>

- This form of reasoning uses knowledge of past experience
- An alternative to general-rule heuristics
- Solutions are chosen by finding good past solutions to similar cases
- Structure:
  - Retrieve appropriate cases
  - Modify old case to fit new situation
  - Apply case
  - Save solution with record of success or failure

## Strong-method systems<sup>125</sup>

3 approaches:

- *Rule-based* using heuristics, separating knowledge from control in narrow domains, with good explanation facility
- *Model based* using structural theoretical knowledge, with advantage of robustness, good explanation features
- *Case based*, encoding past knowledge directly, with weak explanation features, good learning features
- *Hybrid approach*

## Blocks world<sup>126</sup>

- Assume a world of stackable blocks with positions
- Robot arm can grip, move, and drop blocks
- Goal is to retrieve or stack blocks
- *State*: for each block or pair,  
on-table(x)  
on(x, y)  
clear(x) if x has nothing on top of it  
gripping(x) or gripping() for none
- *Plan*: a sequence of operations that creates a path in state space to goal state

## Teleo-reactive planning<sup>127</sup>

- 1990s, Stanford
- Uses tree data structure of condition-action pairs
- Combines feedback and planning, bottom-up with top-down control
- System adapts tree with new events in environment, takes advantage of new opportunities
- Applied in control of distributed robots; particle-beam accelerator

<sup>121</sup> Luger05, pp. 248-256

<sup>122</sup> Luger05, pp. 256-258

<sup>123</sup> Luger05, pp. 298-302

<sup>124</sup> Luger05 pp. 305-310

<sup>125</sup> Luger05, pp. 310-314

<sup>126</sup> Luger05, pp. 315-317

<sup>127</sup> Luger05, pp. 323-325

## 5. The LISP language<sup>128</sup>

- A *procedural* (imperative) language developed by John McCarthy, 1950s
- Also a *functional* language based on recursive function theory
- *Basic unit*: *s*-expressions – *atoms* or *lists*, which represent both data and programs
- Elements of lists are atoms or lists (recursive structure)
- Function expressions are represented prefix as lists, e.g.,  
(*\** 3 8)
- LISP interpreter accepts expression (e.g., (*+* 2 5), displays value (7))

### Important LISP keywords

*Functions*:

*Null, list, nth, length, member, quote, eval, defun, cond, if, and, or, not, car, cdr, append, set, setq, setf, let, square, sqrt, minusp, oddp, evenp*

### Data typing in LISP<sup>129</sup>

- Data objects have type
- LISP symbols may be bound to objects of any type
- Type checking is at runtime

### Pattern matching in LISP<sup>130</sup>

```
(defun match (x1 x2)
  (cond (or (atom x1)(atom x2))
        (match-atom x1 x2)
        (t (and (match (car x1) (car x2))
                 match (cdr x1) (cdr x2))
          )
        )
  )
)
```

- Recursive algorithm is based on idea that lists match iff their heads and tails match
- *Match-atom* tests matches of nils, literals, or variables

### Unification in LISP<sup>131</sup>

- *Parameters*: Two patterns, normally at least one containing variables
- *Return value*: specification of substitutions of variables s.t. patterns match
- *Examples*:  
(unify '(p a (var x)) '(p a b) ()) [returns (((var x) . b))]  
(unify '(p a (var y)) '(p a (var x) ())) [returns (((var x) . b))  
((var y).a)]  
(unify '(p a) '(q a))() [failed]

### Meta-linguistic abstraction<sup>132</sup>

- *Definition*: Use of a language to define another language suited to solving a particular class of problems
- *Examples*:
  - Extensions of LISP by use of *defun*
- Use of Prolog to create meta-interpreter

<sup>128</sup> Luger05, pp. 723-727

<sup>129</sup> Luger05, p. 746

<sup>130</sup> Luger05, pp. 759-760

<sup>131</sup> Luger05, pp. 761ff

<sup>132</sup> Luger05, p. 767

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 4: Uncertainty and probabilistic reasoning

1. Uncertainty
2. Probability theory and belief
3. Bayes' Theorem and hidden Markov models

#### 1. Uncertainty

##### Modal logic<sup>133</sup>

- Modal operators reflect belief:
  - *Unless*:  $p(x) \rightarrow q(x)$  unless  $r(x)$
  - *Ab*:  $p(x)$  unless  $ab$   $p(x) \rightarrow q(x)$ , where  $ab$  refers to abnormal instance of  $p$ , e.g., bird with broken wing
  - $M$ :  $(\forall x)$   $good\text{-}student(x) \wedge M$   $study\text{-}hard(x) \rightarrow graduates(x)$   
where  $M$  means “is consistent with”, i.e., the fact that  $x$  studies hard is consistent with what we know

##### Truth maintenance<sup>134</sup>

- When beliefs are retracted, conclusions from them must be revised
- *Dependency-directed backtracking* allows backing up to the point where a belief changes, in order to revise conclusions
- Retracted conclusions are not necessarily false, so must be checked to see if they are supported independently

##### Truth maintenance systems<sup>135</sup>

- Truth maintenance systems work with *modal logic*
- *Varieties*:
  - *Justification based*: separates truth maintenance system from reasoning system; beliefs are supported by justifications
  - *Assumption based*: Handles multiple alternative states of belief
  - *Logic based*
  - *Multiple belief*

##### Logics based on minimal models<sup>136</sup>

- *Model*: an interpretation (set of allowable assignments/substitutions) that satisfies a formula, for all variable assignments
- *Minimal model* is the norm in describing the real world, because there are an infinite number of predicates that describe it
- *Closed world assumption*, based on minimal model of the world, is used to define certain problems

<sup>133</sup> Luger05, p. 337

<sup>134</sup> Luger05, pp. 339-340

<sup>135</sup> Luger05, pp. 340-344

<sup>136</sup> Luger05, p. 345

## Minimal model and logic<sup>137</sup>

- Under minimal model norm, if we cannot conclude  $p(x)$ , then can conclude  $\neg p(x)$
- *Example*: if a student is not in registration list, then he/she is not in the class
- Axioms implicit in use of minimal models:
  - *Unique name*: all atoms with distinct names are distinct
  - *Closed world*: only instances of a relation are those implied in clauses
  - *Domain closure*: atoms in domain are exactly those of model

## Circumscription<sup>138</sup>

- J. McCarthy, 1980
- Uses axiom scheme meta-rules to guarantee minimal predicates needed to specify problem
- *Example*: circumscribing  $is-person(A) \wedge is-person(B)$  gives  $(\forall x) is-person(x) \rightarrow x = A \vee x = B$
- Circumscription is less limiting than closed-world assumption

## Certainty factor calculation<sup>139</sup>

- Measure of belief and disbelief  $MB(H | E) / MD(H | E)$  is measure of belief (disbelief) of hypothesis  $H$  given evidence  $E$
- Certainty factor:
  - $CF(H | E) = MB(H | E) - MD(H | E)$
  - $CF(P1 \wedge P2) = \min\{CF(P1), CF(P2)\}$
  - $CF(P1 \vee P2) = \max\{CF(P1), CF(P2)\}$
  - This allows CF to be used in expert systems to label conclusions with levels of certainty
- When a result  $R$  follows from firing of two rules,  $CF(R) = CF(R1) + CR(R2) - (CF(R1) \times CF(R2))$

## Extensions to predicate logic<sup>140</sup>

- *Multi-valued*: e.g., true, false, unknown
- *Modal*: e.g., to support belief, necessity, possibility
- *Temporal*: assertion that a formula will hold always, eventually, or until another condition
- *Higher order*: quantification of predicates
- *fuzzy*

