Lecture 1
Introduction to AI

Kristóf Karacs
PPKE-ITK

Questions?

- What is intelligence?
- What makes it artificial?
- What can we use it for?
- How does it work? How to create it?
- How to control / repair / improve it?
- What are the consequences?
- Do we need to be afraid of it?
- What can we do?

Good to know

- Slides in English
- Vox Populi
- Requirements: later today

Administration

- Contact
  - Instructor: Kristóf Karacs
    room 231, karacs@itk.ppke.hu
  - TAs
    - Attila Stubendek, Attila Sulyok
      room 224, stubendek.attila@itk.ppke.hu
      sulyok.a.attila@gmail.com
- Web
  - http://users.itk.ppke.hu/~karacs/AI/
- Lectures
  - Mon 12:15am, Jedlik Lecture hall
- Seminars
  - Group 1: Wed 8:15am, room 322
  - Group 2: Wed 13:15pm, room 220
  - Group 3: Tue 12:15pm, room 220

What is intelligence?

intelligere: to comprehend, to perceive

- Sense
- Reason rationally
- Learn and discover
- Compete
- Communicate and cooperate

What is AI? (1)

- "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978)
- "The exciting new effort to make computers think ... machines with minds, in the full and literal sense" (Haugeland, 1985)
- "The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985)
- "The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990)
- "A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1990)
- "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1991)
- "The study of the computations that make it possible to perceive, reason, and act" (Winston, 1992)
- "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993)

Russell Beale
(University of Birmingham)

- "AI can be defined as the attempt to get real machines to behave like the ones in the movies."

John McCarthy
(Stanford)

- "It is the science and engineering of making *intelligent* machines, especially intelligent computer programs.
- It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."

Ray Kurzweil (Google)

- “Artificial intelligence is the ability to perform a task that is normally performed by natural intelligence, particularly human natural intelligence.”

Elaine Rich (University of Texas at Austin)

- “Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.”

What is AI? (2)

"The synthesis and analysis of computational agents that act intelligently."

- Science and engineering
  - *Understanding principles that make intelligent behavior possible in natural or artificial systems*
  - *Specifying methods for the design of useful, intelligent artifacts*

[Poole - Mackworth: Artificial Intelligence, Cambridge University Press, 2010]

What is AI? (3)

"Intelligence measures an agent's ability to achieve goals in a wide range of environments."

- Implicitly includes
  - *ability to learn and adapt*
  - *to understand*

[S. Legg – M. Hutter, A formal measure of machine intelligence, Benelearn Conference, 2006]

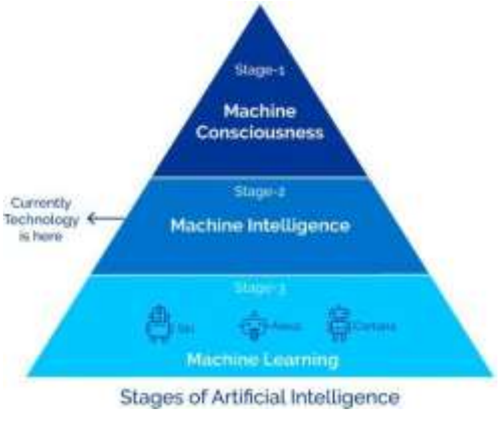
What is AI? (4)

- Study of the principles by which
  - knowledge is acquired and used,
  - goals are generated and achieved,
  - information is communicated,
  - collaboration is achieved,
  - concepts are formed,
  - languages are developed.

Intelligent agents

- act according to the circumstances and its goals
- adapt to dynamic environments and goals
- learn from experience
- are aware of their own limitations (sensors, memory, speed, etc.)

Levels of intelligence



Levels of intelligence

- Difficulty levels for humans and machines

- Playing team sports, driving a car
- Playing chess or go
- Recognizing a cat
- Solving partial differential equations
- Solving logic puzzles

Old captchas

Newer captchas

Minimum requirements

- Assignments: 50%
- Seminar tests: passing 60% of all
- Project (code and documentation): 50%
- Midterm exam: 40%

Grade composition

■ Project	30%
□ Proposal	2%
□ Code	18%
□ Documentation	10%
■ Midterm	30%
■ Final	40%

■ Activity, presentations	+ 10%
■ Competition (for top positions)	+ 20%
■ Worked out problems	+ 10%

Grading

- Grades
  - 5: 87.5%-
  - 4: 75.0%-
  - 3: 62.5%-
  - 2: 50.0%-
- Grade offer requirements
  - Min. 75% at the midterm
  - Project presentation on the last week of the semester

Presentation

- Optional
- 5 minutes
- Topics
  - Anything AI related you find interesting and think that it may be interesting to others
  - Some topics are posted on the website

Project work

- Goal: Demonstrating the use of some AI techniques
- Self defined or Challenge-type
- Proper documentation according to the rules outlined on the website
- Project submission deadlines
  - Proposal: February 27
  - First prototype: March 27
  - Final version: May 10

Project work

- Start thinking about it now, to come up with your own!

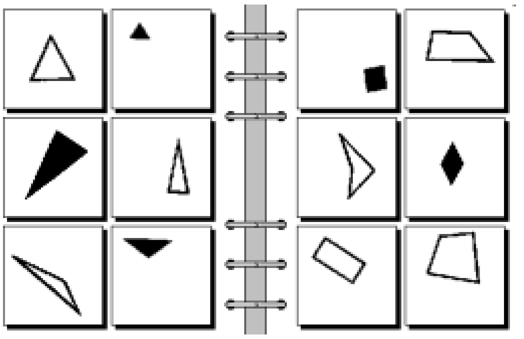
Sample project ideas

- Visual scene understanding
- Reading sheet music
- Predicting structure of protein fragments
- Object detection
- Bongard problems
- Captcha solver
- Intelligent vacuum cleaner
- Route searching for a carpooling system

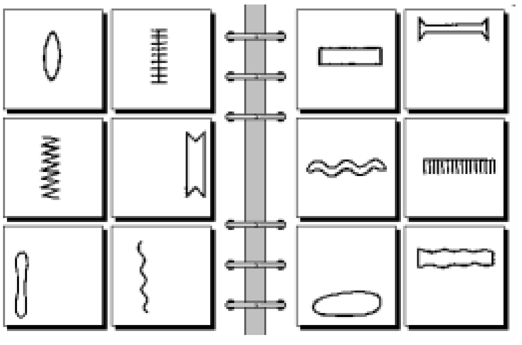
Bongard problems

- Mikhail Moiseevich Bongard, 1967
- Given 2 x 6 figures
- Task: describe what is common in one set not shared with the other set

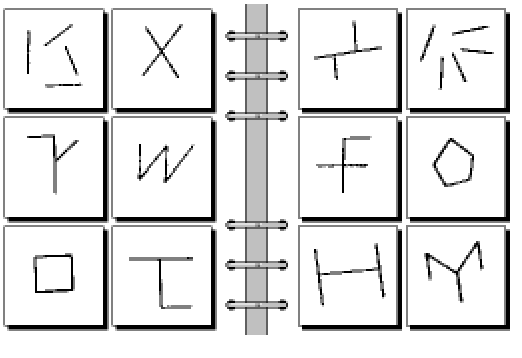
Bongard problem #6



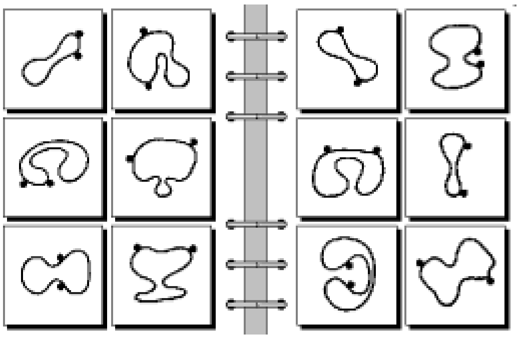
Bongard problem #7



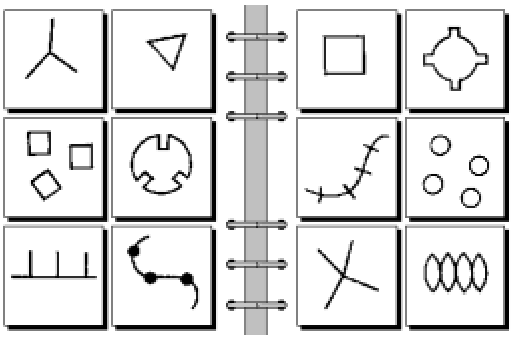
Bongard problem #87



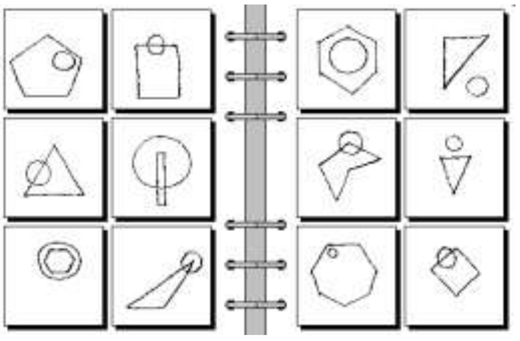
Bongard problem #20



Bongard problem #91



Bongard problem #116



Typical problems

- Exponential blow-up
- Representation of information

Methods

- Analytical
- Empirical

- Hybrid

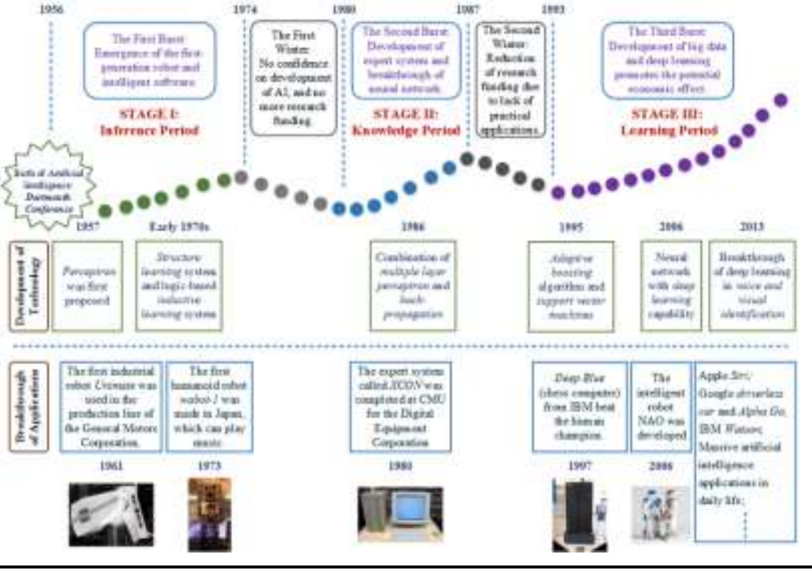
Early milestones

- 1950. Turing test
- 1955. GPS by H. Simon and A. Newell
- 1956. The term “AI” was born at a conference organized by John McCarthy in Dartmouth College, Hanover, NH

Turing Test

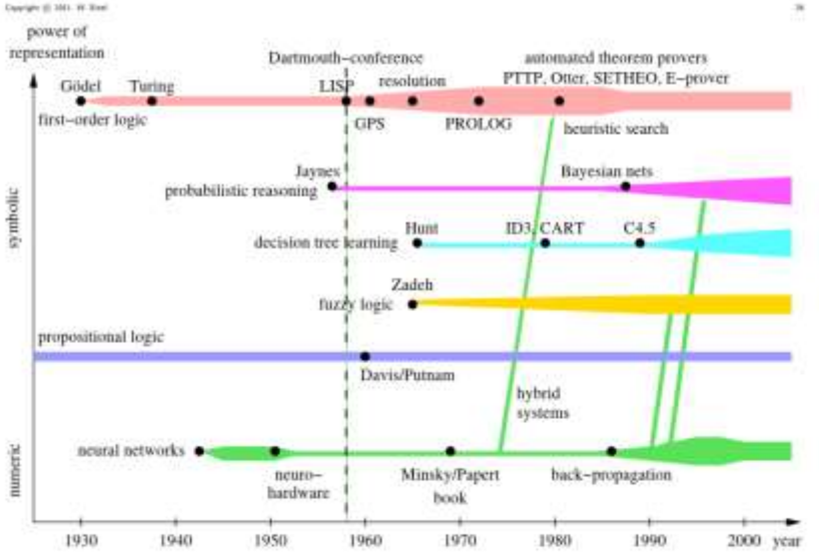
Source: Jack Copeland, alanturing.net

Stages of AI



Stages of AI

- Initial enthusiasm
- Recession
- Successes
- AI industry
- Wide-spread, sophistication



Source: Wolfgang Ertel

Source: Wolfgang Ertel

Related sciences

- Computer science / data science
  - Data mining, machine learning
- Mathematics:
  - Logic, complexity theory, probability theory
- Psychology
- Cognitive science
- Linguistics
- Biology
- Philosophy, ethics

Application areas

art, astronomy, bioinformatics, engineering, finance, fraud detection, law, mathematics, military, music, story writing, telecommunications, transportation, tutoring, video games, web search

Branches detached from AI

- Machine learning, deep learning
- Computer vision
- Speech recognition
- Optical character recognition, handwriting recognition
- Natural language processing
- Expert systems

Program

- Problem solving by search
- Search including other agents
- Logic and inference
- Search in logic representation, planning
- Inference in case of constraints
- Bayesian networks
- Fuzzy logic
- Machine learning

AI highlights (1)

- **SKICAT**: automatically classifies data from space telescopes and identifying interesting objects in the sky. 94% accuracy, way better than human *(decision trees)*
- **Deep Blue**: the first computer program to defeat human champion Garry Kasparov *(minimax search + alpha-beta-pruning + optimizations)*
- **Pegasus, Jupiter, etc.**: speech recognition systems *(Hidden Markov Models)*
- **HipNav**: a robot hip-replacement surgeon *(planning algorithms)*
- **DARPA Grand/Urban Challenge**: autonomous driving *(filtering and planning algorithms)*

AI highlights (2)

- **Deep Space 1**: NASA spacecraft that did an autonomous flyby an asteroid *(logic-based AI)*
- **Credit card fraud detection** and loan approval *(decision trees and neural networks)*
- **Chinook**: the world checker's champion *(game theory)*
- **Spam Assassin** and other spam detectors *(naïve Bayes learning)*
- **Soccer playing Aibo robots** *(reinforcement learning)*
- **Watson** *(natural language processing, knowledge aggregation)*
- **AlphaGo, AlphaZero, AlphaStar** *(deep reinforcement learning)*

Principles of academic integrity

- Projects
  - Cite all sources properly
- Assignments
  - Discuss and research the problem before you start writing
  - Do not copy cat ready solutions
  - Work on your own
  - After you start putting it into writing
    - Do not talk to others
    - Do not consult external materials

Textbooks

- S. J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Third Edition, Prentice Hall, 2009
- S. J. Russell, P. Norvig, *Mesterséges intelligencia modern megközelítésben*, második kiadás, Panem, 2005
  - available at: tankonyvtar.hu/hu/tartalom/tamop425/0026_mi_4_4
- D. Poole, A. Mackworth, *Artificial Intelligence*, Cambridge University Press, 2010
  - available at: artint.info

Other resources

- I. Futó (ed.), *Mesterséges intelligencia*, Aula, 1999
- Kevin P. Murphy, *Machine Learning – A probabilistic perspective*, MIT Press, 2012
- C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Verlag, 2006
- AAAI (Association for the Advancement of Artificial Intelligence): aaai.org
- Agent portal: agent.ai

Intelligent agents

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- What can we use it for?
- How does it work? How to create it?
- How to control / repair / improve it?
- What are the consequences?
- Do we need to be afraid of it?
- What can we do?

Do we need to be afraid of it?

- You may, but it is better to take action

  learn – know – act

Reminders

- Project: February 27, next Wednesday
- Quick presentations
- Worked out problems

Program

- Problem solving by search
- Adversarial search
- Logic and inference
- Search in logic representation, planning
- Inference in case of constraints
- Bayesian networks
- Fuzzy logic
- Machine learning

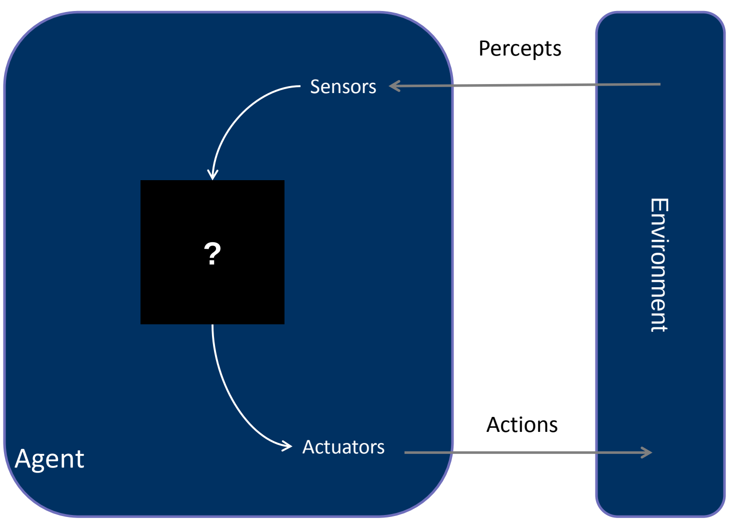
Outline

- Agents and environments
- Rationality
- PEAS: performance measure, environment, actuators, sensors
- Models of agents
- Aspects of environments

Intelligent agents

- An **agent** is anything that can be viewed as
  - *perceiving* its environment through *sensors* and
  - *acting* upon the environment through *actuators*.