## 2. Probability theory and belief<sup>141</sup>

### Conditional probability<sup>142</sup>

- $P(a | b)$ : probability of  $a$ , given that  $b$  is all we know
- Note that this does *not* mean, “When  $b$  holds,  $P(a) = \dots$ ”
- *Independence*:  
 $a, b$  are independent iff  
 $P(a | b) = P(a)$  or  $P(b | a) = P(b)$  or  $P(a \wedge b) = P(a) P(b)$
- Independence can greatly reduce information necessary to specify a full joint distribution

### Conditional probability<sup>143</sup>

- *Unconditional* (prior) probability:  $P(E)$ , probability of an event
- *Conditional* (posterior) probability ( $E | C$ ): probability of an event given some new evidence
- *Example*: referring to “probabilistic inference” slide,  
 $P(\text{slowdown}) = 0.32$ ;  
 $P(\text{slowdown} | \text{accident}) = 0.01 + 0.16 = 0.17$
- *General rules*:  
 $P(E | C) = P(E \cap C) \div P(C)$   
 $P(E \cap C) = P(E | C) P(C)$

### Conditional independence<sup>144</sup>

- Can enable decomposition of large probability domains into weakly connected subgroups: a very important AI development
- *Pattern*:  $P(\text{cause}, \text{effect}_1, \dots, \text{effect}_n) = P(\text{cause}) \prod_i P(\text{effect}_i | \text{cause})$
- *Conditional independence*:  
 $P(\text{effect}_1 \wedge \text{effect}_2 | \text{cause}) = P(\text{effect}_1 | \text{cause}) P(\text{effect}_2 | \text{cause})$

<sup>137</sup> Luger05

<sup>138</sup> Luger05, pp. 345-346

<sup>139</sup> Luger05, pp. 350-351

<sup>140</sup> Luger05, p. 380

<sup>141</sup> RN03, pp. 466-468

<sup>142</sup> RN03, pp. 470-471, 478-479

<sup>143</sup> Luger05, pp. 178-179

<sup>144</sup> RN03, pp. 481-482

## Permutations and combinations<sup>145</sup>

- *Set*: A non-duplicating collection of items, not defined by ordering
- *Sequence*: An aggregate defined by ordering; possibly with duplication
- *Cartesian product*: A set of all ordered sequences chosen from a series of sets.  
*Example*:  $\{1, 2, 3\} \times \{1, 2\} = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)\}$
- *Permutations*: The possible orderings of elements of a set
- *Combinations*: The subsets of a set

## Events and sample spaces<sup>146</sup>

- *Atomic event* (Laplace, 1816): Occurrence that cannot be made up of other events
- *Event*: A set of atomic events
- *Sample space* (universe): Set of all possible outcomes for an event
- *Example*: Sample space for event of throwing two dice is  $\{(1, 1), (1, 2), \dots, (2, 1), \dots, (6, 6)\}$ , where each two-dice outcome is an event, and throwing a 3, or 4, etc., is an atomic event

## Probabilities of events<sup>147</sup>

- Let  $E$  be an event
- Let  $S$  be a sample space
- Then probability of  $E$  in  $S$ ,  $P(E) = |E| \div |S|$
- *Example*: Probability of throwing a 3 with two dice is  $\#\{(1, 2), (2, 1)\} \div 6^2 = 2/36 = 1/18$

## Theorems<sup>148</sup>

- For any event  $E$ ,  $P(E) \leq 1$
- For independent events  $E_1, \dots, E_n$ ,  $\sum_{k \leq n} P(E_k) = 1.0$
- Probability of *complement* of an event  $E$  is  $(1 - P(E))$
- $P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$

## Independent events<sup>149</sup>

- *Definition*:  $E_1, E_2$  are independent iff  $P(E_1 \cap E_2) = P(E_1) P(E_2)$ , i.e., each does not affect the probability of the other
- For non-independent events, notion of *conditional probability* is used, i.e., probability of  $E_1$  given  $E_2$
- *Notation*:  $P(E_1 | E_2)$

## Probabilistic inference<sup>150</sup>

- *Example*: Correlation of traffic slowdown at a location, presence of accident, and presence of construction

| Slowdown | Construction | Accident | Probability |
|----------|--------------|----------|-------------|
| t        | t            | t        | 0.01        |
| t        | t            | f        | 0.03        |
| t        | f            | t        | 0.16        |
| t        | f            | f        | 0.12        |
| f        | t            | t        | 0.01        |
| f        | t            | f        | 0.05        |
| f        | f            | t        | 0.01        |
| f        | f            | f        | 0.01        |

- *Meaning*:  $S$ ,  $C$ , and  $A$  were all present 1% of the time
- $P(S) = 0.01 + 0.03 + 0.16 + 0.12 = 0.32$

## Random variables<sup>151</sup>

- *Definition*: A random variable is a function  $f: S \rightarrow \mathbf{R}$  where a probability is assigned to each outcome in the sample space
- *Kinds*: Boolean, discrete, continuous
- A random variable is a distribution that describes the likelihood of outcomes
- *Example*: random variable for throw of two dice:

|   |   |    |   |
|---|---|----|---|
| 0 | 0 | 6  | 5 |
| 1 | 0 | 7  | 6 |
| 2 | 1 | 8  | 5 |
| 3 | 2 | 9  | 4 |
| 4 | 3 | 10 | 3 |
| 5 | 4 | 11 | 2 |
|   |   | 12 | 1 |

<sup>145</sup> Luger05, pp. 168-169

<sup>146</sup> Luger05, p. 171

<sup>147</sup> Luger05, p. 171

<sup>148</sup> Luger05, p. 172

<sup>149</sup> Luger05, p. 173

<sup>150</sup> Luger05, pp. 174-176

<sup>151</sup> Luger05, pp. 176-

### Chain rule<sup>152</sup>

- $P(A \cap B) = P(A | B) P(B)$
- Probability of events  $A$  and  $B$  both occurring is probability of  $A$  given  $B$  multiplied by the probability of  $B$
- This generalizes to the chain rule:  

$$P(A_1 \cap A_2 \cap \dots \cap A_n) = P(A_1) P(A_2 | A_1) P(A_3 | A_1 \cap A_2) \dots P(A_n | \bigcap_{i=1..n-1} A_i)$$

### Stochastic methods<sup>153</sup>

- With stochastic methods we estimate a most likely cause or a course of action that is most likely to succeed
- To do this, we must compute *probabilities*
- Probability quantifies likelihood of events

### Application of stochastic methods<sup>154</sup>

Some applications:

- *Diagnostic reasoning*, because cause-effect relationship is not always obvious
- *Natural language processing*, because semantics are fuzzy or ambiguous
- *Planning*, because of uncertainty of future events and cause-effect relationships
- *Learning*, because conclusions to draw from experience are ambiguous and probabilistic

## 3. Bayes' Theorem and hidden Markov models

### Maximum a posteriori value for an hypothesis<sup>155</sup>

- This is the value of  $h_i$  that maximizes  $P(E | h_i) P(h_i)$ , the maximum likelihood hypothesis that might explain an event:  

$$\operatorname{argmax}(h_i) P(E | h_i) P(h_i)$$
- It is useful for diagnostic reasoning as the best hypothesis to explain evidence  $E$

### Bayes' theorem<sup>156</sup>

- Suppose hypotheses  $h_i$  partition the sample space
- Then  $P(E) = \sum_i P(E | h_i) P(h_i)$
- $\therefore P(E) = \sum_i P(E \cap h_i)$
- *Theorem* (Bayes):  

$$P(h_i | E) = P(E | h_i) P(h_i) \div \sum_i P(E | h_i) P(h_i)$$
- Gives probability of a hypothesis given a piece of evidence, based on probabilities of observing evidence given other hypotheses