How do agents work?



Type of agents

- Human

- Robot

- Software

Rational agent

- A **rational agent** is one that does the right thing

- Assessing the agent’s performance
  - Performance measure: objectively tells how successful the agent is

Evaluation of rationality

- Goals and a performance measure
- Prior knowledge about the environment
- Abilities: possible primitive actions
- History
  - Percept sequence
  - Past experiences (data to learn from)

Internal Structure

- Agent = Architecture + Program
  HW, bg. SW + *actual algorithm*
- Knowledge of Environment
  - Source
    - Given a-priori
    - Learned from sensory input
  - May include
    - Present / past states of environment
    - Influence of actions on the environment

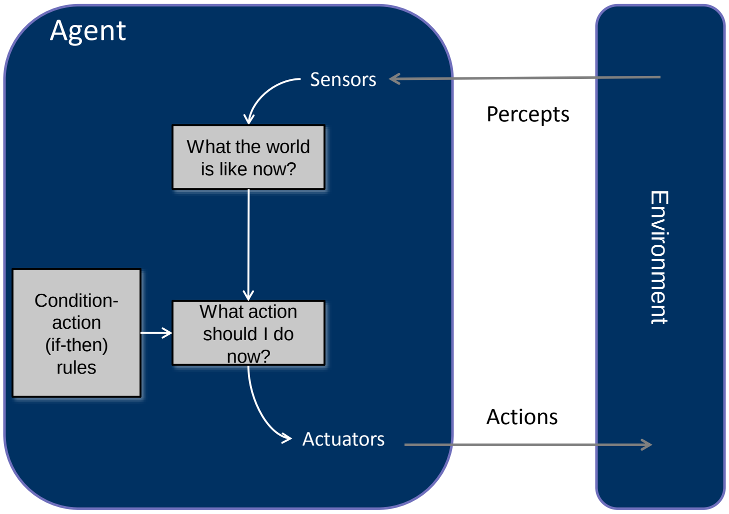
Complexity levels

- Reflex agents
  - Lookup table: if-then rules
  - Problems: size, time, flexibility
- Model-based reflex agents
  - Internal state
- Goal-based agents
  - Search and planning
- Utility-based agents
  - Non-binary measure

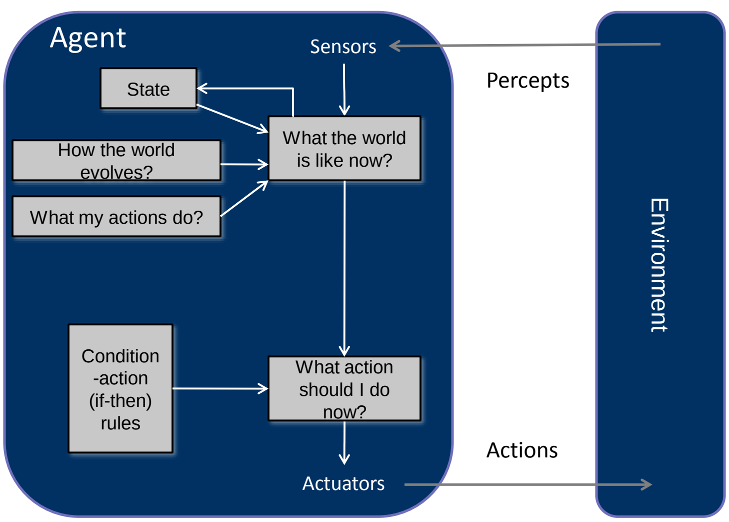
Reflexes

- Action depends only on sensory input
- Background knowledge not used
- Humans – flinching, blinking
- Chess – openings, endings
  - Lookup table (not a good idea in general)
  - 35^{100} entries required for the entire game

Reflex Agents



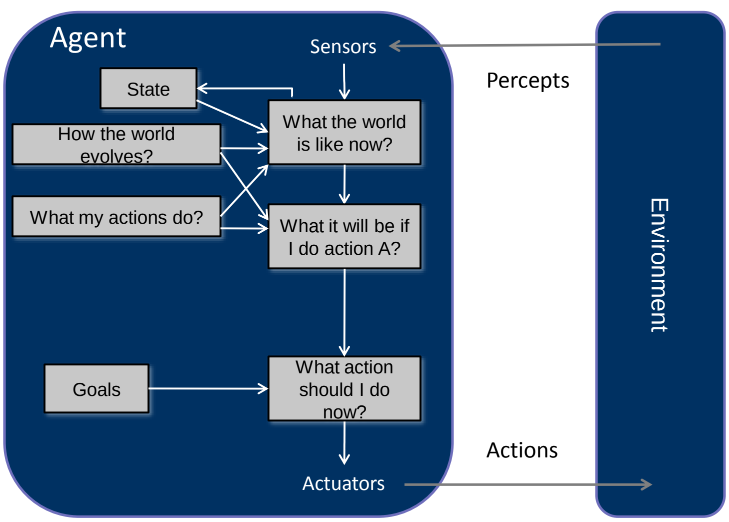
Model-based Reflex Agents



Goal of an agent

- Environment in itself is often not enough to decide what to do
- Goal is described by some properties
- A goal based agent
  - uses knowledge about a goal to guide its actions (search and planning)
  - compares the results of possible actions
- Principle: The action taken should modify the environment towards the goal

Goal-based Agents

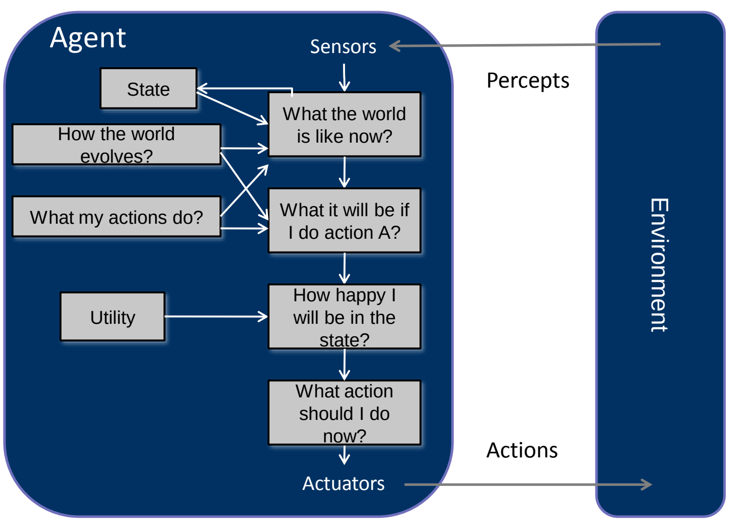


Search & Planning

Utility Functions

- Knowledge of a goal may be difficult to pin down (e.g. checkmate in chess)
- Agent may have multiple, controversial goals
- Comparing utility of states
  - Utility functions measure value of world states
  - Localized measures
- Choose action which best improves utility (Best First Search)

Utility-based Agents



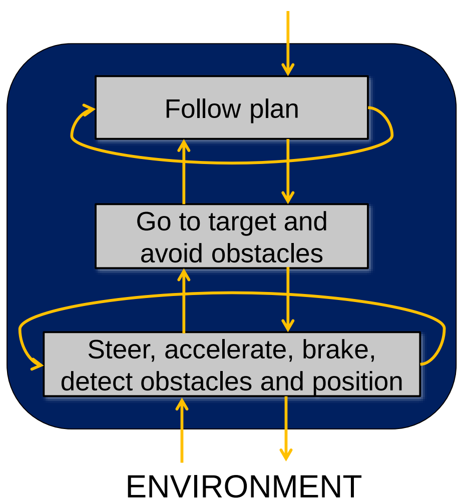
utility function: X (state space) $\rightarrow \Re$

Other aspects

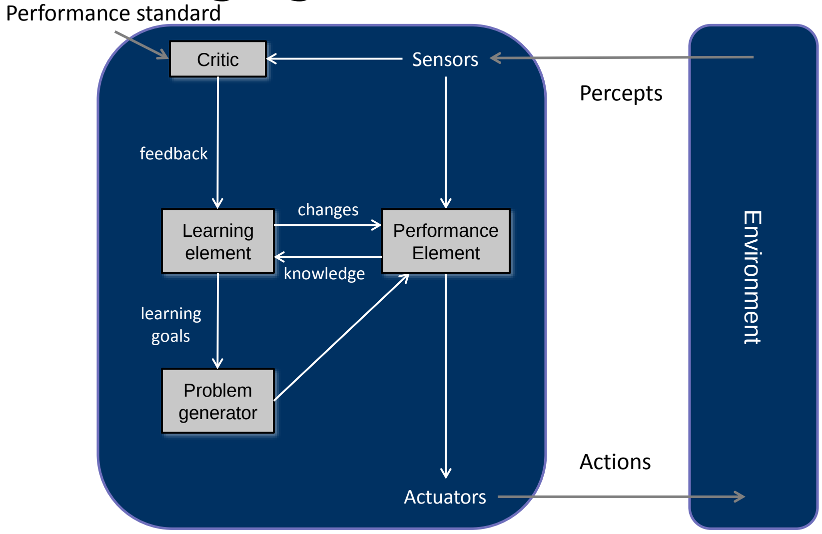
- Hybrid agents
  - Hierarchical architecture
  - Trade-off between efficiency and flexibility
- Capability of learning
- Multi-agent systems
  - Competitive vs. cooperative relationship

Hierarchical control

- Delivery robot



Learning Agents



Autonomy of Agents

- Autonomy = extent to which the agent's behaviour is determined by its own experience
- Extreme cases
  - No autonomy – ignores input (environment)
  - Complete autonomy – acts randomly/no program
- Ideal agent: some autonomy
  - Gradually increasing over time
  - Example: baby learning to crawl & navigate

Details of the Environment

- Properties of the world are different (real-world robot vs. software agent)

Uncertainty

Fully observable	vs.	Partially observable
Deterministic	vs.	Stochastic
Episodic	vs.	Sequential
Static	vs.	Dynamic
Discrete	vs.	Continuous
Single agent	vs.	Multiple agents

Observability (sensing uncertainty)

- An environment is *fully observable*, if the agent can access every information in its environment it takes into account when choosing an action
- *Partially observable* if parts of the environment are not observable
- Unobservable information must be guessed → the agent needs a model
- Example: chess (fully) vs. poker (partially)

Determinism (effect uncertainty)

- An environment is *deterministic* if a change in the world state depends only on
  - current state of the world
  - agent's action
- *Non-deterministic* environments
  - have aspects beyond the control of the agent
  - non-observable can seem to be non-det.
  - can be treated as stochastic or probabilistic
- Example: chess (det.) vs. poker (non-det.)

Episodicity

- An environment is *episodic* if the choice of current action does not depend on previous actions
- In *sequential* environments
  - Agent has to plan ahead
  - Current choice affects future actions
- Example: mail sorting system (episodic) vs. poker, chess (sequential)

Time variance

- *Static* environments don't change over time
- *Dynamic* environments: changes have to be taken into account by either
  - sensing the change
  - predicting the change
  - neglecting the change (in the short run)
- Example: poker, chess (static) vs. taxi driving (dynamic)

Continuity

- Type of sensor data and choices of action
- *Discrete*: distinct, clearly defined set
- *Continuous*: non-sectionable
- Example: chess (discrete) vs. taxi driving (continuous)

Number of agents

- *Single agent*: the environment is not changed by other actors
- *Multi-agent*: the agent is aware of other agents, who also modifying the environment
  - Modelling question: multi-agent vs. stochastic single agent
- Examples: solitaire (single) vs. poker (multi-)

Summary

- Agent: defined in connection with the environment
  - Perceives and acts
- Rationality
- Basic mode: reflex, model, goal, and utility based
  - Hierarchical control
- Learning
- Environments: observable?, deterministic?, static?, episodic?, continuous?, multi-agent?

Problem solving

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Program

- **Problem solving by search**
- Search including other agents
- Machine learning
- Logic and inference
- Search in logic representation, planning
- Inference in case of constraints
- Bayesian networks
- Fuzzy logic

Outline

- Concepts
  - State, state space, search tree, search path
  - Search strategy, solution
- Formalizing search
- Evaluation
  - Complexity, completeness, optimality, soundness
- Example
- Comparing strategies

Search and AI

- Search methods are ubiquitous in AI systems
- An autonomous robot uses search
  - to decide which actions to take and which sensing operations to perform,
  - to quickly anticipate collision,
  - to plan trajectories,
  - to interpret large numerical datasets provided by sensors into compact symbolic representations,
  - to diagnose why something did not happen as expected,
  - etc...
- Many searches may occur concurrently and sequentially

Applications

- Route finding: airline travel, networks
- Package/mail distribution
- Pipe routing, VLSI routing
- Comparison and classification of protein folds
- Pharmaceutical drug design
- Design of protein-like molecules
- Games
- Automated Theorem Proving
- Machine learning

Concepts in search

- State
- State space
- Search tree, search path
- Strategy
- Solution

Assumptions in basic search

- World is
  - static
  - discretizable
  - observable
- Actions are deterministic

- In many real world problems these assumptions do not hold →

Extended search techniques are required

Steps of problem solving

- Goal formulation
- Problem formulation
- Search
- Solution
- Execution

Formal definition of search problems

- Initial state
- Successor function: maps a state to a set of (action, successor state) pairs
- Goal test
- Action costs

Search strategy

- Decision function
  - State expansion
  - Choosing an action
- Non informed search
- Informed search

Search strategy

- The fringe is the set of all search nodes not yet expanded
- The fringe is implemented as a priority queue
  - `insert`(n, Q)
  - `remove`(Q)
- The ordering of the nodes in the queue defines the search strategy

Revisiting states

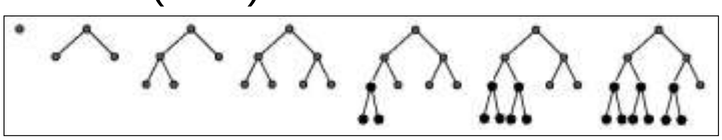
- Most search strategies have two versions
  - States may be revisited
  - States may *not* be revisited
- Implementations
  - Flag for each state
  - Visited list
- Not appropriate for all strategies

Evaluation

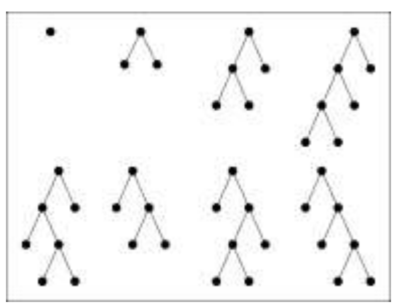
- Time and space complexity
- Completeness
  - State space
  - Pruning
- Optimality
- Soundness
  - Search for nonexistent solutions
  - Incorrect search strategy

Search strategies

- Breadth-first (BFS)



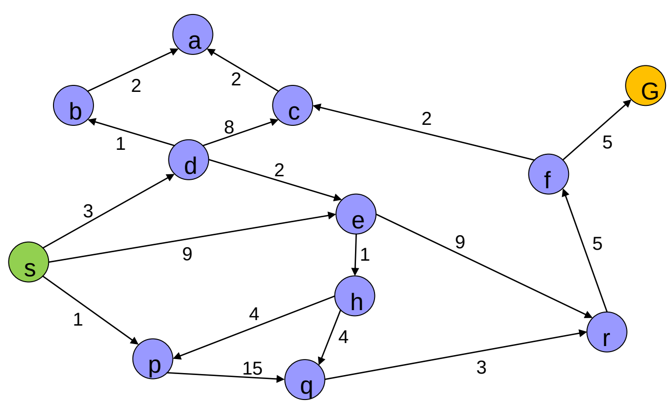
- Depth-first (DFS)



Search strategies

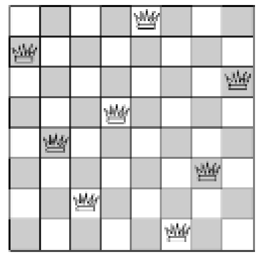
- Uniform-cost (UCS)
- Depth limited (DLS)
- Iterative deepening depth-first (IDS)
- Bidirectional (BS)

Example

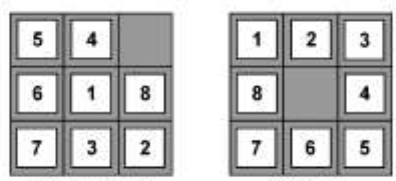


Example – 8 queens problem

- Place 8 queens on board
  - No one can “take” another



- 8-puzzle



Initial state Goal state

Implementation

- Open http://users.itk.ppke.hu/~karacs/AI/lab/search
- Download `search_demo_UI_1.html` and `search1.js` to the same folder
- Open example.html in a browser, and open the Javascript console by pressing F11, or right-click anywhere, Inspection, Console tab
  - Alternatively you can use a node.js console as well
- Function stubs are included with some coding hints
- Open `search1.js` with an editor, and implement
  - BFS
  - DFS
  - Optional: add a visited list to both algorithms
  - Optional: iterative deepening DFS
- Count the iteration steps, define an upper bound for the steps and for the size of the queue / stack and the visited list as well

Implementation

- States: arrays of numbers with fixed length (the length is given by the length of the *initialState* array)
- Goal: reach the state in which elements are sorted incrementally
  - Function goal(*state*) is already implemented, returns true or false
- State transition: Swap two elements in the array
  - Function stateTransitions(*state*) is implemented, returns an array of all states available from the *state*
- Auxiliary functions
  - isMember(*state, list*) returns true if *state* is already in *list*
  - shuffle(*array*) shuffles the instances in *array* randomly
  - log(*message*) prints *message* to the text area or to the console

Formal definition of search problems

- Initial state
- Successor function: maps a state to a set of (action, successor state) pairs
- Goal test
- Action costs

Comparison of search strategies

d: depth of shallowest solution
m: maximum depth of search tree
b: branching factor
l: depth limit

Measure	BFS	UCS	DFS	DLS	IDS	BS
Time						
Space						
Optim.						
Compl.						