### Applications of Bayes' Theorem<sup>157</sup>

- *Example*: Determining the likelihood of finding copper at a mining site
- Must know evidence available and probability of finding each of a set of minerals
- Must know probability of finding given evidence for each mineral found

### Requirements for using Bayes' theorem<sup>158</sup>

- Must know probabilities relating evidence to various hypotheses,  $P(E | h_i)$
- Must know probabilistic relationships among pieces of evidence
- Must know that  $H$  partitions the evidence set
- Must update probability data when new discoveries are made

### Time in Bayesian inference<sup>159</sup>

- Under the laws of classical physics, "the future determines the past in just the same way that the past determines the future"
- Since the laws of physics apply symmetrically w.r.t. time, cause may occur after effect

### Probability and philosophy/theology<sup>160</sup>

- Bayes, a minister, sought in part to refute skepticism of D. Hume which removed causality as a foundation from arguments for deism
- R. Price used Bayes' theorem, 1763, to show strong evidence for Bible miracles

<sup>152</sup> Luger05, pp. 180

<sup>153</sup> Luger05

<sup>154</sup> Luger05, p. 166

<sup>155</sup> Luger05, p. 185

<sup>156</sup> Luger05, p. 186

<sup>157</sup> Luger05, p. 186

<sup>158</sup> Luger05, pp. 186-188

<sup>159</sup> Penrose, 1989, p. 306

<sup>160</sup> Luger05, p. 190

## Plausibility measure<sup>161</sup>

- *Plausibility* (Dempster-Shafer):  $pl(p) = 1 - \text{belief}(\neg p)$
- Belief and plausibility define an interval  $\leq 1.0$
- For competing hypotheses, accumulating evidence may change belief/plausibility from no-evidence interval of  $[0, 1]$  to a different interval
- This theory takes amount of evidence into account, distinguishes lack of certainty from ignorance
- Probability theory does not give strength of belief

## Approximate inference in Bayesian nets<sup>162</sup>

- *Sampling techniques* include direct sampling, rejection sampling, and likelihood weighting
- *Markov chain simulation* is a Monte Carlo method
  - Generates events by making random changes to previous event
  - Wanders around state space conditionally flipping variable values
  - Generates estimated conditional probabilities

## Hidden Markov models<sup>163</sup>

- Let state be hidden, i.e., let observation be a probability function of current state
- *Example*: Noisy acoustic signals in speech recognition
- *Applications*:
  - Viterbi algorithm for decoding phonemes
  - Uses a table (dynamic programming) to update probability estimates
  - Algorithm starts with phoneme observations, returns most likely English spelling

## Hidden Markov models and Kalman filtering<sup>164</sup>

- A temporal probabilistic model in which the state of a process is described by one random variable
- Other variables may be added but are all combined into one random variable
- Matrix representation:  $T_{ij} = P(X_t = j \mid X_{t-1} = i)$  ( $T_{ij}$  is probability of transition from state  $i$  to state  $j$ )
- Kalman filtering seeks to use percepts, and motion-describing transition model, to estimate state of a system

## Dynamic Bayesian networks<sup>165</sup>

- DBN represents a temporal probabilistic model
- HMMs can be represented as DBNs with one state variable and one evidence variable
- DBNs may have exponentially fewer states than equivalent HMMs
- Requirements to specify a DBN:
  - Prior distribution over state variables
  - Transition model
  - Sensor model (evidence)

## Temporal probabilistic reasoning<sup>166</sup>

- Dynamic environment
- 3 models:
  - Hidden Markov models
  - Kalman filters
  - Dynamic Bayesian networks
- Form core of speech recognition systems
- *Applications*: disease, economic problems, speech understanding
- *Assumption*: stationary (unchanging) process of change

## Abductive inference: $((p \rightarrow q) \wedge q) \rightarrow p$ <sup>167</sup>

- Unsound because not always true
- But quite often applicable
- It is valid to reason this way with a certain degree of confidence
- $p \rightarrow q$  (0.9) means if  $p$  is believed to be true then it is believed that  $q$  will hold 90% of the time

<sup>161</sup> Luger05, pp. 357-359

<sup>162</sup> RN03, pp. 511-517

<sup>163</sup> Luger05, pp. 374-378

<sup>164</sup> RN03, pp. 549-551

<sup>165</sup> RN03, pp. 559ff

<sup>166</sup> RN03, pp. 537-539

<sup>167</sup> RN03, pp. 568-573

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 5: Supervised learning and natural language

1. Supervised learning
2. Connectionist learning
3. Natural language processing

#### 1. Supervised learning

##### Philosophical-epistemological issues<sup>168</sup>

- *Generalization*: recognition of invariant patterns
- *Inductive bias*: of system designer in choice of heuristics, models, algorithms
- Empiricists' dilemma

##### Symbol-based learning<sup>169</sup>

- Dimensions of classification:
  - Data and learning goals
  - Representation of learned knowledge
  - Set of operations on representations
  - Concept space
  - Choice of heuristic search
- Concept: a general description of a class of objects, e.g., in predicate calculus

##### Symbol-based learning example<sup>170</sup>

- *Example data*:  
 $\text{size}(A, \text{small}) \wedge \text{color}(A, \text{red}) \wedge \text{shape}(A, \text{round})$   
 $\text{size}(B, \text{large}) \wedge \text{color}(A, \text{red}) \wedge \text{shape}(A, \text{round})$
- *Result, denoting concept*:  
 $\text{size}(s, y) \wedge \text{color}(x, \text{red}) \wedge \text{shape}(x, \text{round})$
- *Generalization*: balls are round and red and of different sizes

##### Training

- *Assumption*: Distribution of training examples matches distribution of test examples
- *Learning task*: Generalize from training examples (or past experience) to state-perspect action function

##### Complexity and guidelines for logic-based abduction<sup>171</sup>

- Exponential number of solutions makes abduction intractable
- Even to find one solution may be NP-hard
- *Rule*: prefer hypothesis sets that are minimal, prefer simple explanations

##### Decision tree induction learning<sup>172</sup>

- Represents concepts as decision trees
- Example: convert an example table listing credit history, debt, collateral, income, risk, into a decision tree giving risk in cases of different levels of debt, collateral, income
- ID3 algorithm assumes that best tree for correctly classifying new example instances is simplest: Occam's Razor
- Decision trees can straightforwardly be converted to rule sets: convert each path through tree into a rule, with decisions leading to leaves being the left side of rules

<sup>168</sup> Luger05

<sup>169</sup> Luger05, pp. 391-392

<sup>170</sup> Luger05, p. 392

<sup>171</sup> Luger05, pp. 348-349

<sup>172</sup> Luger05, pp. 408-417

## Learnability<sup>173</sup>

- Some concepts can be easily learned; others not (e.g., randomly chosen sets of objects)
- *Learnability* may be defined in terms of a language for definition of concepts
- PAC (probably approximately correct) *learnability* (Valiant) of a concept is existence of an efficient algorithm that has high probability of finding an approximately correct concept
- *Example*:  $k$ -CNF expressions are PAC learnable

## Version space search<sup>174</sup>

- *Generalization operators*:
  - Replace constants with variables  
color( ball, red) ... color (x, red)
  - Drop conditions in conjunction  
shape(x, round)  $\wedge$  color(x, red) ... shape (x, round)
  - Add disjunction  
color (x, red) ... color (x, red)  $\vee$  color(x, blue)
  - Replace property with its class-hierarchy parent  
color(x, red) ... color (x, primary-color)
- *Inductive learning* is definable as search of the instance space for a *concept* consistent with all *training examples*
- *Version space*: the set of such concepts