Comparison of search strategies

d: depth of shallowest solution
m: maximum depth of search tree
b: branching factor
l: depth limit

Measure	BFS	UCS	DFS	DLS	IDS	BS
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optim.	Y	Y	N	N	Y	Y
Compl.	Y	Y	N	Y, if $l \geq d$	Y	Y

Summary

- Concepts
  - State, state space, search tree, search path
  - Search strategy, solution
- Formalizing search
- Evaluation
  - Complexity, completeness, optimality, soundness
- Comparison of strategies

Informed search I.

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- Agent model
- Problem solving by search
  - Non-informed search strategies
  - Informed search strategies

Program

- **Problem solving by search**
- Search including other agents
- Machine learning
- Logic and inference
- Search in logic representation, planning
- Inference in case of constraints
- Bayesian networks
- Fuzzy logic

Outline

- Best-first search
- What information is available?
- Heuristic, heuristic function
- Strategies: UCS, greedy, A*
- Properties of heuristics
- Designing heuristics
- Comparing search algorithms
- Further informed search strategies
  - IDA*, RBFS, SMA*

Search and AI

- Search methods are ubiquitous in AI systems
- An autonomous robot uses search
  - to decide which actions to take and which sensing operations to perform,
  - to quickly anticipate collision,
  - to plan trajectories,
  - to interpret large numerical datasets provided by sensors into compact symbolic representations,
  - to diagnose why something did not happen as expected,
  - etc...
- Many searches may occur concurrently and sequentially

Applications

- Search plays a key role in many applications
  - Route finding: airline travel, networks
  - Package/mail distribution
  - Pipe routing, VLSI routing
  - Comparison and classification of protein folds
  - Pharmaceutical drug design
  - Design of protein-like molecules
  - Video games

Assumptions in basic search

- World is
  - static
  - discretizable
  - observable
- Actions are deterministic

- In many real world problems these assumptions do not hold →

Extended search techniques are required

Search strategy

- The fringe is the set of all search nodes not yet expanded
- The fringe is implemented as a priority queue
  - `insert`(n, Q)
  - `remove`(Q)
- The ordering of the nodes in the queue defines the search strategy

Revisiting states

- Most search strategies have two versions
  - States may be revisited
  - States may *not* be revisited
- Implementations
  - Flag for each state
  - Visited list
- Not appropriate for all strategies

Best-first search

- Which node is good?
- *f*() : evaluation function (typically cost function)
- Selection criteria: minimal value of *f*()
- Note: “Best” does not guarantee optimality of the solution path

Properties of best-first search

- If the state space is *infinite*, then in general the search is *not complete*
- If the state space is *finite* and revisited states are *not discarded*, then in general the search is *not complete*
- If the state space is *finite* and revisited states are *discarded*, then the search is *complete*, but in general it is *not optimal*

Search algorithm

- `insert`(*initial-node*, *Q*)
- Cycle
  - If *Q* is empty then return failure
  - *n* ← `remove`(*Q*)
  - *s* ← `state`(*n*)
  - If `is-goal`(*s*) then return *s* and/or path
  - For every state *s*' in `succ`(*s*)
    - Create a node *n*' as a successor of *n*
    - `insert`(*n*', *Q*)

Using information in search

- Intelligence: situation evaluation
- Cost of an action
  - Distance in route planning
  - Power consumption
- Path cost
  - *g*() : Sum of all action costs in the path

Defeating exponential blow up

- Decreasing the number of actions in a given state (policy function)
- Decreasing search depth (value function)

- Monte Carlo tree search…

Heuristic searches

- Heuristic = Rule of thumb
  - Different from heuristic measures
  - Influences the node to expand
- Values that can help
  - Path cost *g*()
  - Heuristic measures *h*()

Heuristic Function

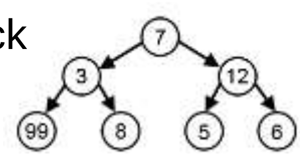
- *h*(`node`)
  - Estimates path cost of reaching the solution
  - Independent of the actual search tree
  - *h*(`goal state`) = 0
- Methods to derive a heuristic function
  - Mathematically
  - By introspection
  - Inspection of particular searches
  - Computer programs (e.g.: Absolver)
- Example: straight line distance

Uniform Cost Search (non-informed)

- ~BFS
- Expands node with smallest *g*()
  (ignores heuristic measures)
- Finds a solution with least cost
  - Condition: action costs must be positive
- Optimal and complete
- Can be very slow

Greedy Search

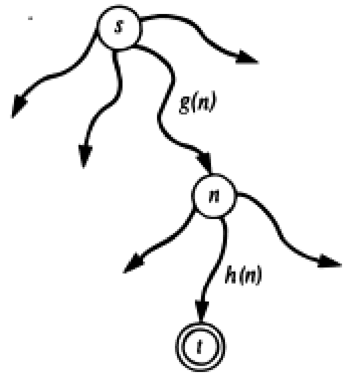
- Expand node with smallest *h*()
  (ignores path cost)
- If in a dead-end then backtrack



- Problems
  - Blind alley effect: estimates can be wrong, leading to superfluous curves
  - May lead to non-optimal solution (*h*() is only an estimate of path cost to the goal)

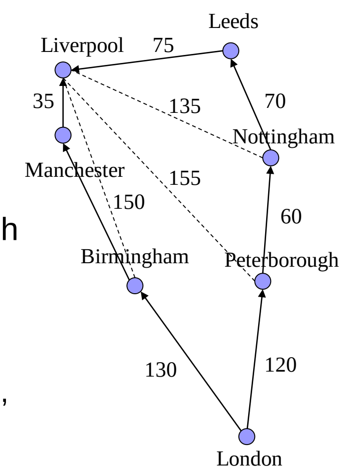
A* search

- Combines
  - uniform cost search and
  - greedy search
- f(n) estimates the cost of the best path through n
- $f(n) = g(n) + h(n)$
- Hart, Nilsson and Raphael, 1968



Example: route finding

- $g(n)$ = distance from London
- $h(n)$ = straight line distance to Liverpool
- $f(n) = g(n) + h(n)$
- 1st round: Birmingham, Peterborough
  - f(Peterborough) = 120 + 155 = 275
  - f(Birmingham) = 130 + 150 = 280
- Expands Peterborough
- Returns to Birmingham in the next step, because 120+60+135 > 130+150



Properties of heuristics

- A heuristic $h(n)$ is **admissible** if it never overestimates the path cost from node n to the goal node, i.e. $0 \leq h(n) \leq h^*(n)$.
- A heuristic $h(n)$ is **consistent** (monotone) if, for every node n and every successor n' of n generated by any action a

$$h(n) \leq c(n, a, n') + h(n').$$

- h_1 **dominates** h_2 if $h_1(n) \geq h_2(n)$.

Completeness theorem

- A* always finds an optimal solution path (even for non-admissible heuristics) if there are finitely many nodes with $f(n) \leq f^*$, f* being the cost of the optimal path. This is guaranteed if
  - all action costs $\geq \varepsilon$, for some fixed $\varepsilon > 0$, and
  - the branching degree of all nodes are finite
- Proof
  - Let f* be the cost of the optimal path
  - All nodes with $f(n) < f^*$ will get expanded
  - Some further nodes with $f(n) = f^*$ may get expanded

Optimality theorems

- If *h*(*n*) is admissible, then A* is optimal with no visited list
- If *h*(*n*) is consistent, then A* is optimal using a visited list
- If *h*(*n*) is admissible, then A* is optimally efficient: with any given heuristic no other search strategy expands fewer nodes

Dominance theorem

- If *h* and *h*' are heuristic functions and *h* dominates *h*', then any node expanded by an A* search using *h* is also expanded by A* using *h*'
- Thus using a dominant heuristic will result in fewer expanded nodes

Missionaries and cannibals

- Three missionaries and three cannibals must cross a river, using a boat that can carry at most two.
- Find a sequence of operations that ensures that cannibals never outnumber missionaries on either side of the river!

Designing heuristics

- Good heuristics can be hard to find
  - Often they are implicit in the problem, such as the Euclidean distance heuristic for route-finding
  - They may be found by relaxing some constraint in the problem
    - 8-puzzle, 15-puzzle: allow two tiles to occupy the same square
    - Missionaries: don't worry about missionaries getting eaten
- Good heuristics can be hard to compute
  - Overall goal: minimizing the total time
  - (Avg. time of computing the heuristic value + node expansion) * (total no. of nodes expanded during search)
  - Trade off between the branching factor and heuristic complexity

Comparing search algorithms

- Effective branching factor (ebf): b^*
  - Branching rate of a search tree, in which each node has the same number of outgoing edges (BFS)
- Calculation
  - d : depth of solution
  - N : number of nodes expanded

$$N = 1 + b^* + {b^*}^2 + {b^*}^3 + \cdots + {b^*}^d = \frac{{b^*}^{d+1} - 1}{b^* - 1}$$

  - Solve for b^*

Example: Effective Branching Factor

- Suppose
  - $N = 15$ steps
  - $d = 4$
- Solve: $\frac{{b^*}^{4+1} - 1}{b^* - 1} = 15$
- Result: $b^* = 1.57$

Numerical comparison

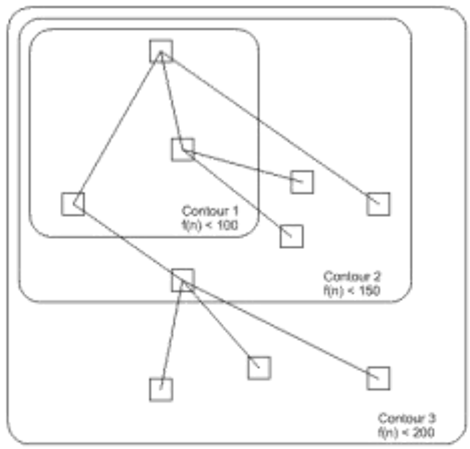
- The shortest solution for the missionaries-and-cannibals problem takes 12 steps

Search strategy	Number of steps	Effective branching factor
BFS	24,464	2.21
A* search, $h_1(x)$ = number of people still on the left bank of the river	1,202	1.67
A* search, $h_2(x)$: relaxes the requirement that cannibals not outnumber missionaries	40	1.18

Iterative-Deepening A* search

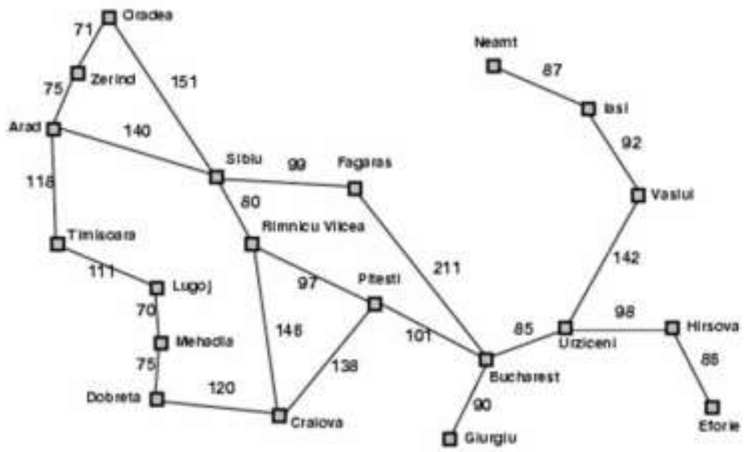
- Bottleneck of A* is *memory* (not time)
  - All visited nodes have to be recorded
- Iterative deepening like in IDS
  - Define contours based on evaluation function
  - Iteratively increase the limit
  - At each iteration use a cutoff value equal to the smallest $f(n)$ of any node that exceeded the limit in the previous iteration

IDA* search - contours



Recursive best-first search (RBFS)

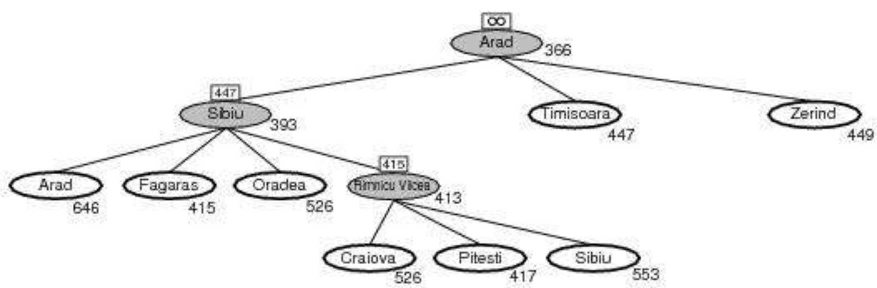
- Best-first (using *f*())
  - Stores search tree and the best alternative solution for each expanded node
  - If there is a better alternative among the nodes visited earlier → forget the current subtree and continue there
  - When recursion unwinds, replace the *f*() value of a node with best *f*() value of its children
- Requires linear space



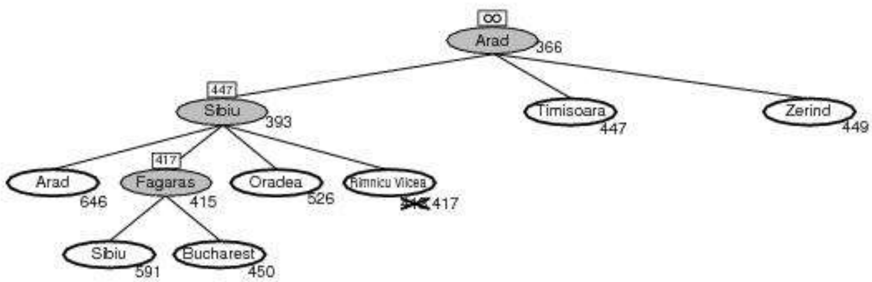
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Task: A* from Arad to Bucharest

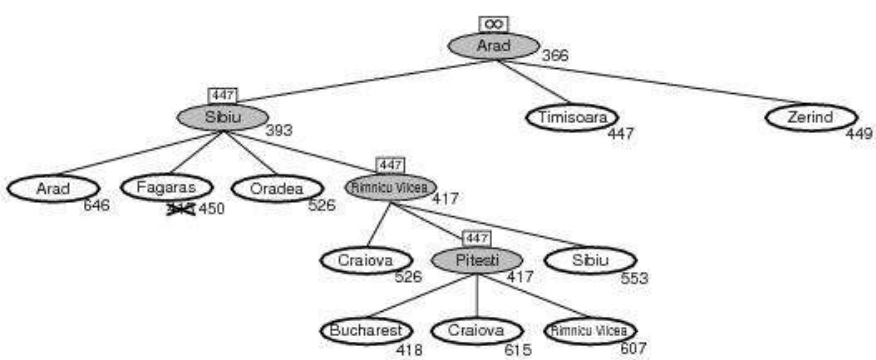
Recursive best-first search (RBFS)



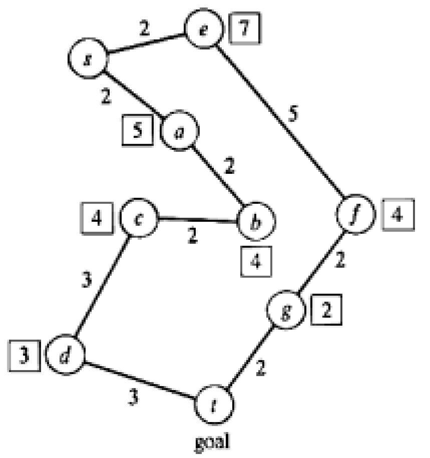
Recursive best-first search (RBFS)



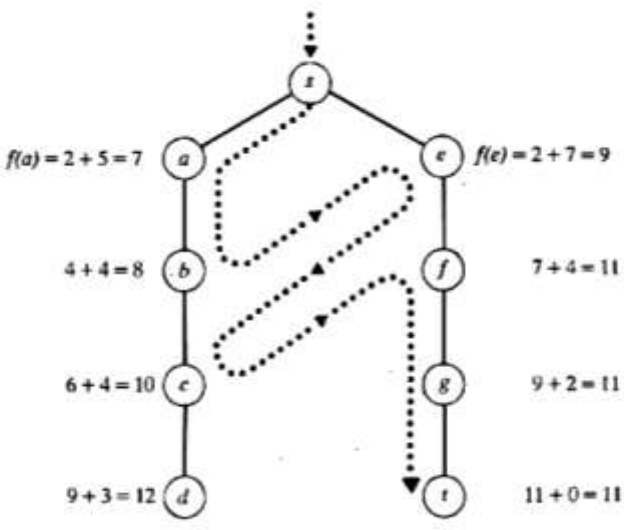
Recursive best-first search (RBFS)