## Specific-to-general version space search algorithm<sup>175</sup>

- Finds maximally specific generalization, replaces non-matching specific cases with generalizations, deletes too-general hypotheses and hypotheses that match negative instances
- Concepts  $c$  are maximally specific generalizations if they cover all positive examples, none of negative examples, and for any concept  $c'$  that covers all positive examples,  $c \leq c'$

## General-to-specific algorithm<sup>176</sup>

- Maintains a set of maximally general concepts  $c$  that cover all positive examples, no negative ones, and for any other  $c'$  that covers no negative examples,  $c \geq c'$
- Algorithm uses negative instances to reach specialization of concepts, uses positive instances to eliminate overly specialized ones

## Explanation-based learning<sup>177</sup>

- Creates general rules to cover a class of cases
- Explaining value of an idea is easier than inventing it
- First uses prior knowledge, then defines cases where explanation applies
- Hypothesis spaces are superexponential in size; number of examples required is exponential in number of features of an example

## Explanation based learning<sup>178</sup>

- Generalizes from explanations, not from instances
- Starts with:
  - Target concept
  - Training example
  - Domain theory: rules and facts that explain how example is instance of concept and allows learner to select relevant aspects of a training example
  - Operability criteria describing form of concept definition

## Advantages and weaknesses of EBL<sup>179</sup>

### Advantages:

- A single training example is sufficient
- Only generalizations relevant to specific goals are made

### Weaknesses:

- All rules could be deduced from explanations without examples, hence requires starting with all information learned

<sup>173</sup> Luger05, pp. 420-422

<sup>174</sup> Luger05, p. 396

<sup>175</sup> Luger05, pp. 398-399

<sup>176</sup> Luger05, p. 399

<sup>177</sup> RN03, pp. 690-695

<sup>178</sup> Luger05, pp. 424-433

<sup>179</sup> Luger05

## Inductive logic programming<sup>180</sup>

- A rigorous approach to knowledge-based inductive learning
- Produces hypotheses that can be used in scientific work because they are understandable, unlike neural-net classifiers
- *Example:* learning *Ancestor* and *Brother-in-law* relationships from family-relationships examples
- *Constructive induction:* generation of new predicates

## Inverse resolution<sup>181</sup>

- Uses fact that  $Background \wedge Hypothesis \wedge Descriptions \models Classifications$  (examples), hence resolution can prove this entailment
- New predicates can be inferred from evidence
- By running the proof backward, a hypothesis can be found that supports the proof
- Rules of protein folding have been found by this method

## Statistical learning<sup>182</sup>

- *Example:* predict next candy's flavor (lime or cherry) based on checking a number of previous candies in a bag with certain possible percentages lime
- True hypothesis eventually dominates Bayesian prediction
- *Maximum a posteriori hypothesis* (MAP) is competitive with Bayesian learning, employs Ockham's Razor

## Parameter and Bayesian network learning<sup>183</sup>

- *Parameter learning with complete data* uses examples to derive log likelihood of certain parameter values, e.g., proportion of cherry candies in a bag based on a sample
- With complete data, this probability decomposes into separate learning probabilities
- *Bayes nets* may be learned from sample data

## Learning with hidden variables and data<sup>184</sup>

- *Hidden variables* may be learned by *expectation maximization* (EM)
- Similar to Bayes net learning, but probabilities are obtained by smoothing rather than filtering
- Bayes nets may be learned with hidden variables, with difficulty

## Instance-based learning<sup>185</sup>

- *Nonparametric learning* allows complexity of hypothesis to grow with size of example data
- *Nearest neighbor* models assume that properties of an input data item are similar to those of nearby inputs

## Prior knowledge and learning<sup>186</sup>

- *Inductive bias:* criteria used by a learner to constrain concept space or select concepts
- Example of size of class for space: for bit strings of length  $n$ , there are  $2^{2^n}$  different classifications for learning
- *Problem:* to choose plausible generalizations from limited data
- One inductive bias is to prefer simple classifications
- Syntactic bias constrains representation of learned concepts

## Analogical reasoning<sup>187</sup>

- Analogies assume that two situations similar in some ways are similar in other ways
- A kind of single-instance induction
- Not logically sound
- *Source* in analogy is a well-understood solution, example, or theory
- *Target* is one not understood
- Analogy maps source to target
- *Systemicity:* an attribute of analogical reasoning that favors use of essential rather than superficial properties of things
- *Example:* learning concept of atom from concept of solar system

<sup>180</sup> RN03, pp. 697-700

<sup>181</sup> RN03, pp. 703-706

<sup>182</sup> RN03, pp. 712-715

<sup>183</sup> RN03, pp. 716-728

<sup>184</sup> RN03, pp. 731-733

<sup>185</sup> RN03, pp. 733-736

<sup>186</sup> Luger05, pp. 417ff

<sup>187</sup> Luger05, pp. 430-431

## Candidate-elimination algorithm<sup>188</sup>

- Combines specific-to-general search with general-to-specific in bidirectional search
- Maintains maximally-general concept set, and maximally-specific, refines each until they converge
- *Advantage*: works incrementally while examples are being received; provides usable constraints on a concept as it proceeds toward it

## Neural networks

- *Human brain*:  $10^{11}$  neurons, each connected to average of  $10^4$  others. Max. switching time:  $10^{-3}$  sec
- Each neuron “fires” (output pulse) if summed inputs exceed a threshold
- Artificial NN ANVINN drove car at 70 mph, 1993
- Good for problems with noisy, complex sensory information (cameras, microphones)
- Most common algorithm: Backpropagation
- Slow learning, fast application
- Hard for humans to understand correspondence between problem and NN solution

## Unsupervised and reinforcement learning<sup>189</sup>

- No teacher
- *Model*: science
- *Categories*:
  - Discovery (e.g., number theory)
  - Conceptual clustering

## Conceptual clustering<sup>190</sup>

- Agglomerated clustering examines pairs of objects, finds similarities, defines features
- One clustering metric is the proportion of features two objects have in common
- Definitions of clusters
  - Extensional: by enumeration
  - Intensional: by definition, with advantage of representing semantics

## Neural network<sup>193</sup>

- Properties
  - Network topology
  - Learning algorithm
  - Encoding scheme
- McCulloch-Pitts model, 1943
- Gates where  $a, b \in \{0, 1\}$ , weights are 1
- [pic 2]

## Perceptrons<sup>194</sup>

- F. Rosenblatt, 1958
- Inputs, outputs are in  $\{-1, 1\}$
- Threshold function  $y =$ 

$$1 \text{ if } \sum_i w_i x_i \geq t$$

$$\text{else } -1$$
- Teacher provides correct result after perceptron generates its result
- Perceptron adjusts weights so result will go in direction of correct result

## 2. Connectionist learning<sup>191</sup>

### Artificial neuron<sup>192</sup>

- Input signals  $x_i \in \{0, 1\}$  or  $\{-1, 1\}$  etc.
- Weights  $w_i \in \mathbb{R}$
- Activation level  $= \sum_i w_i x_i$
- Threshold function  $f$
- [pic 1]

### Multi-layer feedforward nets<sup>195</sup>

- Each hidden unit operates as a perceptron; output unit combines a set of perceptron threshold functions
- Errors may be back-propagated from output, in learning
- Neural net performance competes with that of decision tree learning algorithms
- Back-propagation learning adjust weights according to errors in handling examples

<sup>188</sup> Luger05, pp. 399-403

<sup>189</sup> Luger05, pp. 433-434

<sup>190</sup> Luger05, pp. 433-434

<sup>191</sup> Luger05, 453-454

<sup>192</sup> Luger05, 455-456

<sup>193</sup> Luger05, 456

<sup>194</sup> Luger05, pp. 458-

<sup>195</sup> RN03, pp. 744-746

### Linear separability<sup>196</sup>

- Sets of positive, negative examples of a concept must be separable by a line segment on coordinate graph
- Linearly separable: AND
- [pic 3]
- Not linearly separable: XOR
- [pic 4]

### Generalized delta rule<sup>197</sup>

- Perceptron may be generalized by using other activation functions than strict hard-limiting threshold
- E.g., sigmoidal, a continuous function enabling more precise measures of error: [pic 5]
- *Delta rule* enables learner to find direction of rapidly reducing error: *gradient descent learning*, related to hill climbing

### Backpropagation learning<sup>198</sup>

- Multi-layer networks, in contrast to perceptron nets, have input, output, and possibly hidden layers
- [pic 6]
- Backpropagation algorithm assigns error easily at output layer, then computes error at hidden layers going backwards from output layer, usually using sigmoidal activation function
- Backprop works for non-linearly-separable data sets as well as linearly separable ones

### Example: neural net for XOR

- [pic 7]

### Finding clusters in 2D space<sup>199</sup>

- A neural net can learn to classify clusters of points in 2-dimensional space
- One method is to use perceptron learning with examples of elements of the two clusters
- Another is *unsupervised*, uses Kohonen backpropagation net to have network perform classification itself.
- Another method: *outstar* algorithm with counterpropagation

### Hebbian coincidence learning<sup>200</sup>

- Hebb, 1949: in nature, when a neuron contributes to the firing of another one, the connection between them tends to be strengthened
- Unsupervised Hebbian learning simulates Pavlovian conditioned response
- New stimulus can be associated with old response by presenting a new stimulus, together with old stimulus
- New stimulus elicits same response as conditioned earlier on old stimulus

### Linear associator<sup>201</sup>

- *Linear associator*: A network that associates a vector of inputs with a vector of outputs by a formula  $w = y_1x_1 + y_2x_2 + \dots + y_nx_n$
- A way to store and retrieve patterns from memory
- Each output node is given its correct output signals
- Pattern retrieved is a literal copy of the pattern

### Attractor networks of memories<sup>202</sup>

- Perceptrons, Hebbian learners, and linear associators are “feed-forward” nets in that processing goes from input to output nodes
- In feedback nets, output signals are cycled back and can return as input
- Output is state of net at convergence (cessation of change)

### Attractors and memory<sup>203</sup>

- A model for content-addressable memory
- *Attractor*: a state toward which the state of a nearby region evolves
- Patterns desired are stored as attractors
- Attractors can also solve optimization problems

<sup>196</sup> Luger05, pp. 459-460

<sup>197</sup> Luger05, pp. 464-465

<sup>198</sup> Luger05, pp. 467-468

<sup>199</sup> Luger05, pp. 477, 480

<sup>200</sup> Luger05, pp. 484-488

<sup>201</sup> Luger05, pp. 490-491

<sup>202</sup> Luger05, pp. 495-

<sup>203</sup> Luger05, pp. 495-

### Bidirectional associative memory<sup>204</sup>

- Maps from  $n$ -dimensional input vector  $x_1 \dots x_n$  to  $m$ -dimensional output vector  $y_1 \dots y_m$
- Different weights apply to each direction of each connection
- Auto associative net has connection between nodes and themselves; hence can be one-layer
- Weight matrix  $W = y_1x_1 + y_2x_2 + \dots + y_nx_n$
- Weights are calculated in advance, not learned

#### BAM example<sup>205</sup>

- Suppose we want to associate as  $(x, y)$  the pairs  $((1, -1, -1, -1), (1, 1, 1))$  and  $((-1, -1, -1, 1), (1, -1, 1))$
- Weight matrix  $W = y_1x_1 + y_2x_2 + \dots + y_nx_n$
- [pic 8]
- Can retrieve  $y$  from  $x$  or  $x$  from  $y$

#### Hopfield nets<sup>206</sup>

- Network convergence corresponds to energy minimization (stable state)
- Want guarantee that net will converge on a stable state
- Hopfield proved that for a certain class of autoassociative single layer feedback nets, there is always an energy function that guarantees convergence
- Hopfield nets have weight-updating formula and invariants  $w_{ii} = 0, w_{ij} = w_{ji}$
- Weights are calculated in advance, not learned

## 3. Natural language processing

### Levels of analysis of natural language<sup>207</sup>

- *Prosody*: rhythm and tone
- *Phonetics*: sounds that are combined as words
- *Morphology*: components of words, e.g., roots, prefixes, suffixes
- *Syntax*: grammar rules
- *Semantics*: meaning and ways to communicate it
- *Pragmatics*: effects of language constructs on listener (D'you know the time?)
- *World knowledge*: background knowledge required for understanding

### Stages of natural language processing<sup>208</sup>

- *Lexical*: isolation of words, punctuation
- *Parsing*: sentence structuring (e.g. parse tree)
- *Semantic interpretation*: internal representation of meaning (e.g., conceptual graph)
- *Contextual interpretation*: application of world knowledge, yielding extended representation

### Syntax of context-free grammars<sup>209</sup>

- Rewrite rules (productions) define language elements, e.g.,
- Nonterminals defined by sequences of nonterminals and terminals  
sentence  $\rightarrow$  noun-phrase verb-phrase  
np  $\rightarrow$  noun | article noun  
vp  $\rightarrow$  v | v np
- Nonterminals defined as terminals (literals or lexemes)  
article  $\rightarrow$  a | the  
n  $\rightarrow$  man | dog  
v  $\rightarrow$  likes | bites

### Parsing<sup>210</sup>

- Terminals and nonterminals are combined according to rules to form structures including sentences
- Parse tree [pic, p. 599]
- Transition networks may be used to diagram parsing processing

### Augmented transition networks<sup>211</sup>

- An alternative to CSLs in which constraints (e.g., verb-noun agreement) are imposed in parsing
- *Example*: "likes" has features part-of-speech = verb, root = like, number = singular
- ATN parser builds a parse tree of four-part nodes

<sup>204</sup> Luger05, pp. 496-497

<sup>205</sup> Luger05, pp. 498-499

<sup>206</sup> Luger05, pp. 500-503

<sup>207</sup> Luger05, pp. 594-595

<sup>208</sup> Luger05, pp. 596-597

<sup>209</sup> Luger05, p. 597

<sup>210</sup> Luger05, pp. 598-604

<sup>211</sup> Luger05, pp. 606-611

## Parse trees<sup>212</sup>

- [see DS slide; graphs]
- Output of parsing process is a parse tree
- Complexity of parsing CFGs is  $O(n^3)$ , worst case
- Parsing may be viewed as state-space search and logical inference
- Origin of CFGs: ancient Sanskrit linguists; reinvented by Chomsky, 1956; Backus, 1958

## Semantic analysis<sup>213</sup>

- *Case frames* specify, for verbs:
  - Linguistic relationships (agent, experiencer, instrument)
  - Constraints, e.g., only dogs bark, only people talk
  - Default values of components, e.g., teeth for “bite”
- *Conceptual graphs* define relations among concepts, e.g.:
  - *Agent* is animate object that acts
  - *Experiencer* is animate object that has state
  - *Instrument* is used in an action
- Parse trees may be transformed to give semantic representation

## Stochastic analysis of language<sup>214</sup>

- Consider a sentences as
  - A sequence of random variables, each of which is a given word in vocabulary with a certain probability; and
  - A sequence of part-of-speech tags which are also random variables
- For words  $w_1, \dots, w_n$ , choose tags  $t_1, \dots, t_n$  to maximize  $P(t_1, \dots, t_n \mid w_1, \dots, w_n)$
- If each sentence had only one correct set of tags,  $P$  would be 1.0