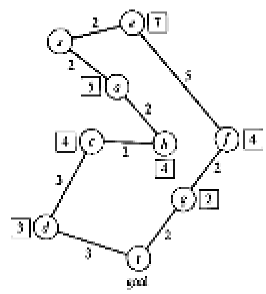


A* vs. RBFS example

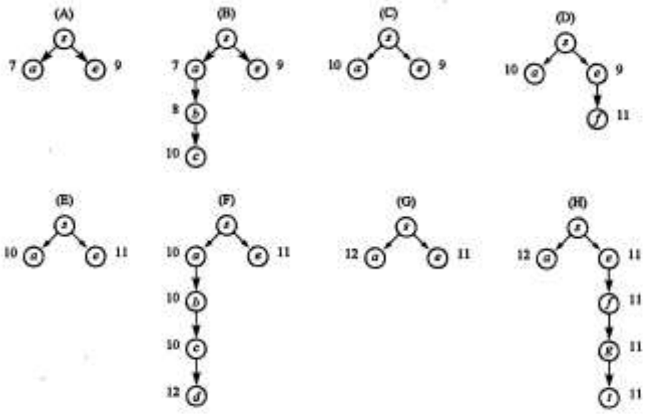


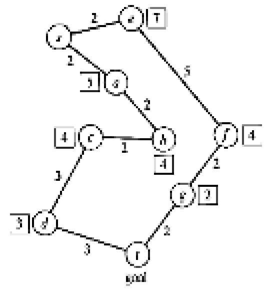
A* search





RBFS





RBFS analysis

- More efficient than IDA* and still optimal
  - Best-first Search based on next best f() contour; fewer regeneration of nodes
  - Exploit results of search at a specific f() contour by saving next f() contour associated with a node whose successors have been explored
- Like IDA* still suffers from excessive node regeneration
- IDA* and RBFS not good for graphs
  - Can't check for repeated states other than those on current path
- Both are hard to characterize in terms of expected time complexity

Simplified memory-bounded A* (SMA*)

- B: bound on memory
- If memory is full when performing A* → drop worst leaf node (with lowest *f*()) and back-up the value of the forgotten node to its parent

- If correctly parameterized, SMA* can solve more complex problems than A*

SMA* analysis

- Complete, if there is any reachable solution
- Optimal, if any optimal solution is reachable

- Problem: if B is too low → thrashing may occur (among a small set of candidate nodes)

Summary

- Assumptions and applications
- Best-first search
- Path cost, heuristic function
- Strategies: UCS, greedy, A*, IDA*, RBFS, SMA*
- Admissible, consistent, dominant heuristics
- Designing good heuristics
- Effective branching factor (comparison)

Local and online search

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- Agent model
- Problem solving by search
  - Non-informed search strategies
  - Informed search strategies

Program

- **Problem solving by search**
- Search including other agents
- Machine learning
- Logic and inference
- Search in logic representation, planning
- Inference in case of constraints
- Bayesian networks
- Fuzzy logic

Outline

- Local search algorithms
- Online search methods

Local search

- Path to solution is irrelevant
  - *n*-queens problem, circuit design, automatic layout of graphs
- Global optimization problem
  - State space: set of “complete” configurations
- Advantages
  - Low memory footprint
  - Easy implementation
  - Possibility of iterative improvement steps

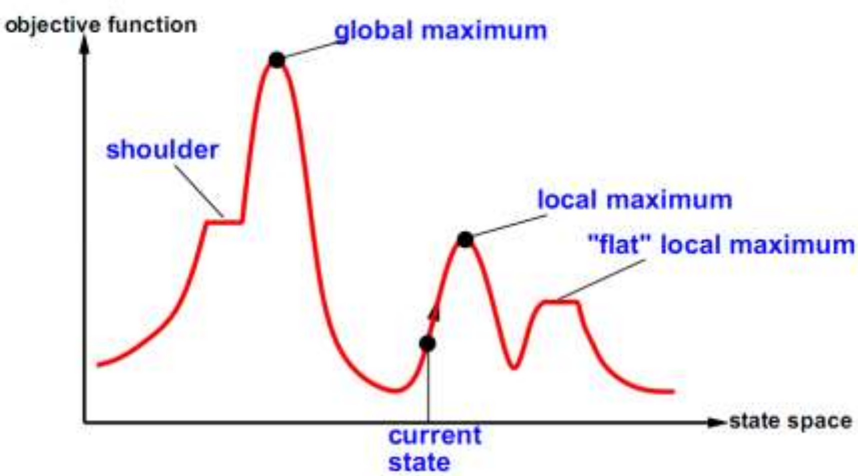
Local search algorithms

- Discrete
  - Hill climbing
  - Random walk
  - Simulated annealing
  - Local beam search
  - Genetic algorithms
- Continuous
  - Gradient

Hill climbing

- Applicable when goal is to find resulting artefacts
- *e*(): evaluation function, measures proximity
- Algorithm
  - Random initial state
  - Improve *e*() in every step
- Advantage
  - Memory requirement: one state
- Problems…

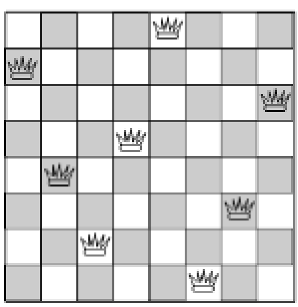
Hill climbing problems



Variants of hill climbing

- Steepest ascent
- Sideways moves
- Random restart
- Stochastic
  - probability of selection is proportional to the steepness of the surface
- First-choice
  - first state is chosen in a random follower state sequence that gives a better value

Example – 8 queens problem



- Hill Climbing:
  - put queens on randomly
  - *e*() = number of queen pairs attacking each other
  - move a queen out of other's way
  - if it's not possible, then throw queens on randomly again

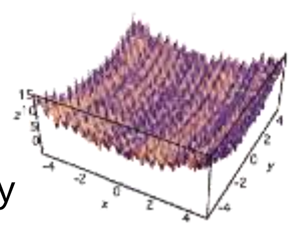
Random walk

- Randomly choose a step in an arbitrary direction

- Lattice random walk
- Gaussian random walk

Simulated annealing

- Overcomes local maxima problem
- Random step
  - If it improves, then do it!
  - If not, then do it with probability prop. to $e^{-\Delta E/T(t)}$
- $T(t)$: cooling schedule
  - T: thermic noise
  - slow cooling $\rightarrow$ global optimum
  - From Random Walk to Stochastic Hill climbing
- Question: acceptance probability



Local beam search

- Take *k* parallel threads
- Generate all follower states ($N > k$)
- Choose the *k* best states

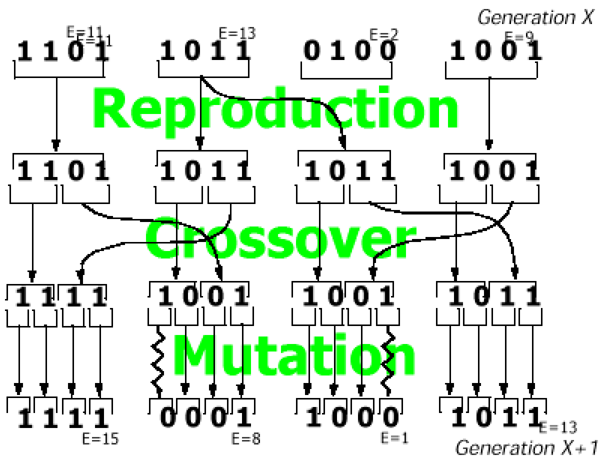
- Influence between beams
  - If one state generates several good successors, they all end up in the next iteration
  - States generating bad successors are weeded out

- Stochastic beam search

Genetic algorithms

- Population of individuals
  - basic GA: binary strings
- Goal: optimizing some function of the bit-strings
- Evaluation: Fitness function
- Start: random individuals
- Operators
  - Selection, crossover, mutation
- Stop: based on fitness threshold

Operators in GAs



[Dean, Allen, Aloimonos, AI: Theory and Practice, Benjamin Cummings, 1995]

Cycle in GAs

G_X : current generation of N bitstrings (b_0, b_1, … b_{N-1})

- For each b_i, let p_i = fitness(b_i) / Σ_j fitness(b_j).
- $G_{X+1} = \varnothing$
- For $k = 0$; $k < N/2$; $k = k+1$
  - **Select:** two parents each with probability P(parent = b_i) = p_i
  - **Crossover:** randomly swap bits in the two parents to obtain two new bitstrings
  - **Mutation:** for each bit in the new bitstrings, randomly invert it with some low probability
  - Add the two new bitstrings to G_{X+1}

Online search

- Compute, act, observe
- The agent only knows
  - Actions (s)
  - Step-cost function c(s,a,s')
  - Goal-test (s)
- Competitive ratio = actual cost / cost of shortest path
- Exploring in physical order (evident choice: DFS)
  - Backtracking also has to take place in a physical manner → actions have to be reversible
  - Worst case: every node is expanded twice
  - Solution: online iterative DFS

Online local search

- Hill climbing
  - Already online
  - Random restart version is not feasible (without teleportation)
- Adding exploration
  - Random walk
  - LRTA*: Hill climbing with memory
    - *H*(s) : maintains best estimates of cost to goal for each node
    - Assumes lowest possible cost for unexplored nodes
    - Update *H*(s) through experience
    - Demo: http://www.youtube.com/watch?v=idNr7YhAUWM

Summary

- Local search algorithms
- Online search methods

Adversarial search - Strategies in games

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- Agent model
- Problem solving by search
  - Non-informed search strategies
  - Informed search strategies

Outline

- Modeling two player games
- Game theoretic value
- Minimax search
- Cutoff search
- Pruning, alpha-beta
- Expectimax

Categorization of games

- Number of players (2 or higher)
- Competitive or cooperative
- Zero sum (game theory)
  - Total gains = Total losses
- Discrete or continuous
- Finite or infinite
- Deterministic or stochastic
- Perfect or partial information

Playing the game

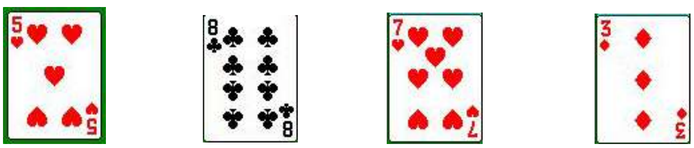
- Choosing the best move on each turn
  - Episodic search (no backtracking)
- Conventions
  - Turns alternate
  - Player 1 moves first

Search problem

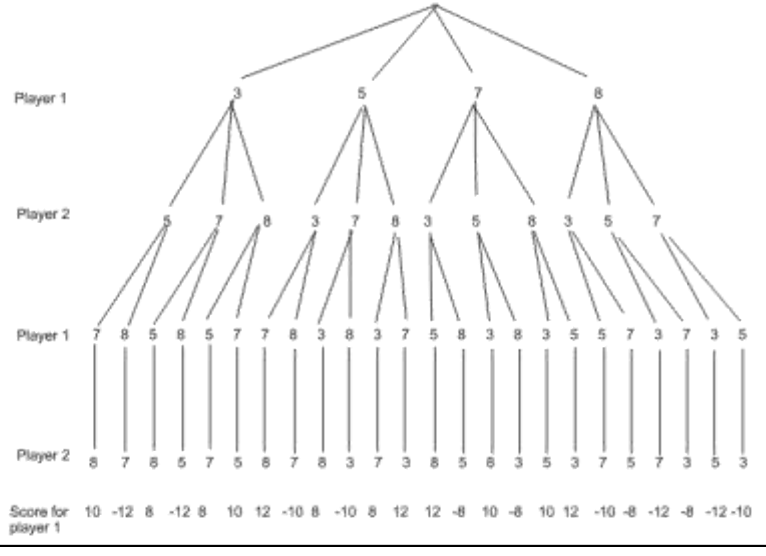
- (S, S_0, $succ()$: $S \rightarrow \mathrm{P}(S)$, F, $V()$: $\mathrm{F} \rightarrow \Upsilon$)
  - S a finite set of states (state includes player due to move)
  - S_0 initial state
  - $succ()$ follower states function
  - F terminal states
  - V value function for terminal states

Example: A trivial card game

- Deal four playing cards out, face up
- Players take cards alternating
- The player with the highest **even** sum scores the amount



Entire search space



Minimax search

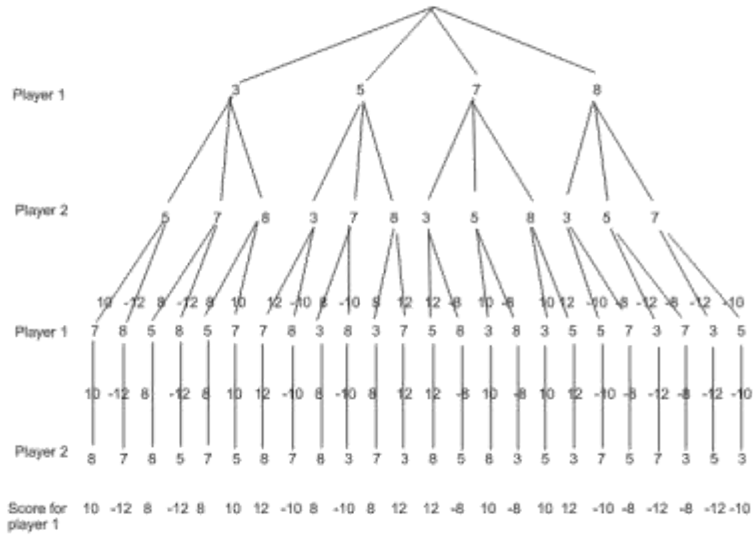
- Classic method how bandits share the gold
- Recursive search method (DFS like)
  - For own moves choose the state that maximizes the game theoretic value
  - For the moves of the opponent choose the state that minimizes the game theoretic value

Minimax algorithm

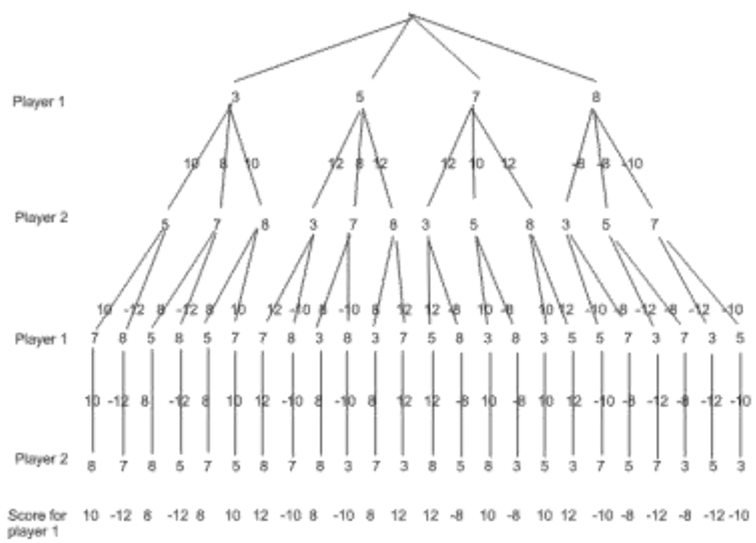
- At first assign the values associated with terminal states
- Then move the values toward the root node using minimax decision
- Game theoretic value

```
GTV(S) =
    if (S is terminal)
           return V(S)
    else
           let { S_1, S_2, … S_k } = succ(S)
           let V_i = GTV(S_i) for each i
           if (player-to-move(S) == 1)
                  return max(V_i)
           else
                  return min(V_i)
```

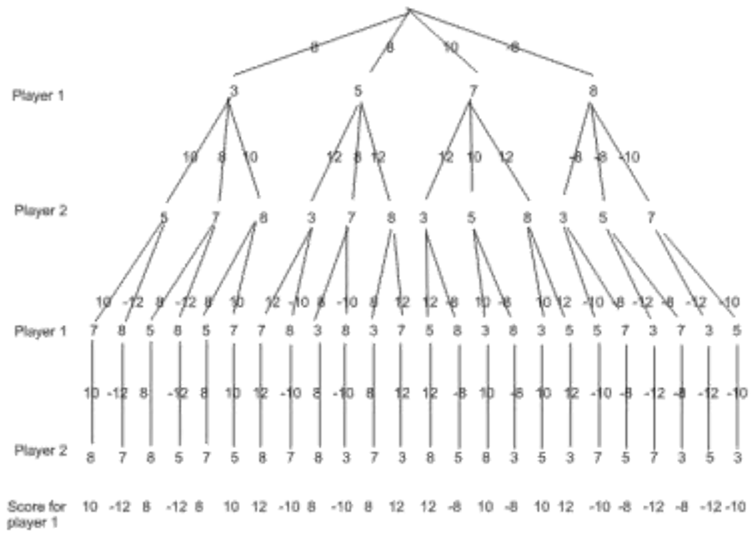
Moving the scores



Moving the scores



Moving the scores



Exercise: Nim

- There are some piles of matches
- On each turn one may remove any number of matches, but at least one from a single pile
- The last person to remove a match loses (misère game)
- In II-Nim, one begins with two piles, each with two matches

(ii ii)

II-Nim state space

(_ , _)-A	(_ , i)-A	(_ , ii)-A
(i , _)-A	(i , i)-A	(i , ii)-A
(ii , _)-A	(ii , i)-A	(ii , ii)-A
(_ , _)-B	(_ , i)-B	(_ , ii)-B
(i , _)-B	(i , i)-B	(i , ii)-B
(ii , _)-B	(ii , i)-B	(ii , ii)-B

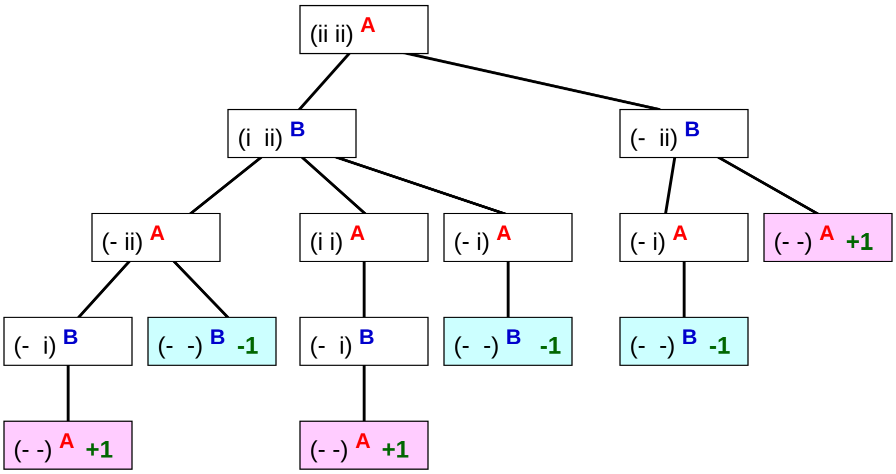
(_ , _)-A	(_ , i)-A	(_ , ii)-A
	(i , i)-A	(i , ii)-A
		(ii , ii)-A
(_ , _)-B	(_ , i)-B	(_ , ii)-B
	(i , i)-B	(i , ii)-B
		(ii , ii)-B

- Equivalent states due to symmetry (e.g. (_,ii)-A and (ii,_)-A)
- Merge them using a canonical description (e.g. left pile never larger than right)!

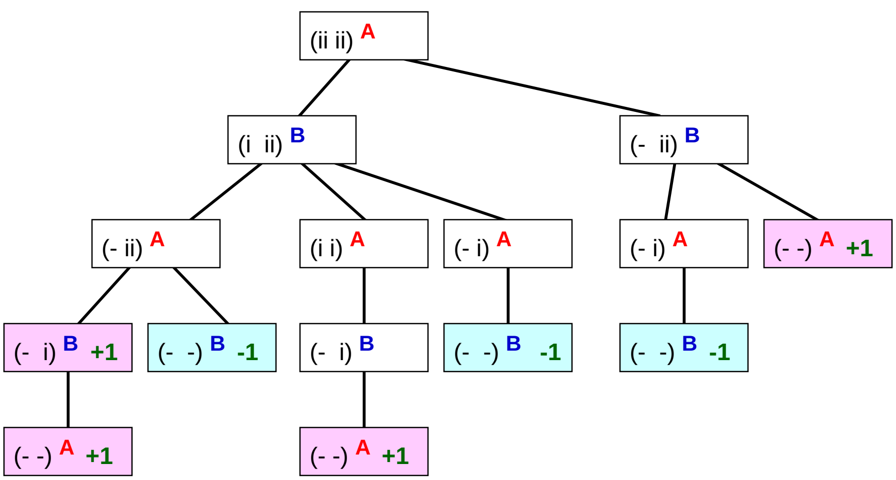
II-Nim formal definition

S	=	(_ , _)-A (_ , i)-A (_ , ii)-A (i , i)-A (i , ii)-A (ii , ii)-A (_ , _)-B (_ , i)-B (_ , ii)-B (i , i)-B (i , ii)-B (ii , ii)-B	
S_0	=	(ii , ii)-A	
succ()	=	succ(_,i)-A = { (_,_)-B } succ(_,ii)-A = { (_,_)-B , (_,i)-B } succ(i,i)-A = { (_,i)-B } succ(i,ii)-A = { (_,i)-B (_,ii)-B (i,i)-B} succ(ii,ii)-A = { (_,ii)-B , (i,ii)-B }	succ(_,i)-B = { (_,_)-A } succ(_,ii)-B = { (_,_)-A , (_,i)-A } succ(i,i)-B = { (_,i)-A } succ(i,ii)-B = { (_,i)-A , (_,ii)-A (i,i)-A } succ(ii,ii)-B = { (_,ii)-A , (i,ii)-A }
F	=	(_ , _)-A	(_ , _)-B
V	=	V(_ , _)-A = +1	V(_ , _)-B = -1

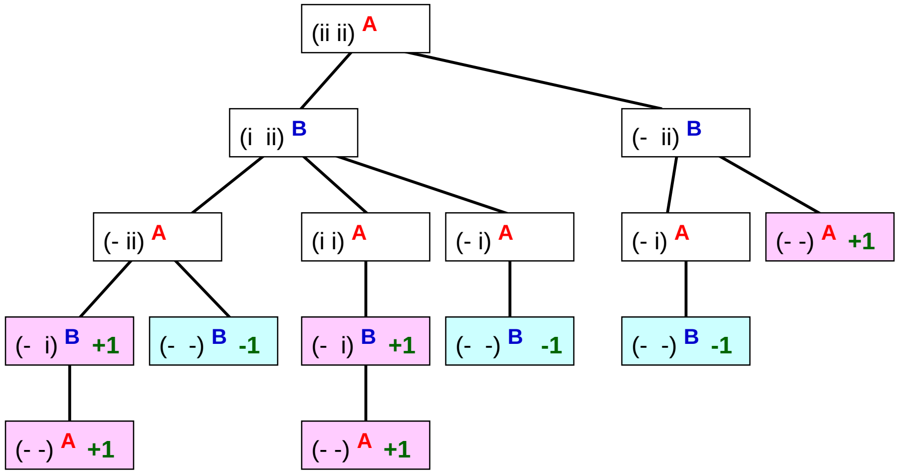
II-Nim Game Tree



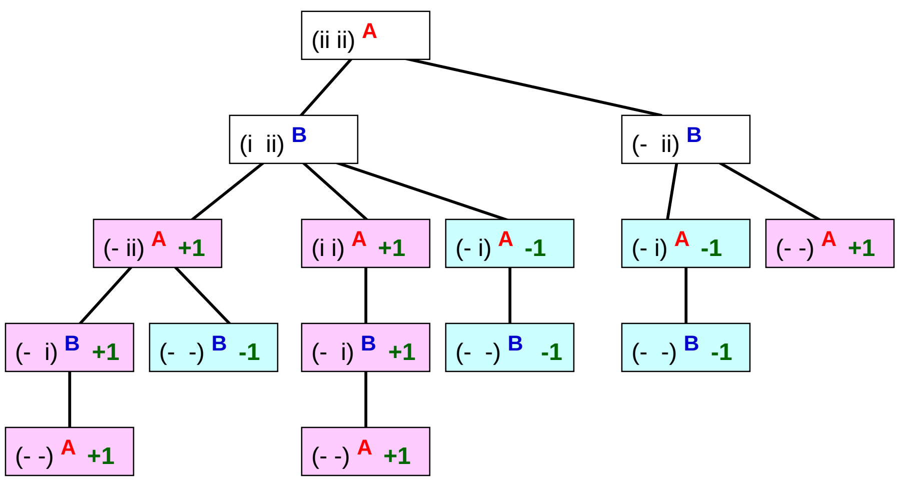
II-Nim Game Tree



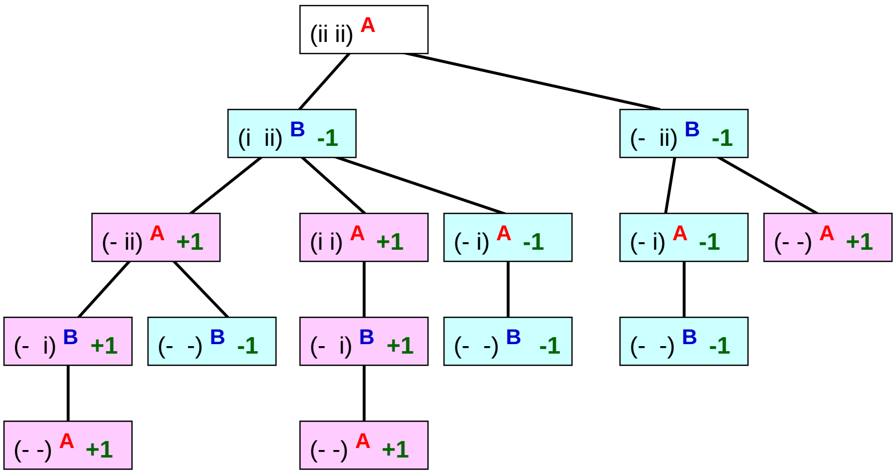
II-Nim Game Tree



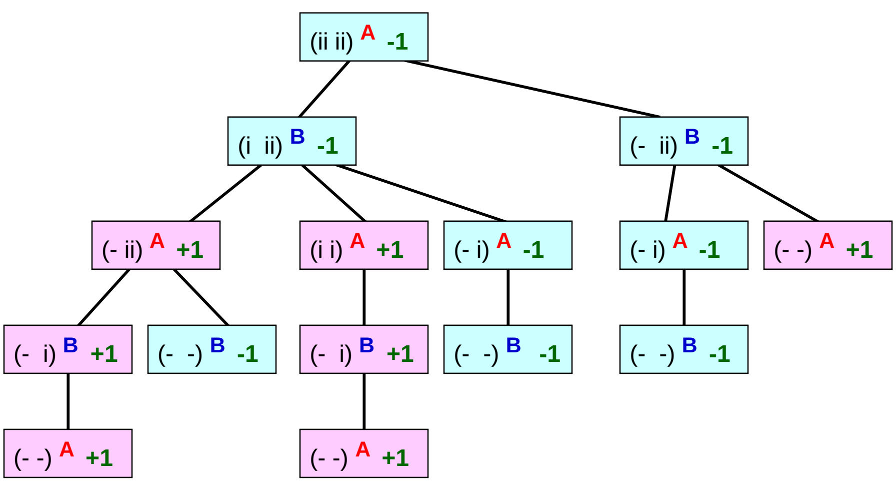
II-Nim Game Tree



II-Nim Game Tree



II-Nim Game Tree



Real games

- Search space is too large
- Real-time decision requirement
- Chess
  - Branching factor is ~35
  - Allows for a 4-ply look ahead
    - Capacity: < 2 million states per move (at 10k states/sec for 3 minutes)
    - 35^4 = 1 500 625; 35^5 = 52 521 875
  - Average humans can look ahead 6-8 plies
- Guaranteed solution not possible
- Solution: heuristic evaluation function

Cutoff search

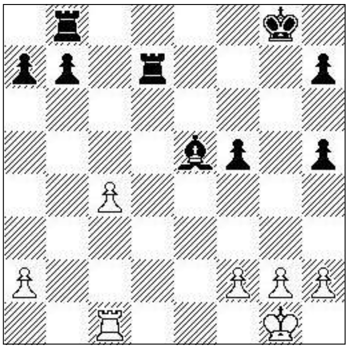
- Use an evaluation function
  - Estimate the guaranteed score
  - Draw search space to a certain depth
  - Depth chosen to limit the time taken
- Put the estimated values at the end of paths
- Propagate them to the top as before

Evaluation function

- Estimates game theoretic value of a state
- Enables comparing different states

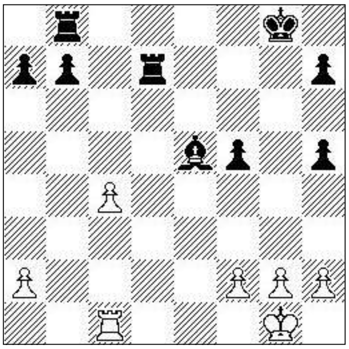
- Search + evaluation function
  - Combines many estimates → good for noise filtering

Example: Scores in chess



- Assigning weights to pieces
  - Pawn → 1
  - Knight → 3
  - Bishop → 3
  - Rook → 5
  - Queen → 9
- Position also matters in real-life evaluation functions

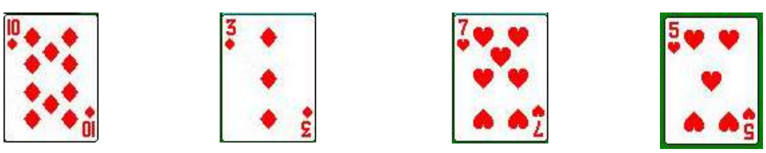
Example: Scores in chess



- Black
  - 5 pawns * 1 = 5
  - 1 bishop * 3 = 3 $\sum$ =18
  - 2 rooks * 5 = 10
- White
  - 5 pawns * 1 = 5
  - 1 rook * 5 = 5 $\sum$ =10
- Net scores
  - Black: 18-10 = 8
  - White: 10-18 = -8

Evaluation function for the example

- Odd cards: zero
- Even cards: actual value



  - In this case the evaluation function chooses 10 … which is the worst choice

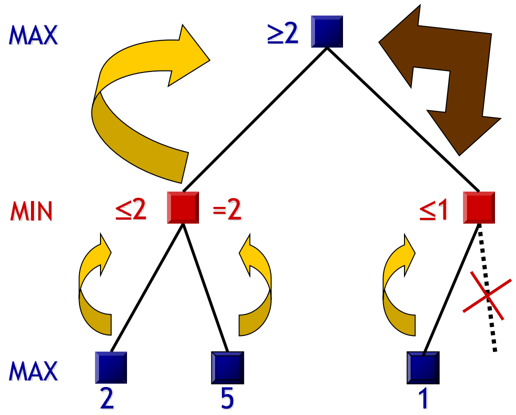
Problems with evaluation functions

- Non-quiescent states → likely to change drastically
  - Wild swings in the evaluation function
  - E.g.: captures in chess when using the sample evaluation function
  - Solution: expand the state until quiescent positions are reached
- Horizon problem
  - Good and bad possibilities in search spaces deeper than the horizon cannot be taken into account
  - Possible solution: reduce the number of initial moves to look at, thus pushing the horizon farther

Pruning

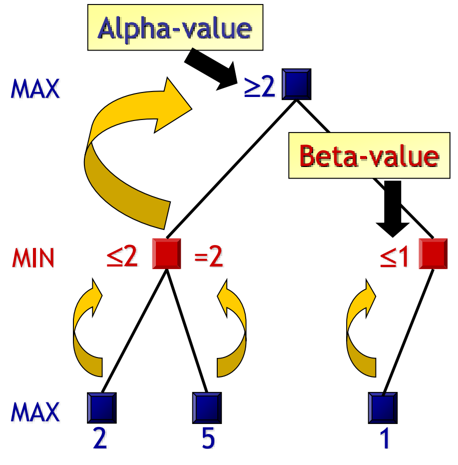
- Visit as many board states as possible
- Skip bad branches (prune them)
  - Best value is still worse than other branches
  - Example: having your queen taken in chess
- Alpha-beta pruning
  - Can be used for entire search or cutoff search
  - Recognize surely inferior branches

Idea of Alpha-Beta pruning



- The MIN-value (1) is already smaller than the MAX-value of the parent (2)
- The MIN-value can only decrease further
- The MAX-value is only allowed to increase
- No point in computing further below this node

Terminology



- Temporary values at
  - MAX-nodes are called Alpha-values
  - MIN-nodes are called Beta-values

Principles

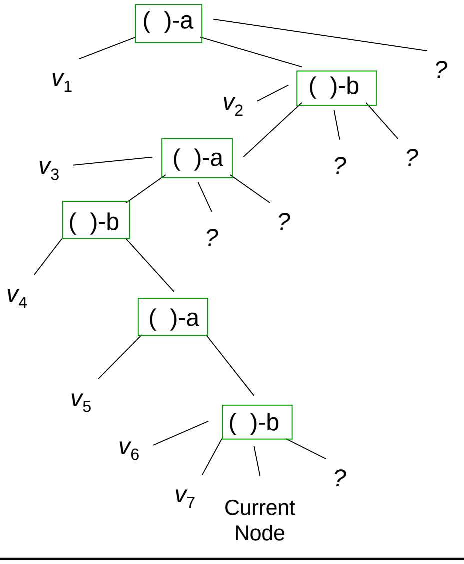
- If an Alpha-value is *greater than or equal to* the Beta-value of a descendant node, then no more children of the descendant need to be considered
- If a Beta-value is *less than or equal to* the Alpha-value of a descendant node, then no more children of the descendant need to be considered

The general cutoff rule

In example: let $\alpha = \max(v_1, v_3, v_5)$. If $\min(v_6, v_7) \leq \alpha$, then we can be certain that it is worthless searching the tree from the current node or the sibling on its right.

In general: if at a B-move node, let α = max of all A's choices expanded on current path. Let β = min of B's choices, including those at current node. Cutoff is $\beta \leq \alpha$.