## Markov guided tagging<sup>215</sup>

- *Markov assumption*: The assumption that an event’s probability is dependent only on recent previous events
- Simplification of full probability equation  $\pi_{i=1 \text{ to } n} P(t_1 \mid t_{i-1}) P(w_i \mid t_i)$
- Viterbi dynamic programming algorithm can efficiently compute this
- Markov approach is 97% accurate in giving parts of speech in  $O(t^n)$  time for  $t$  different tags, sentences of size  $n$

## Stochastic methods of disambiguation<sup>216</sup>

- Most common statistical method: trigram model of word prediction (basing tag of a word on previous two words)
- *Grammarless parsing* uses mathematical models effectively
- *Mutual information checking* with decision trees makes use of patterns such as similar roles of “cat” and “kitten” in sentences

## Natural language in database querying<sup>217</sup>

- Database stores data in tables where columns are object attributes and rows are records
- Tables may be linked  
[pic]
- Table:  
[pic]
- Query:  
[pic]

<sup>212</sup> RN03, pp. 798-807

<sup>213</sup> Luger05, pp. 612-614

<sup>214</sup> Luger05, pp. 616-617

<sup>215</sup> Luger05, pp. 618-619

<sup>216</sup> Luger05, pp. 620-623

<sup>217</sup> Luger05, pp. 624-627

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 6: Adaptation and reinforcement learning

1. Interaction and intelligent behavior
2. Decision theory and expected utility
3. Partially observable Markov decision problems (POMDPs)
4. Reinforcement learning
5. Evolutionary computation

#### 1. Interaction and intelligent behavior

##### What is an environment?<sup>218</sup>

- An *environment* of a system of computing entities is a *physical* or *virtual* setting that acts as the producer of the system's inputs and consumer of its outputs.
- The environment is a participant and a memory, not just a medium for message transport
- This creates a need to elevate the environment to first-class status

##### Accessible vs. inaccessible environments<sup>219</sup>

- In accessible environments, agent percepts identify current state
- Hence if environment is stochastic, probabilities of state transitions depend only on current state, not on past history of interaction
- Markov decision problems (MDPs) are associated with stochastic problems in accessible environments

##### Intelligence may be seen as interactive<sup>220</sup>

- What kind of interaction is needed to act intelligently?
- *Example*: Mars rovers need not just to compute answers to questions, but to *adapt* to a changing environment
- *Intelligence*: “the capability of a system to *adapt* its behavior to meet its goals in a range of environments.” – *David Fogel, 'Evolutionary Computation' (2000)*
- “*Interaction* with the world is the key to *intelligence*.” – *Rodney Brooks, MIT AI lab*

##### Driving is nonalgorithmic<sup>221</sup>

- The problem of driving a car is interactive, not reducible to an algorithm
- External conditions can affect car's motion during driving

##### Mechanism design<sup>222</sup>

- A.k.a. inverse game theory; solving problems by inventing games played by rational agents, such as auctions, routing protocols allocating personnel
- *Mechanism*: language to describe strategies of agents; payoff rule
- *Auctions* involve rounds of bids by bidders, where bids interact

<sup>218</sup> D. Keil, dissertation proposal, 2006

<sup>219</sup> Idem.

<sup>220</sup> Keil, Modeling indirect interaction, 2005

<sup>221</sup> Idem.

<sup>222</sup> RN03, pp. 640-641

## 2. Decision theory and expected utility<sup>223</sup>

- Whereas goal-directed agent partitions states as good or bad, a decision-theoretic one has a continuous utility function for actions and environments
- Expected utility of an action  $A$  given evidence  $E$ :  

$$EU(A | E) = \sum_i P(\text{Result}_i(A) | \text{Do}(A), E) \cup (\text{Result}_i(A))$$
- Principle of maximum expected utility recommends choice of action that maximizes utility
- Probabilities and utilities are hard to compute

### Decision networks<sup>224</sup>

- A framework for decision making; aka *influence diagrams*
- Uses and extends Bayesian networks
- Represents information about current state, possible actions, result states, and their utilities
- [see pic, p. 598]
- Edges denote influences including influences of actions and events on events and on utility of actions

## 3. Partially observable Markov decision problems (POMDPs)

### Decision-theoretic agents for POMDPs<sup>225</sup>

- Transition and observation models are dynamic Bayesian networks
- DBN is extended to *dynamic decision network* by decision and utility nodes
- DDNs may represent POMDPs
- Deterministic belief state is used

### Overview

- POMDP: Partially observable Markov decision process
- POMDPs are problems in inaccessible stochastic environments with Markov property
- *Markov property*: holds for a system if the probability that system will be in a given state at next step depends only on current state, not on past history (L. Lipsky)

### Stochastic vs. deterministic problems<sup>226</sup>

- Transition models
- *Definition*: Set of probability values for given state transitions under given actions
- $M^a$  denotes probability of transition from state  $i$  to state  $j$  on action  $a$
- Transition model is only needed for stochastic problems

### Policy search in POMDPs<sup>227</sup>

- *Value iteration* algorithm calculates utility of each state, from which optimal actions can be computed
- *Policy iteration* chooses a policy, calculates utility of state under that policy, then repeats at predecessor states

### Value of information vs. reward value<sup>228</sup>

- In POMDPs, utility of an action may be influenced by
  - future reward being brought closer by an action
  - future percepts being made possible by an action
- Value of information obtained in future percepts must be part of a state-action pair's utility

<sup>223</sup> RN03, pp. 584-585

<sup>224</sup> RN03, pp. 597-598

<sup>225</sup> RN03, pp. 629-631

<sup>226</sup> Idem.

<sup>227</sup> RN95, pp. 502-505

<sup>228</sup> RN95, pp. 501-502

## 4. Reinforcement learning

### Model-based vs. model-free learning<sup>229</sup>

- Transition model is used in adaptive dynamic programming (ADP) learners
- Use of models is associated with knowledge-based AI
- Temporal difference (TD) learning does not use model
- Q learning is model free

### Reinforcement learning<sup>230</sup>

- RL has agent discover how to act so as to maximize a reward
- Features:
  - Trial and error search
  - Delayed reinforcement
  - Exploration-exploitation choices
- Components:
  - Policy  $\pi$ : states  $\rightarrow$  actions
  - Reward function  $r$ : states  $\times$  actions  $\rightarrow \mathbf{R}$
  - Value mapping  $V$ : states  $\rightarrow \mathbf{R}$ , where some states have high value despite low immediate reward
  - Model of environment

### RL example<sup>231</sup>

- Tic-tac-toe may be played, either with reference to idealized opponent (minimax) or actual opponent (RL)
- *Temporal difference learning* approach adjusts a table of estimated state values, starting with reward states and giving lesser values to states that precede reward states directly or indirectly
- *Update*:  $V(S_n) \leftarrow V(S_n) + c(V(S_{n+1}) - V(S_n))$

### Temporal difference (TD) learning<sup>232</sup>

- Uses observed transitions and differences between utilities of successive states to adjust utility estimates
- Update rule based on transition from state  $i$  to  $j$ :  

$$U(i) \leftarrow U(i) + \alpha(R(i) + U(j) - U(i))$$
 where
  - $U$  is utility,
  - $R$  is reward
  - $\alpha$  is learning rate

### Monte Carlo and Q learning RL methods<sup>233</sup>

- Sample sequences of states, actions, rewards are collected
- Sample returns are averaged
- Table of Q values is compiled: Q: state  $\times$  action  $\rightarrow$  value

### Q learning<sup>234</sup>

- Learns values of actions rather than utilities
- $Q(a, s)$  = value of action  $a$  in state  $s$
- $U(s) = \max_q Q(a, s)$
- Model-free
- Raises a foundational issue: is it better to learn models and utilities, or action-value functions?
- *Knowledge-based* approach assumes that knowledge of model is important

### Function approximation in RL<sup>235</sup>

- The Q function may be represented in a table with a cell for each state-action pair or the function may be approximated by generalization
- *Direct estimation* is a form of supervised learning
- *Examples*:
  - Checkers (Samuels, 1959)
  - TD-gammon (Tesauro, 1992)
  - Pole balancing (Michie, Chambers, 1968)

<sup>229</sup> D. Keil, 2004

<sup>230</sup> Luger05, pp. 442-443

<sup>231</sup> Luger05, pp. 444-447

<sup>232</sup> D. Keil, 2004

<sup>233</sup> Luger05, pp. 448-449

<sup>234</sup> RN03, pp. 775-

<sup>235</sup> RN03, pp. 777-781

### Explicit vs. implicit representation of policy<sup>236</sup>

- Learned function may be represented as table (explicit)
- Policies for games with  $10^{120}$  states have been represented as weights in a linear function of board features (implicit)
- Implicit representation allows generalization from states visited to those not

### Value function methods vs. evolutionary methods of RL

- Evolutionary methods evaluate a policy that is fixed over many test sequences
- Value function methods update policy during execution
- Value function update benefits from information obtained during interaction

### Pole Balancing: the BOXES program

- Focused on “information-versus payoff dilemma”
- Solution to complexity: reduce complex game to simpler model, then solve subgames
- Pole balancing is “game against nature”
- Infinite continuous state set is reduced by quantization to discrete state set
- No optimal policy was known in control theory

### Problem definition

- *Inputs*: either a state tuple or a failure signal
- *State*:
  - cart position (5 values)
  - angle of pole (5 values)
  - cart velocity (3 values)
  - angle rate of change (3 values)
- *Actions*: {left, right}, i.e., “bang-bang” control
- Interval from action to percept: 0.05 sec
- *Data used in learning*: For each of the 225 states, a “demon” keeps score of the following, based on past inputs:

### Policy learning overview

- “State signal” represents percept, which contributes to state knowledge
- Policy maps from states to actions
- State value is computed using accumulated experience
- Learner adapts policy online, using this experience (*LL, LU, RL, RU, target*)

## 5. Evolutionary computation in dynamic environments

### No Free Lunch<sup>237</sup>

- *Theorem*: For any function-optimization *algorithm*, for any environment (fitness function) in which the algorithm performs well...
- ...there is some environment in which it performs equally poorly
- Precisely, no function-optimization algorithm performs better in the general case than random choice
- *Result*: Function optimization usually requires knowledge of the domain, i.e., heuristics; can only achieve approximation to optimality

### Interactive computation<sup>238</sup>

- *Definition*: An *interactive computation* is an ongoing exchange of data among computing agents, such that the output of each may causally influence its later inputs
- *Definition*: A *computing agent with persistent state* (CAPS) is an agent that accepts inputs, emits outputs, and has a *state* or *memory* whose value may evolve from one I/O step to the next
- *Discussion*: It can be shown that computing agents with persistent state are capable of a wider range of behaviors than ones without persistent state

### The evolutionary algorithm revisited

- When environment *E* is dynamic, EA must be parameterized with it

<sup>236</sup> RN95, p. 615-616

<sup>237</sup> D. Keil, EC in DPEs, 2004

<sup>238</sup> D. Keil, EC in DPEs,

### No Free Lunch theorem (1996)

- No algorithmic procedure can optimize cost functions better than any other algorithmic procedure, averaged over all cost functions.
- $c$  = histogram of cost function  $f$ ;  
 $a_1, a_2$ : arbitrary function-optimization algorithms  
 $P$  = probability of histogram  
 $m$  = a sample size
- *NFLT corollary*: If a given optimizing algorithm does well on one problem, it will do poorly on another one
- *Result*: Human domain knowledge is needed for most evolutionary computation

### Resolving the NFLT paradox

- *Paradox*: Whereas by NFLT good general-purpose problem-solving *algorithms* can't exist...
- ...still, such *processes* are known to exist, such as natural evolution of life, and the scientific method
- *Solution*: NFLT applies to *algorithms* in *static* environments; does not apply to interactive learning processes occurring in dynamic persistent environments
- *Adaptation* to environments (learning of policies) is interactive, not algorithmic

### Resolving the No Free Lunch paradox<sup>239</sup>

- Whereas no general-purpose function optimizing algorithm exists, the process of evolution has provably yielded adaptive and even intelligent life
- The key to resolving the paradox is that natural evolution does not optimize (static) functions, but operates in *dynamic persistent environments*

---

<sup>239</sup> D. Keil, EC in DPEs, 2004

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 7: Distributed artificial intelligence

1. Distributed AI and emergent intelligence
2. Multi-agent systems
3. Stigmergy and self-organizing systems
4. Robotic, situated, and embodied intelligence

#### 1. Distributed AI and emergent intelligence<sup>240</sup>

- First workshop, MIT, 1980
- Earlier instances: demons, actors, blackboards
- Bolstered by work of R. Brooks, subsumption architecture, showing lack of need for centralized store of knowledge
- *Our interest*: intelligent behavior through self-organization, via indirect interaction, in multiagent systems

#### Game theory with multiple agents<sup>241</sup>

- Presence of other agents introduces uncertainty
- *Examples*: Morra (1 finger or 2); prisoner's dilemma
- *Strategy*: policy for game
- *Solution*: strategy that is rational
- *Outcome*: result of a game
- *Dominant strategy*: better than others regardless of opponent strategy

#### Dynamic systems<sup>242</sup>

- *Definition*: a system that changes over time according to formulas that relate previous and subsequent values of variables
- Systems have *state spaces* and *transitions*
- Linear systems are simple: nonlinear ones may have behavior that tends toward *attractors* – stable states
- *Chaos* is behavior very sensitive to initial conditions
- *Mind* is a dynamic system
- Mood, sleep may be better accounted for by dynamic-system view than representational one

#### Emergent intelligence<sup>243</sup>

- Rodney Brooks: intelligence emerges from interactions of a number of simple elements
- “Intelligence without representation”
- Explicit representation and models of the world “simply get in the way” (Brooks)
- Subsumption architecture decomposes system into layers of task achieving behaviors

<sup>240</sup> Luger05, p. 265

<sup>241</sup> RN03, pp. 631ff.

<sup>242</sup> Thagard05, Ch. 12

<sup>243</sup> Luger05, p. 536

## Teleo-reactive agent control<sup>244</sup>

- Nilsson et al, 1970s
- A more global architecture than subsumption architecture
- Designed for flexible performance in dynamic environments
- Continually takes into account changes in environment
- Includes planning (teleo) as well as rapid response (reactive)
- Multiple goals are maintained

## Emergent computation<sup>245</sup>

- *Definition:* appearance of global structures for processing information in systems explored by Crutchfield and Mitchell
- Research based on one-dimensional cellular automata and interaction of “particles” in CA configuration space
- Computation is said to emerge from local interactions in a spatially distributed system

## A taxonomy of environments

- Amnesic
- vs. Persistent

## Adaptation in difficult environments

- The most difficult problem environments are *persistent*, *dynamic*, and *physical*
- MASs can offer powerful *adaptive*, *flexible* solutions in such environments
- *Conjecture:* Indirect interaction provides added power in MAS solutions because of anonymity, asynchrony, space decoupling, non-intentionality

<sup>244</sup> Luger05, pp. 536-537

<sup>245</sup> Luger05, pp. 537-540

## Some interaction patterns in natural dynamic settings

- **Termites gathering chips**  
Protocol: Move at random, pick up chip when encountered, put down when another found
- **Ants foraging for food**  
Ants leave chemical trail, prefer existing trails, blaze shorter and shorter trails to and from food
- **Slime mold dividing and aggregating**  
These amoeba may aggregate by emitting chemical, migrating toward its greatest concentration