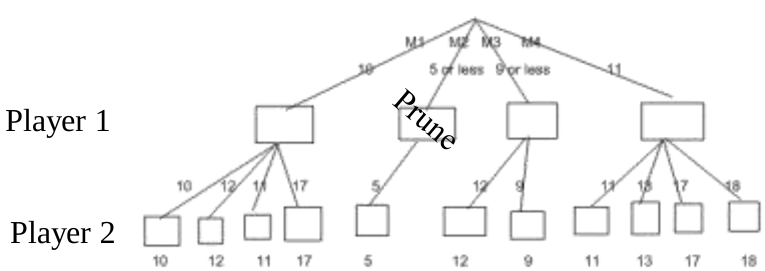
In general: Converse rule at an A-move node.



How much do we gain?

- Assuming a uniform branching factor b, minimax examines O(b^h) nodes
  - So does alpha-beta in the worst-case
- But: alpha-beta is sensitive to the order of nodes
- The gain for alpha-beta is *maximum* when
  - the MIN children of a MAX node are ordered in decreasing backed up values
  - the MAX children of a MIN node are ordered in increasing backed up values
- Then alpha-beta examines O($b^{h/2}$) nodes [Knuth and Moore, 1975]
- But this requires an oracle
- If nodes are ordered at random, then the average number of nodes examined by alpha-beta is ~O($b^{3h/4}$)

Alpha-beta pruning for the four-card game



Games with chance

- Many games have an element of chance (e.g. backgammon)
- Guaranteed scores can no longer be calculated
- Solution: calculate expected scores using probability

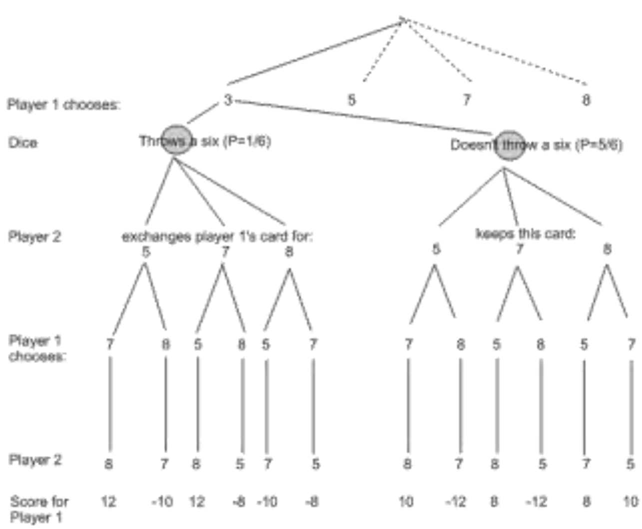
Expectimax Search

- Based on minimax tree
- For random events an extra node is added for each possible outcome that changes the possible board states after the event
- Moving score values up through a chance node
  - $E(n) = \sum p(n)*s(n)$

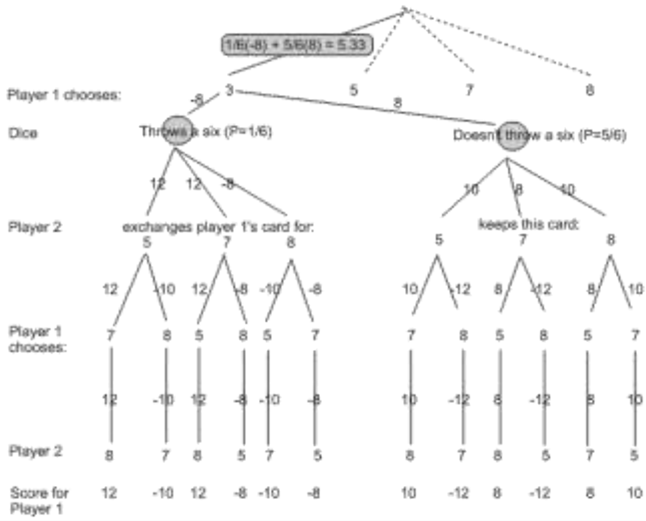
A simple game with chance

- Deal four cards face up
- Player 1 chooses a card
- Player 2 throws a die
  - If it's a 'six', then player 2 chooses a card, swaps it with player 1's and keeps player 1's card
  - If it's not a 'six', then player 2 just chooses a card
- Player 1 chooses next card
- Player 2 takes the last card

Expectimax Diagram



Expectimax Calculations



Games Played by Computer

- Games played perfectly
  - Connect four, noughts & crosses (tic-tac-toe), draughts (checkers)
  - Best move pre-calculated for each board state
    - Small number of possible board states
- Games played at superhuman level
  - Backgammon, chess, go
  - Scrabble, tetris
- Games played badly
  - Bridge, ulti, soccer :)

Game complexity

Game	State-space complexity	Game-tree complexity	Branching factor
Nine man's morris	~ 10^{10}	~ 10^{50}	10
Checkers	~ 10^{20}	~ 10^{31}	2.8
Rubik's cube	~ 10^{19}		12
Chess	~ 10^{47}	~ 10^{123}	35
Go (9x9)	~ 10^{38}		
Go (19x19)	~ 10^{171}	~ 10^{360}	250
Gomoku (15x15)	~ 10^{105}	~ 10^{70}	210

Summary

- Modeling two player games
- Game theoretic value
- Minimax search
- Cutoff search
- Pruning, alpha-beta
- Expectimax

Propositional logic

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Outline

- Logics of different order: 0, 1, 2, higher
- Basic concepts and nomenclature
  - Syntax vs. semantics
  - Entailment
- Propositional logic
- Entailment and proof methods
  - Truth table, equivalence, resolution

Logics of different order

- Propositional logic (a. k. a. Boolean logic)
  - Only constant Boolean statements
- First order predicate logic (FOPL)
  - Introduces variables, predicates, functions, and quantifiers
- Higher order logics
  - Quantifiers can also be applied to predicates and functions
  - Meta level reasoning

Logic

- A formal language in which knowledge can be expressed
- In problem solving we enumerate states
- Logic provides a means of describing set of states and carrying out reasoning
  - "Peter is hungry": refers to all world states in which Peter is hungry regardless of other things influencing the state

Basic concepts

- Syntax: specifies what expressions are legal
  - Well-formed sentences
- Semantics: meaning of sentences
  - Interpretation: assigns meaning to logic symbols
  - Semantics define the truth of sentences w. r. t. all possible interpretations
  - An interpretation *i* is a model of a set of sentences iff each of the sentences is true in interpretation *i*
- Logical inference: entailment
  - A set of sentences KB entails φ (KB ⊨ φ) iff every model of KB is also a model of φ
  - Sentence φ logically follows from KB

Syntax of propositional logic

- Atomic sentences: Propositions
  - Symbols: P, Q, R, … (uppercase letters)
  - Special cases: T (true) and F (false)
- Complex sentences
  - Brackets
  - Connectives in order of precedence (high to low)
    - not (¬), and (∧), or (∨), implies (→), equivalent (↔)
  - If φ and ψ are sentences, then

$$(\varphi),\ \neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,\ \varphi \rightarrow \psi \text{ and } \varphi \leftrightarrow \psi$$

are also sentences

Semantics

- Meaning of a sentence is a truth value
  - {***T*, *F***}
- An interpretation is an assignment of truth values to the propositional variables
  - $\models_i \varphi$ Sentence φ is ***T*** in interpretation *i*
  - $\not\models_i \varphi$ Sentence φ is ***F*** in interpretation *i*

Semantic rules

- $\models_i$ ***T*** for all *i*
- $\not\models_i$ ***F*** for all *i*
- $\models_i \neg\varphi$ iff $\not\models_i \varphi$
- $\models_i \varphi \wedge \psi$ iff $\models_i \varphi$ and $\models_i \psi$ (conjunction)
- $\models_i \varphi \vee \psi$ iff $\models_i \varphi$ or $\models_i \psi$ (disjunction)
- $\models_i$ P iff *i*(P) = ***T***

Properties of sentences

- Equivalence $\varphi \equiv \psi$
  - φ and ψ are true for the same models
- Validity $\models \varphi$
  - A sentence is valid iff its truth value is ***T*** in all interpretations
  - Valid sentences are called tautologies
  - Examples: ***T***, $P \vee \neg P$, $A \rightarrow A$
- Satisfiability
  - A sentence is satisfiable iff it has at least one model

Entailment theorem

$$KB \models \varphi \quad \text{iff} \quad \models (KB \rightarrow \varphi)$$

- Enables proving **entailment** if we have means to prove the **validity** of a sentence
- This theorem is valid for all logics

Proving validity

- Truth table
- Equivalence rules
- Resolution

- $(X\to(Y\wedge Z))\leftrightarrow((X\to Y)\wedge(X\to Z))$

Proving by truth table

Proving by truth table

X	Y	Z	$Y \wedge Z$	$X \rightarrow Y$	$X \rightarrow Z$	$X \rightarrow (Y \wedge Z)$	$((X \rightarrow Y) \wedge (X \rightarrow Z))$	S

Proving by truth table

X	Y	Z	$Y \wedge Z$	$X \rightarrow Y$	$X \rightarrow Z$	$X \rightarrow (Y \wedge Z)$	$((X \rightarrow Y) \wedge (X \rightarrow Z))$	S
T	T	T	T	T	T	T	T	T
T	T	F	F	T	F	F	F	T
T	F	T	F	F	T	F	F	T
T	F	F	F	F	F	F	F	T
F	T	T	T	T	T	T	T	T
F	T	F	F	T	T	T	T	T
F	F	T	F	T	T	T	T	T
F	F	F	F	T	T	T	T	T

Equivalence (re-write) rules

- Logical equivalence
  - Different syntax
  - Same semantics
- Usage
  - Proving via showing equivalence
  - Modifying to a particular syntax to allow the use of other techniques (e.g. resolution)

Commutativity and associativity of connectives

- Commutativity:
  - $P \wedge Q$ can be replaced by $Q \wedge P$ (& vice-versa)
  - $P \vee Q$ can be replaced by $Q \vee P$ (& vice-versa)
  - $P \leftrightarrow Q$ can be replaced by $Q \leftrightarrow P$ (& vice-versa)

- Associativity
  - $((P \wedge Q) \wedge R)$ can be replaced by $(P \wedge (Q \wedge R))$ (& vice-versa)
  - $((P \vee Q) \vee R)$ can be replaced by $(P \vee (Q \vee R))$ (& vice-versa)

Distributivity of connectives

- And over or, or over and:
  - $(P\wedge(Q\vee R))$ can be replaced by $((P\wedge Q)\vee(P\wedge R))$
  - $(P\vee(Q\wedge R))$ can be replaced by $((P\vee Q)\wedge(P\vee R))$
- Over the implies sign
  - $(P\rightarrow(Q\vee R))$ can be replaced by $((P\rightarrow Q)\vee(P\rightarrow R))$
  - $(P\rightarrow(Q\wedge R))$ can be replaced by $((P\rightarrow Q)\wedge(P\rightarrow R))$

Double negation

- Double negations can be removed
  - $\neg\neg P$ is equivalent to P
- Caution when translating from natural language

de Morgan's laws and contraposition

- de Morgan's laws
  - $\neg(P \wedge Q)$ is equivalent to $(\neg P \vee \neg Q)$
  - $\neg(P \vee Q)$ is equivalent to $(\neg P \wedge \neg Q)$

- Contraposition
  - $(P \rightarrow Q)$ is equivalent to $(\neg Q \rightarrow \neg P)$

Other equivalences

- $(P \rightarrow Q)$ is equivalent to $(\neg P \vee Q)$
- $(P \leftrightarrow Q)$ is equivalent to $((P \rightarrow Q) \wedge (Q \rightarrow P))$
- $(P \leftrightarrow Q)$ is equivalent to $((P \wedge Q) \vee (\neg P \wedge \neg Q))$
- $(P \wedge \neg P)$ is equivalent to ***F***
- $(P \vee \neg P)$ is equivalent to ***T***

Propositional implication rules

- Re-write rules are good for bidirectional search
  - What if equivalence does not hold
- Modus Ponens

$$\frac{A \rightarrow B,\ A}{B}$$

  - Comma used for conjunction
  - Above the line: what we know
  - Below the line: what we can deduce

Proving Modus Ponens

A	B	$A \rightarrow B$	α: $A \rightarrow B$, A	β: B
True	True	True	True	True
True	False	False	False	False
False	True	True	False	True
False	False	True	False	True

Elimination and introduction of “and”

- “and” elimination

$$\frac{A_1, A_2, \ldots, A_n}{A_i}$$

[$1 \leq i \leq n$]

- “and” introduction

$$\frac{A_1, A_2, \ldots, A_n}{A_1 \wedge A_2 \wedge \ldots \wedge A_n}$$

Introduction of “or”; Unit resolution

- “or” introduction

$$\frac{A_i}{A_1 \vee A_2 \vee \ldots \vee A_n}$$

[$1 \leq i \leq n$]

- Unit resolution
  - Basis for theorem proving

$$\frac{(A \vee B) \wedge \neg B}{A}$$

Problems

- Too many predicates
  - Sample r.: “If you see a stop sign, then stop!”
  - A new predicate for every stop sign
- Slow inference
- No variables (many constants needed)
  - Even more predicates

First order predicate logic (FOPL)

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- Agent model
- Problem solving by search
- Propositional logic

Outline

- Semantics
- New elements
  - Predicates
  - Functions
  - Variables
  - Quantifiers
- Parts of logic formulas
- Instantiation and substitution

First order predicate logic

- More powerful than propositional logic
- Models contain objects
  - Domain of a model is its set of objects (domain elements)
- Domain elements are related in various ways; formally as a set of ordered tuples
  - capital = {<Hungary, Budapest>, <Italy, Rome>, … }

Semantics of FOPL

- An **interpretation** maps domain objects, relations, and functions to symbols
- Domains may have infinitely many objects (e.g., all integers)
  - Number of models and interpretations is unbounded
  - Model checking not applicable to check entailment

Predicates

- Predicates express relations between certain things
  - Predicate **name** identifies the relationship
  - **Arguments** are the things being related (constants, functions and variables)
  - **Arity** is the number of arguments
- Examples
  - Binary (arity=2): capital = {<Hungary, Budapest>, <Italy, Rome>, … }
  - Unary (arity=1) city = {Budapest, Rome, … }

Functions

- Special predicates
- In a function of arity *n*
  - The first *n* - 1 arguments are inputs
  - The last argument is the output (single-valued)
- Example
  - Predicate form: sum_of(2,3,5)
  - Functional form: sum_of(2,3) → 5
  - inputs: 2,3; output: 5

Variables

- How to express a sentence like
  - "There's a drink in Starbucks the price of which is $2."
- *price_of(drink, starbucks) = $2*
  - Problem: drink is a constant
- *price_of(X, starbucks) = $2*
  - *X* is a variable referring to some drink
  - Problem: which drink?

Quantifiers

- Symbol for "there exists": ∃ ("existential quantifier")
  - *∃X (price_of(X, starbucks) = $2)*
- Symbol "for all": ∀ ("universal quantifier")
  - "All cats like milk." : *∀X (cat(X) → likes(X,milk))*
  - "All drinks cost $2." expresses
    - "Beer costs $2."
    - "Wine costs $2."
    - etc.
- Variables can be **instantiated**

Terms

- A **term** is a logical expression that refers to an object in the domain
  - Constant symbols (e.g. Hungary)
  - Variables
  - Function values (e.g. capital(Hungary))

Logic formulas

- An **atomic formula** is statement that combines
  - Terms (referring to objects), and
  - Predicate symbols (referring to relations)
- Example: capital(Hungary, Budapest)
- A **literal** is an atomic formula or its negation
- A **compound formula** is formed from literals using logical connectives
- A **sentence** is a logic formula in which all variables are bound

Instantiation and substitution

- FOPL sentences have quantified variables
  - Instantiation ("**grounding**")
  - Ground terms (constants, functions of ground terms)
- Substitution of a variable
  - Grounding: replacing the variable by a ground term
  - Replacing the variable by another variable
- Example
  - $\forall$ X,Y friend(X, Y)
  - Subst({X/Sue, Y/Mary})
  - friend(X, Y) = friend(Sue, Mary)

Summary

- Semantics
- New elements
  - Predicates
  - Functions
  - Variables
  - Quantifiers
- Parts of logic formulas
- Instantiation and substitution

Inference in FOPL

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Logical inference schemes

- *Deduction*: formal logical reasoning
  - Premises: 1. All men are mortal. 2. Aristotle is a man.
  - Conclusion: Aristotle is mortal.
- *Induction*: generalization
  - Premise: The sun has risen in the east every morning up until now.
  - Conclusion: The sun will also rise in the east tomorrow.
- *Abduction*: choosing an explanation
  - Premise: 1. Flu causes fever. 2. Peter has fever.
  - Conclusion: Peter has flu.

The case of the silk gloves

"It was elementary my dear Watson. The killer always left a silk glove at the scene of the murder. That was his calling card. Our investigations showed that only three people have purchased such gloves in the past year. Of these, Professor Doolally and Reverend Fisheye have iron-clad alibis, so the murderer must have been Sergeant Heavyset. When he tried to murder us with that umbrella, we knew we had our man."