## Self-organization and emergent behavior

- *Definition:* **Self-organization** is the interaction of a set of processes or structures at a lower level of a system to yield global structures or behavior at a higher level
- *Example:* Chemical reactions
- *Contrast to:* Centralized, algorithmic behavior
- System behavior that is not the sum of component behaviors is called *emergent*

## Cellular automata<sup>246</sup>

- A CA is a grid of FSMs ( $Q, \Sigma, q_0, \delta$ ) where  $Q$  is a set of states,  $\Sigma$  an input alphabet,  $q_0$  a starting state,  $\delta: Q \times \Sigma \rightarrow Q$  a state-transition function
- States in CA may be colors of neighbors
- CA operation may be considered social learning and adaptation
- Game of Life rule: Square in grid is alive at time  $t + 1$  iff exactly 2 or 3 of its eight neighbors are alive at time  $t$

## Artificial life<sup>247</sup>

- “life made by human effort rather than by nature” (p. 532)
- Patterns emerge from interactions
- Game of Life CA (Conway) simulates some aspects of interaction of organisms
- Pix, p. 534
- Self-replicating automata were explored by Von Neumann and others, 1949-90s

<sup>246</sup> Luger05, pp. 531-532

<sup>247</sup> Luger05, pp. 532-535

## 2. Multi-agent systems<sup>248</sup>

### Features of representations<sup>249</sup>

- *Alternative schemes*: logic, rules, semantic networks, frames
- May have reasoning scheme, and centralized knowledge base, or not
- *Choice of symbols*: e.g., single words in natural language
- *Exhaustiveness*: avoids frame problem, e.g., compensating for lowering of truck bed as boxes are loaded
- Modifiability of representation
- Computational efficiency

## 3. Stigmergy

### Ubiquity of indirect interaction

- **Social biology**: Social insects interact by modifying common structures or through pheromones
- **Operating systems**: Processes communicate via *semaphores* in shared memory
- **Coordination languages**: Shared *tuple spaces* enable coordination in Linda
- **Anatomy**: Cells exchange information via *hormones* in the blood stream
- **Economics**: A *market* is an environment for buyers and sellers that serves as a medium for indirect interaction

### Properties of indirect interaction

- Time decoupling (asynchrony): State changes persist
- **Anonymity**: Recipient ID not used in access
- **Space decoupling**: Agents need not meet
- **Non-intentionality**: Agents need not have goal of communicating
- **Hybrid nature**: Physical environment may play role
- Late binding of recipient

### Bees and ants<sup>250</sup>

- Bees communicate direction and distance of pollen sources by “waggle dance”, an example of *message passing*
- Ants communicate via pheromone trails; the “message” is the entire trail followed by an ant, i.e., no single ant sends a message to another single ant
- *Conjecture*: Difference in means of communication is due to difference in foraging environments

### Food foraging problem for ants<sup>251</sup>

- Food is scattered randomly
- Task is to take it to the nest
- Ants are small and limited in intelligence and communicating power
- Food may appear or disappear dynamically
- A solution:
  - Ants walk semi-randomly dropping pheromone
  - Ants tend also to follow pheromone trails
  - Ants carrying food drop special pheromone
  - Trails evolve toward short paths between nest and food

### Stigmergy in nature<sup>252</sup>

- **Termites gathering chips into pile**: Move at random, pick up chip when encountered, put down when another chip found; the pile structure is used to coordinate creation of pile (*StarLogo*)
- **Slime mold dividing and aggregating**: These amoeba may aggregate by emitting a chemical, migrating toward its greatest concentration

<sup>248</sup> Luger05, p. 266

<sup>249</sup> Luger05, pp. 270ff

<sup>250</sup> D. Keil, Decentralization and Stigmergy, 2008

<sup>251</sup> Idem.

<sup>252</sup> Idem.

## 4. Robotic, situated, and embodied intelligence

### Object and motion processing

- *Shape* remains unchanged under transformations

- *Orientation*: 3D rotation
- *Motion*: detected by optical flow as camera moves relative to object
- *Binocular stereopsis* detects disparity of location of a feature in two images

## Artificial Intelligence: Supplementary notes

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Fall 2011, Framingham State University

### Topic 8: Future prospects and philosophical considerations

1. Theories of mind
2. Objections to weak and strong AI
3. Future prospects

#### 1. Theories of mind

##### Computer science and cognitive science<sup>253</sup>

- “Computer science will provide models and conceptual tools for the cognitive sciences ... today we are beginning the exploration of the intelligent universe of ideas, knowledge structures, and language” (John Hopcroft, Turing Lecture, 1987)

##### Mind and body<sup>254</sup>

- Duality: Descartes
- Unity: Hobbes
- Rationalism: Descartes
- Empiricism: Hume

##### Theories of mind: Buddhist<sup>255</sup>

- Doctrine of the void (Sunyavada): Nagarjuna (Ca. CE 200), or middle way: nothing exists independently of all other things
- Mahayana Buddhists (e.g., Nagarjuna) point to nonverbal experience, what is concrete or actual rather than abstract or conceptual
- *Reflection*: “If you work on your mind with your mind, how can you avoid an immense confusion?” (Seng-ts’an [d. 606])
- *Knowledge*: “Knowing is false understanding, not knowing is blind ignorance” (Chao-chou, Zen teacher, ca. 800 CE)

#### 2. Objections to weak and strong AI

##### Penrose on AI<sup>256</sup>

- Penrose believes in a transcendent, “god-given” mathematical truth that is “discovered” not “invented”
- Truth is not perceived by formal methods but by “insight,” which goes beyond algorithmic processes
- In a sense, this holds, but only in the sense that there is an *interactive, non-algorithmic, non-digital* process by which we perceive truth

#### 3. Future prospects and goals

##### Epistemological commitment<sup>257</sup>

- *Commitment* is implied by invariants
- *Problems*:
  - Possible overgeneralization from data
  - Role of inductive bias in predetermining what is learned in supervised learning
- Empiricist’s dilemma in unsupervised learning
- *One solution*: constructivist

<sup>253</sup> Luger05, p. 821

<sup>254</sup> Luger05, pp. 7-9

<sup>255</sup> A. Watts, *The Way of Zen*, Vintage, 1989, pp. 62-67, 89, 98

<sup>256</sup> Penrose89, passim

<sup>257</sup> Luger05, pp. 841-846

### Constructivist understanding<sup>258</sup>

- *Assertion:* all understanding is due to interaction between patterns of energy in the world and mental categories imposed (Piaget, 1954)
- Epistemology is of cognitive refinement and evolution through revision of schema
- Goal is agent survival
- Empiricist and rationalist traditions are combined

---

<sup>258</sup> Luger05, pp. 846-847

**CSCI 400**  
**Special Topics in Computer Science:**  
**Artificial Intelligence**

David Keil ([dkeil@framingham.edu](mailto:dkeil@framingham.edu))  
Spring 2010, Framingham State College

## References

- Sandy Fritz, Ed. *Understanding Artificial Intelligence*. Warner Books, 2002.
- George Luger. *Artificial Intelligence*. Addison Wesley, 2005.
- Tom Mitchell. *Machine learning*. McGraw-Hill, 1997.
- Hans Moravek, in Fritz, *Understanding AI*, 2002, pp. 101-114
- Michael O'Shea. *The Brain: A Very Short Introduction*. Oxford, 2005.
- Roger Penrose. *The Emperor's New Mind*. 1989.
- Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, 2<sup>nd</sup> ed. Prentice Hall, 2003.
- R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT, 1998
- Paul Thagard. *Mind: Introduction to Cognitive Science*, 2<sup>nd</sup> ed. MIT press, 2005.