Not so elementary…

"The killer always left a silk glove at the scene of the murder." (induction)

"That was his calling card." (abduction)

"…only three people have purchased such gloves in the past year." (model generation)

"Professor Doolally and Reverend Fisheye have iron-clad alibis." (constraint based reasoning)

"…so the murderer must have been Sergeant Heavyset." (deduction)

First Order Predicate Logic (FOPL)

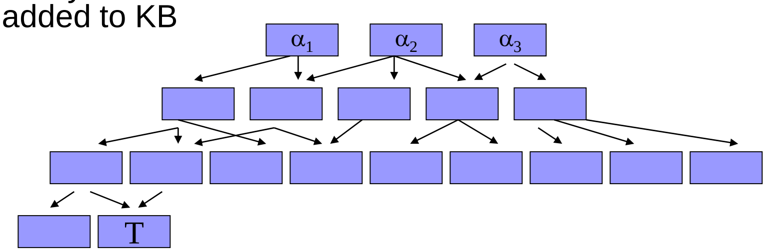
- Most used and analyzed logic
- Completeness: Gödel, Herbrand, 1930
  - If a FOPL statement is valid then it is provable
  - If KB $\models$ a then KB $\vdash$ a
- Validity is semi-decidable
- Resolution: Robinson, 1963

Chains of inference

- Remember the problem we are trying to solve
  - Search for a path from axioms α_i, to theorem T
- Three approaches
  - Forward chaining
  - Backward chaining
  - Proof by contradiction
- Specification of a search problem:
  - Representation of states (first order predicate logic sentences)
  - Initial state (changes with the approach)
  - Operators (rules of inference, usually implication rules)
  - Goal state (changes with the approach)

Forward Chaining

- Start with initial axioms (atomic sentences) and deduce new facts by applying modus ponens
- Repeat until possible or query is answered
- Problems
  - Generates many irrelevant facts
  - Every rule has to be rechecked whenever a new fact is added to KB



Forward Chaining

- A first-order definite clause is a disjunction of literals of which exactly one is positive
  - Example
    - $white(X) \wedge potable(X) \rightarrow milk(X)$ is logically equivalent to $\neg white(X) \vee \neg\, potable(X) \vee milk(X)$
  - Modus ponens can be easily applied to first-order definite clauses
  - All variables are implicitly universally quantified
- Sound, complete

Backward Chaining

- Work backwards from the goal, chaining rules to find facts that support the conclusion
- For each node the inference rule has to be inverted
  - Which operator could have been applied to which state to produce this state (sentence)
- No problem when using equivalences
  - Can also use a bidirectional search (from both ends)
- Difficult when using implications
  - Many possible ways to invert operators

Proof By Contradiction

- "Reductio ad absurdum"
- Most often used method
- Idea: by showing that the assumption contradicts a set of axioms we can prove that the assumption is false

- KB' = Set of axioms (KB) + negated theorem (¬Th)
- If the ***F*** statement can be deduced from KB' then ¬Th is false, and thus Th must be true
- Advantage: heuristic function can be defined based on the distance from the 'False' statement

First order implication rules

- Propositional implications and equivalences
- First order implication rules
  - Quantifiers
  - Variables
  - Substitution

Universal elimination

- In a sentence φ any universally quantified variable v can be replaced by any ground term g

$$\frac{\forall v \;\; \varphi}{\text{subst}(\{v/\text{g}\}, \varphi)}$$

- Note: the variable has to be removed from quantification
- Example
  - $\forall x$ friend(Sue, x) becomes friend(Sue, Ann)

Existential introduction

- In a sentence φ any ground term g can be substituted by a variable *v* if it does not appear in φ

$$\frac{\varphi}{\exists v\ \mathrm{subst}(\{g/v\}, \varphi)}$$

- Example
  - friend(Sue, Ann) becomes ∃*x* friend(Sue, *x*)
- Exercise
  - Find a sentence where *v* is in φ such that this implication rule is not sound

Universal introduction

- In a sentence φ any constant k can be substituted by a variable *v* if k is not mentioned in any of the premises or undischarged assumptions and *v* does not appear in φ

$$\frac{\varphi}{\forall v\ \mathrm{subst}(\{k/v\}, \varphi)}$$

- Example
  - friend(Sue, Doe) becomes ∀*x* friend(Sue, *x*)

Existential elimination

- In a sentence φ any existentially quantified variable *v* can be replaced by any constant k, if k appears neither in φ nor anywhere else in the derivation

$$\frac{\exists v \;\; \varphi}{\mathrm{subst}(\{v/k\}, \varphi)}$$

- k is called a Skolem constant
- Existential elimination is a special case of skolemization (see later)

Propositionalization

- Universal and existential elimination allow for inferring non-quantified sentences from quantified ones
- Reduces first-order inference to propositional inference
- Problem
  - Function symbols allow infinitely many ground terms: father(father (father (. . .)))
  - Can be overcome by Herbrand's theorem (R-N pp. 274–275)
- **Entailment** in FOPL is **semi-decidable** (Church)
  - Any entailed sentence can be proven
  - Not all false sentences can be disproven (Halting problem)

Inference with variables

- Premise
  - $\forall x$ (knows(Bob, x) $\rightarrow$ loves(Bob, x))
- From "knows(Bob, Alice)"
  - using modus ponens gives: "loves(Bob, Alice)"
- From "knows(Bryan, Alice)"
  - modus ponens cannot be used
- How to check applicability when variables are present?

Unifying predicates

- Expressions x_1 and x_2 are unifiable iff there exists a substitution Θ such that

  $$\text{subst}(\Theta, x_1) = \text{subst}(\Theta, x_2),$$

  where subst(Θ, x) applies Θ to x
- Unification by substitution ({X/Alice})
  - knows(Bob, X) and knows(Bob, Alice)
- Possibilities
  - variable-variable
  - variable-constant
  - variable-function

The unification algorithm

- A recursive algorithm
  - Passes around a set of substitutions, called *mu*
  - Makes sure that new substitutions are consistent with old ones
- `unify(x,y) = unify_internal(x,y,{})`
  - *x* and *y* can be variables, constants, lists, or compounds
- `unify_internal(x,y,mu)`
  - *x* and *y* are sentences, *mu* is a set of substitutions
  - finds substitutions making *x* look exactly like *y*
- `unify_variable(var,x,mu)`
  - *var* is a variable
  - finds a single substitution (which may be in *mu* already)

unify_internal

`unify_internal(x,y,mu)`

```
1.if (mu==failure) then return failure
2.if (x==y) then return mu
3.if (isa_variable(x)) then return
    unify_variable(x,y,mu)
4.if (isa_variable(y)) then return
    unify_variable(y,x,mu)
5.if (isa_compound(x) & isa_compound(y)) then return
    unify_internal(args(x),args(y),
               unify_internal(op(x),op(y),mu))
6.if (isa_list(x) & isa_list(y)) then return
    unify_internal(tail(x),tail(y),
               unify_internal(head(x),head(y),mu))
7.return failure
```

unify_variable

unify_variable(var,x,mu)

```
1. if (a substitution var/val is in mu) then
    return unify_internal(val,x,mu)
2. if (a substitution x/val is in mu) then
    return unify_internal(var,val,mu)
3. if (var occurs anywhere in x) return
    failure
4. add var/x to mu and return
```

Notes on the unification algorithm

- `unify_internal` will not match a constant to a constant, unless they are equal (case 2)
- Case 5 in `unify_internal` checks that two compound operators are the same (e.g., same predicate name)
- Case 6 in `unify_internal` causes the algorithm to recurse covering the whole list
- Cases 1 and 2 in `unify_variable` check that neither inputs have already been substituted

The occurs check

- When substituting variable x with an expression $f(x,y)$
  - x will be replaced by $f(x,y)$
  - But $f(x,y)$ still contains an instance of x, which has to be replaced again
  - We get $f(f(x,y),y)$, and then $f(f(f(x,y),y),y)$, etc.
  - Infinite recursion → the algorithm will not stop (halt)
- Case 3 in `unify_variable` checks this to avoid this situation
- Problem: Occurs check slows down the algorithm
  - Its complexity is $O(n^2)$, where n is the size of expressions being unified

Unification exercises

- nice(Alice) – nice(Mary)
- sees(x,Alice) – sees(y,Alice)
- sees(x,Alice) – sees(Mary,y)
- x – child(Alice,x)
- friends(x,y,Alice) ∧ father(sonof(Bob),Bob) – father(z,Bob) ∧ friends(Mary,z,u)
- R(F(y),x) – R(x,F(A))
- R(F(y),y,x) – R(x,F(A),F(v))
- F(G(w),H(w,J(x,u))) – F(G(v),H(u,v))
- F(x,F(u,x)) – F(F(y,A),F(z,F(B,z)))

Full resolution rule

- Resolution rules remove predicates in predicate logic
  - This is known as **resolving** the two sentences
- Unit resolution rule

$$\frac{(A \vee B),\ \neg B}{A}$$

- Full resolution rule (using CNF)

$$\frac{(A \vee B),\ (\neg B \vee C)}{A \vee C}$$

- With implication

$$\frac{(\neg A \rightarrow B),\ (B \rightarrow C)}{\neg A \rightarrow C}$$

Generalized resolution rule

- Given two CNF sentences

$$p_1 \vee p_2 \vee \ldots \vee p_m \quad \text{and} \quad q_1 \vee q_2 \vee \ldots \vee q_n$$

- If p_j and $\neg q_k$ can be unified, i.e. unify(p_j, $\neg q_k$) = Θ, then

$$\frac{p_1 \vee \ldots \vee p_j \vee \ldots p_m, \qquad q_1 \vee \ldots \vee q_k \vee \ldots \vee q_n}{\text{subst}(\Theta\ ,\ (p_1 \vee \ldots p_{j\text{-}1} \vee p_{j+1} \vee \ldots \vee p_m \vee q_1 \vee \ldots \vee q_{k\text{-}1} \vee q_{k+1} \vee \ldots q_n))}$$

Resolution with variables

- $P(x) \vee Q(x, y)$
- $\neg P(A) \vee R(B, z)$

- $\text{subst}(\{x/A\}, Q(x, y) \vee R(B, z))$
- $Q(A, y) \vee R(B, z)$

Local variable scope

- $P(x) \vee Q(x, y)$
- $\neg P(A) \vee R(B, x)$

Local variable scope

- $P(x_1) \vee Q(x_1, y)$
- $\neg P(A) \vee R(B, x_2)$

- $subst(\{x_1/A\}, Q(x_1, y) \vee R(B, x_2))$
- $Q(A, y) \vee R(B, x_2)$

CNF in FOPL

- Sentences need to be in conjunctive normal form (CNF)
  - Literals can contain variables, assumed to be universally quantified
- Example
  - $white(X) \wedge potable(X) \rightarrow milk(X)$ becomes $\neg white(X) \vee \neg\ potable(X) \vee milk(X)$

Conversion to clausal form

- 1. Eliminate $\rightarrow$ and $\leftrightarrow$
- 2. Drive in $\neg$ to atomic level
- 3. Rename variables apart
- 4. Skolemize
- 5. Drop universal quantifiers
- 6. Convert to CNF
- 7. Rename variables in each clause

Skolemization

- Substitute a new constant for each existentially quantified variable
  - $$\frac{\exists x\, P(x)}{P(C_S)}$$
- Substitute a new function of all universally quantified variables in enclosing scopes for each existentially quantified variable
  - $$\frac{\forall x\, \exists y\, P(x, y)}{\forall x\, P(x, f_S(x))}$$

"A cat called Tuna" (from textbook)

- Jack owns a dog
- Every dog owner is an animal lover
- No animal lover kills an animal.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- Did Curiosity kill the cat?

A. $\exists x$ (Dog(x) $\wedge$ Owns(Jack,x))
B. $\forall x$ ((($\exists y$) Dog(y) $\wedge$ Owns(x, y)) $\rightarrow$ AnimalLover(x))
C. $\forall x$ (AnimalLover(x) $\rightarrow$ (($\forall y$) Animal(y) $\rightarrow$ $\neg$Kills(x,$\underline{y}$)))
D. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
E. Cat(Tuna)
F. $\forall x$ (Cat(x) $\rightarrow$ Animal(x))
G. Kills(Curiosity, Tuna)

Conversion to clausal form

- 1. Eliminate $\rightarrow$ and $\leftrightarrow$
- 2. Drive in $\neg$ to atomic level
- 3. Rename variables apart
- 4. Skolemize
- 5. Drop universal quantifiers
- 6. Convert to CNF
- 7. Rename variables in each clause

Sentence A & B

- **(A) $\exists x.\ \mathrm{Dog}(x) \wedge \mathrm{Owns}(\mathrm{Jack},x)$**
- $\mathrm{Dog}(D) \wedge \mathrm{Owns}(\mathrm{Jack},D)$

- **(B) $\forall x.\ (\exists y.\ \mathrm{Dog}(y) \wedge \mathrm{Owns}(x,y)) \rightarrow \mathrm{AnimalLover}(x)$**
- $\forall x.\ (\neg\exists y.\ \mathrm{Dog}(y) \wedge \mathrm{Owns}(x,y)) \vee \mathrm{AnimalLover}(x)$
- $\forall x.\ \forall y.\ \neg(\mathrm{Dog}(y) \wedge \mathrm{Owns}(x,y)) \vee \mathrm{AnimalLover}(x)$
- $\forall x.\ \forall y.\ \neg\mathrm{Dog}(y) \vee \neg\mathrm{Owns}(x,y) \vee \mathrm{AnimalLover}(x)$
- $\neg\mathrm{Dog}(y) \vee \neg\mathrm{Owns}(x,y)) \vee \mathrm{AnimalLover}(x)$

Sentence C & D

- **(C) $\forall x.\ \mathrm{AnimalLover}(x) \rightarrow (\forall y.\ \mathrm{Animal}(y) \rightarrow \neg\mathrm{Kills}(x,y))$**
- $\forall x.\ \neg\mathrm{AnimalLover}(x) \vee (\forall y.\ \mathrm{Animal}(y) \rightarrow \neg\mathrm{Kills}(x,y))$
- $\forall x.\ \neg\mathrm{AnimalLover}(x) \vee (\forall y.\ \neg\mathrm{Animal}(y) \vee \neg\mathrm{Kills}(x,y))$
- $\neg\mathrm{AnimalLover}(x) \vee \neg\mathrm{Animal}(y) \vee \neg\mathrm{Kills}(x,y)$

- **(D) $\mathrm{Kills}(\mathrm{Jack},\mathrm{Tuna}) \vee \mathrm{Kills}(\mathrm{Curiosity},\mathrm{Tuna})$**

Sentence E, F and neg. Th.

- **(E) Cat(Tuna)**
- **(F) $\forall x$. Cat(x) $\rightarrow$ Animal(x)**
- $\neg$Cat(x) $\vee$ Animal(x)
- **(Th) $\neg$Kills(Curiosity,Tuna)**

Solution

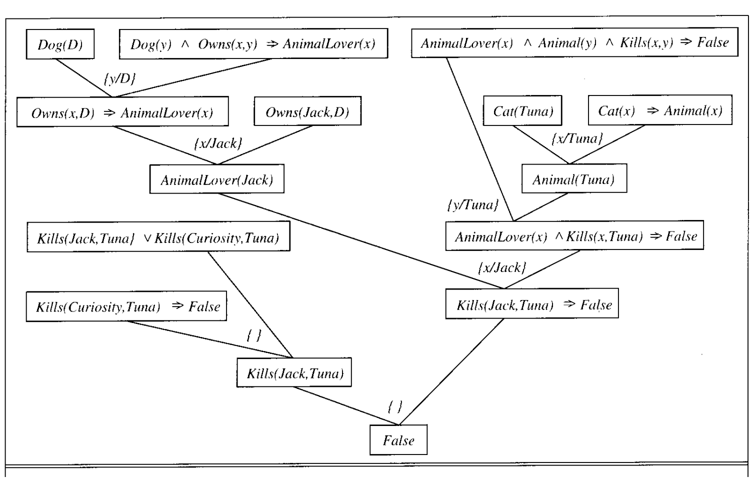
- (D), (Th) Kills(Jack,Tuna) (G)
- (E), (F), {x/T} Animal(Tuna) (H)
- (C), (G), {x/J, y/T}
  $\neg$AnimalLover(Jack) $\vee$ $\neg$Animal(Tuna) (I)
- (H) , (I) $\neg$AnimalLover(Jack) (J)
- (B), (J), {x/J} $\neg$Dog(y) $\vee$ $\neg$Owns(Jack,y) (K)
- (A2) $\neg$Dog(D) (L)
- (A1), (L) False

CNF (Implicative form)

- Jack owns a dog
- Every dog owner is an animal lover
- No animal lover kills an animal.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- Did Curiosity kill the cat?

A1. Dog(D)
A2. Owns(Jack,D)
B. Dog(y) $\wedge$ Owns(x,y) $\rightarrow$ AnimalLover(x)
C. AnimalLover(x) $\wedge$ Animal(y) $\rightarrow$ $\neg$Kills(x,y)
D. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
E. Cat(Tuna)
F. $\neg$Cat(x) $\vee$ Animal(x)

Graph of proof



Equality

- Unification of different constants
  - Today(Thu), Today(Thursday)
- Expanding the KB is not sufficient
  - Thu = Thursday
- Extra axioms are needed
  - Equality is symmetric, reflexive and transitive
- Equality statements for each predicate:
  - $\forall x,y\ x = y \rightarrow (P(x) \leftrightarrow P(y))$ etc.

Demodulation rule

- Takes two input sentences, one expressing an equality ($\alpha = \beta$)
- Finds a unification for α with a term in another clause ($\Theta = \text{unify}(\alpha, \gamma)$)
- Applies Θ to β (not α)
- Replaces occurrence of γ with Subst(Θ, β)

$$\frac{\alpha = \beta,\ (\ldots, \gamma, \ldots)}{(\ldots, \text{Subst}(\Theta, \beta), \ldots)}$$

Demodulation drawbacks

- Cannot bind variables in expression
  - father(Adam) = Bob
    - α: father(Adam) , β: Bob
  - older(father(x), x)
    - γ: father(x), Θ = {x/Adam}, Subst(Θ, β) = Bob
  - older(Bob, Adam) : not derived, only older(Bob, x)

$$\frac{\alpha = \beta, \quad (\ldots, \gamma, \ldots) \quad \Theta = \mathrm{unify}(\alpha, \gamma)}{(\ldots, \mathrm{Subst}(\Theta, \beta), \ldots)}$$

- Equation must be a unit clause
  - (x = Adam $\wedge$ y = Bob) $\rightarrow$ father(x) = y
    - α cannot be father(x), since the equation is inside an implication
  - older(father(x), x)
  - (x = Adam $\wedge$ y = Bob) $\rightarrow$ older(Bob, Adam)

Paramodulation

- F(x) = B
- Q(y) $\vee$ W(y,F(y))
- Q(y) $\vee$ W(y,B)

- $\alpha \vee (s = t)$
- $\beta \vee \gamma\,[r] \quad \Theta = \mathrm{unify}(s,r)$
- $\mathrm{Subst}(\Theta,(\alpha \vee \beta \vee \gamma\,[r]))$

- G(x) $\vee$ F(x) = B
- Q(y) $\vee$ W(y,F(y))
- G(y) $\vee$ Q(y) $\vee$ W(y,B)

- $s = F(x)$; $\quad t = B$
- $\gamma[\cdot] = W(y,\cdot)$; $\quad r = F(y)$
- $\Theta = \{x/y\}$

Horn clauses

- Have the form: $P1 \wedge P2 \wedge \ldots \wedge Pn \rightarrow Q$
- Special cases
  - $P1 \wedge P2 \wedge \ldots \wedge P_n \rightarrow$ False
  - True $\rightarrow Q$
- Enables polynomial time inference
- Prolog (SLD resolution)
  - S: Selection function
  - L: Linear sequence of clauses
  - D: Definite clauses
  - Ordered resolution

Sample Prolog program

```
fun(X) :-
   red(X),
   car(X).

fun(X) :-
   blue(X),
   bike(X).
```

```
car(vw_beatle).
car(ford_escort).
bike(harley_davidson).
red(vw_beatle).
red(ford_escort).
blue(harley_davidson).

?-
  fun(harley_davidson).
yes
```

Resolution proving as search

- Search space: Sentences in FOPL
- Initial state: {KB, ¬Th.}
- Operator: generalized resolution inference rule
- Goal Check: Empty clause found
- Solution: two possibilities
  - Path from axioms to false clause (if we want proof)
  - Just the fact that we have reached the false clause (no proof required)

Elimination strategies

- Identical clause elimination
  - a resolution refutation without a clause occurring twice
- Pure literal elimination
  - A literal with no negated occurrence makes its clause superfluous
- Tautology elimination
  - No effect on satisfiability
- Subsumption elimination
  - Remove clauses that are more specific than others in the KB

Restriction strategies

- Unit resolution
  - One resolvent is always a unit clause (single literal)
- Input resolution
  - One resolved clause is always taken from initial KB
  - Complete, if the KB contains Horn clauses
- Linear resolution
  - One resolved clause is always taken from either the initial KB or from the ancestor of the other resolvent; Complete
- Set of Support
  - One resolvent is always taken from a subset of initial KB or from its descendant
  - Complete, if the clauses outside the SoS are satisfiable
- Ordered resolution
  - Clauses are treated as ordered sets, resolution is allowed only on the first literal

Applications of resolution

- Automated Theorem Proving (ATP)
- Proof verification
- Proof compression
- Automated Conjecture Making
- Interactive proving
- Proof planning

A famous example for ATP

- Axiomatization of Boolean algebra
- Standard axioms
  - $\forall a, b \in B \; a+b = b+a$
  - $\forall a, b, c \in B \; (a+b)+c = a+(b+c)$
  - $\exists 0 \in B$ (unit element for +) $0 + a = a$
  - $\forall a \in B \; \neg\neg a = a$
  - $\forall a \in B \; \neg(a + \neg a) = 0$
  - $\forall a, b, c \in B$
    $a + \neg(\neg b + \neg c) = \neg(\neg(a+b) + \neg(a+c))$

Robbins Problem

- Huntigton's proposal to axiomatize Boolean algebras (1933)
  - Commutativity + associativity
  - $\forall a, b \in B.\; a = \neg(\neg a + b) + \neg(\neg a + \neg b)$
- Herbert Robbins
  - Commutativity + associativity
  - $\forall a, b \in B.\; a = \neg(\neg(a + b) + \neg(a + \neg b))$
  - Got coined "Robbins algebra"

Solving the Robbins Problem

- William McCune and Larry Wos
  - Argonne National Laboratories
  - EQP & Otter (first order provers)
  - EQP solved this in 8 days, completed on Oct. 10,1996
  - One step from the proof:
    $\neg(\neg(\neg(\neg(\neg(x) + x) + \neg(\neg(x) + x) + x + x + x + x) + \neg(\neg(\neg(x) + x) + x + x + x) + x) + x) = \neg(\neg(\neg(x) + x) + \neg(\neg(x) + x) + x + x + x + x)$
  - Otter proved that the proof is OK (its successor is called Prover9)

```
----- EQP 0.9, June 1996 -----
The job began on eyas09.mcs.anl.gov, Wed Oct  2 12:25:37 1996
UNIT CONFLICT from 17666 and 2 at 678232.20 seconds.
---------------- PROOF ----------------
2 (wt=7) [] -(n(x + y) = n(x)).
3 (wt=13) [] n(n(n(x) + y) + n(x + y)) = y.
5 (wt=18) [para(3,3)] n(n(n(x + y) + n(x) + y) + y) = n(x + y).
6 (wt=19) [para(3,3)] n(n(n(n(x) + y) + x + y) + y) = n(n(x) + y).
24 (wt=21) [para(6,3)] n(n(n(n(x) + y) + x + y + y) + n(n(x) + y)) = y.
47 (wt=29) [para(24,3)] n(n(n(n(n(x) + y) + x + y + y) + n(n(x) + y) + z) + n(y + z)) = z.
48 (wt=27) [para(24,3)] n(n(n(n(x) + y) + n(n(x) + y) + x + y + y) + y) = n(n(x) + y).
146 (wt=29) [para(48,3)] n(n(n(n(x) + y) + n(n(x) + y) + x + y + y + y) + n(n(x) + y)) = y.
250 (wt=34) [para(47,3)] n(n(n(n(n(x) + y) + x + y + y) + n(n(x) + y) + n(y + z) + z) + z) = n(y + z).
996 (wt=42) [para(250,3)] n(n(n(n(n(n(x) + y) + x + y + y) + n(n(x) + y) + n(y + z) + z) + z + u) + n(n(y + z) +
      u)) = u.
16379 (wt=21) [para(5,996),demod([3])] n(n(n(n(x) + x) + x + x + x) + x) = n(n(x) + x).
16387 (wt=29) [para(16379,3)] n(n(n(n(n(x) + x) + x + x + x) + x + y) + n(n(n(x) + x) + y)) = y.
16388 (wt=23) [para(16379,3)] n(n(n(n(x) + x) + x + x + x + x) + n(n(x) + x)) = x.
16393 (wt=29) [para(16388,3)] n(n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) + x) = n(n(x) + x).
16426 (wt=37) [para(16393,3)] n(n(n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) + x + y) + n(n(n(x) + x) + y)) =
      y.
17547 (wt=60) [para(146,16387)] n(n(n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) + n(n(n(x) + x) + x + x + x)
      + x) + x) = n(n(n(x) + x) + n(n(x) + x) + x + x + x + x).
17666 (wt=33) [para(24,16426),demod([17547])] n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) = n(n(n(x) + x)
      + x + x + x).
------------ end of proof -------------
```

A problem by Lewis Carol

- The only animals in this house are cats
- Every animal that loves to gaze at the moon is suitable for a pet
- When I detest an animal, I avoid it
- No animals are carnivorous unless they prowl at night
- No cat fails to kill a mice
- No animals ever like me, except those that are in this house
- Kangaroos are not suitable for pets
- None but carnivorous animals kill mice
- I detest animals that do not like me
- Animals that prowl at night always love to gaze at the moon
- Therefore, I always avoid a kangaroo

Summary

- FOPL semantics
- Chains of inference
- Propositionalization
- Resolution
  - Unification algorithm
  - Generalized resolution
  - Equality
  - Resolution strategies
- Automatic theorem proving

Planning

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- Agent model
- Problem solving by search
  - Non-informed search strategies
  - Informed search strategies
- Logic
  - Propositional logic
  - Predicate logic

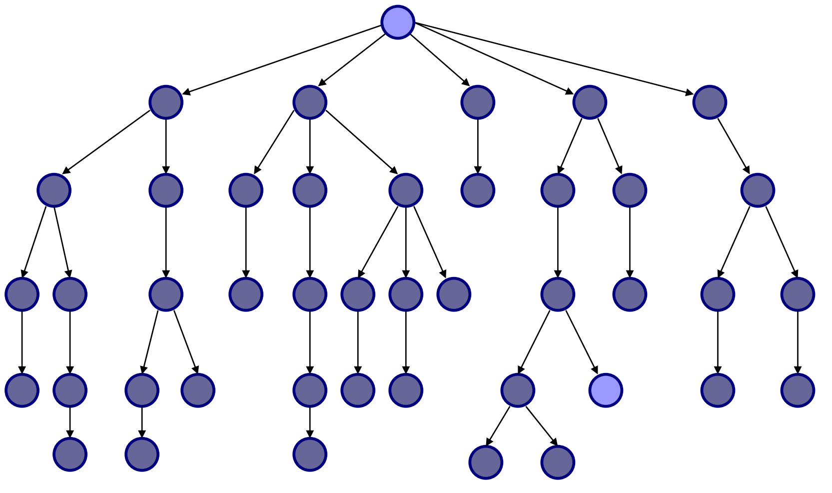
Outline

- Planning and search
- Situation calculus
- Partial order planning
- Graphplan

Planning

- Planning
  - Initial state
  - Goal state
  - Set of actions
- Can be described as a search problem

Planning vs. search

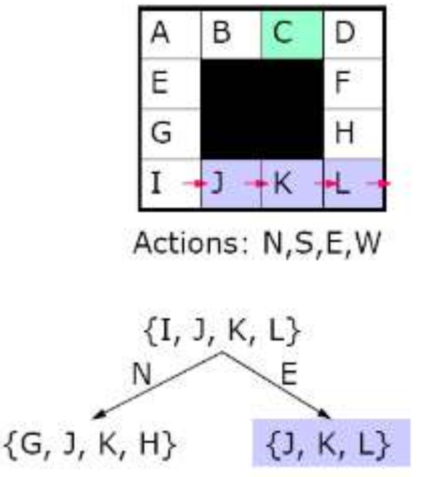


Problems of using search for planning

- Description of actions
  - By defining follower states
- Description of states
  - Every state has to be exactly given
- Description of goals
  - Only by defining goal states (and the heuristic)
- Description of plan
  - Fixed order of actions, can only be started from the start or the goal state

Undefined starting state

- What if initial state is not known exactly?
  - E.g. "Start in bottom row, with goal being C"
- Search over "sets" of underlying (atomic) states
- Inefficient approach
  - Exponential blowup in the number of sets of atomic states



Planning as logic search

- A classic approach to planning: **situation calculus**
- It uses
  - FOPL descriptions of the relevant sets of states and actions
  - ATP to find a plan

Situation Calculus

- Reification – treat situations as objects and use them as predicate arguments
  - At(Agent, Room 13, s_8) where s_8 refers to a particular situation
- Result function – gives the new situation resulting from taking an action in another situation
  - Result(StandUp, s_1) = s_3
- Effect Axioms – what is the effect of taking an action in the world
  - $\forall$ x.s. Present(x,s) $\wedge$ Portable(x) $\rightarrow$ Holding(x, Result(Grab, s))
  - $\forall$ x.s. $\neg$ Holding(x, Result(Drop, s))
- Frame Axioms - what doesn't change
  - $\forall$ x.s. color(x,s) = color(x, Result(Grab, s))
  - Can be included among effect axioms

Planning in situation calculus

- Use theorem proving to find a plan
- Goal state: $\exists$s. At(Home, s) $\wedge$ Holding(Gold, s)
- Initial state: At(Home, s_0) $\wedge$ $\neg$ Holding(Gold, s_0) $\wedge$ Holding(Rope, s_0) …
- Plan: Result(North, Result(Grab, Result(South, s_0)))
  - A situation that satisfies the requirements
  - Course of actions can be read out
  - First, move South, then Grab and then move North

Problems of using situation calculus for planning

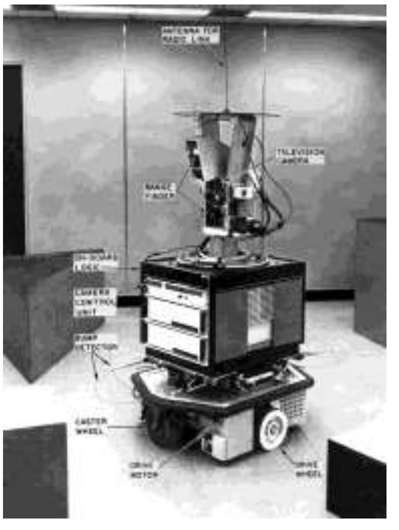
- Reducing specific planning problem to general problem of theorem proving is not efficient
  - Exponential complexity
  - Optimality of plan is difficult to assess
- A more specialized approach can exploit special properties of planning problems

Special properties of planning

- Connect action descriptions and state descriptions (focus searching)
  - If goal contains Holding(Gold) and Grab(Gold) causes Holding(Gold) to be true, then plan should include Grab(Gold)
- Add actions to a plan in any order
- Sub-problem independence
- Restrict language for describing goals, states and actions

STRIPS: Stanford Research Institute Problem Solver

- ~1971: The first real planning system
- Pushing boxes between rooms



STRIPS representation

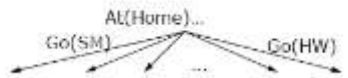
- States: conjunctions of ground literals
  - In(robot, r_3) $\wedge$ Closed($door_6$) $\wedge$ …
- Goals: conjunctions of literals
  - (implicit $\exists$ r) In(Robot, r) $\wedge$ In(Charger, r)
- Actions (operators)
  - Name (implicit $\forall$): Go(r_1, r_2)
  - Preconditions: conjunction of literals
    - At(r_1) $\wedge$ Path(r_1, r_2)
  - Effects: conjunctions of literals (aka add-list & delete-list)
    - At(r_2) $\wedge$ $\neg$ At(r_1)
  - Assumes no inference in relating predicates (only equality)

STRIPS example

- Action
  - Buy(x, store)
    - Pre: At(store), Sells(store, x)
    - Eff: Have(x)
  - Go(x, y)
    - Pre: At(x)
    - Eff: At(y), ¬At(x)
- Goal
  - Have(Milk) ∧ Have(Banana) ∧ Have(Drill)
- Start
  - At(Home) ∧ Sells(SM, Milk) ∧ Sells(SM, Banana) ∧ Sells(HW, Drill)

Planning algorithms

- Progression planners: consider the effect of all possible actions in a given state



- Regression planners: to achieve a goal, what must have been true in previous state
  - Have(M) ∧ Have(B) ∧ Have(D)
  - Buy(M,store)
    At(store) ∧ Sells(store,M) ∧ Have(B) ∧ Have(D)
- Both have the problem of lack of direction – what action or goal to pursue next

Search in plan space

- Situation space – both progressive and regressive planners plan in space of situations
- Plan space – start with null plan and add steps to plan until it achieves the goal
  - Much smaller complexity
  - Planning order independent from execution order
  - Least-commitment
    - “what actions” before “what order”
  - Means-ends analysis – Try to match the available means to the current ends

Partially ordered plan

- Set of steps (instance of an operator)
- Set of ordering constraints $S_i < S_j$
- Set of variable binding constraints $v = x$
  - v is a variable in a step; x is a constant or another variable
- Set of causal links $S_i \rightarrow_c S_j$
  - Step i achieves precondition c for step j

Initial plan

- Steps: {start, finish}
- Ordering: {start < finish}
- start
  - Pre: none
  - Eff: start conditions
- finish
  - Pre: goal conditions
  - Eff: none

Completeness and consistency

- A plan is **complete** iff every precondition of every step is achieved by some other step
- $S_i \rightarrow_c S_j$ ("step *i* achieves c for step *j*") iff
  - $S_i < S_j$
  - $c \in$ effects(S_i)
  - $\neg\exists\, S_k.\ \neg c \in$ effects(S_k) and $S_i < S_k < S_j$ is consistent with the ordering constraints
- A plan is **consistent** iff
  - the ordering constraints are consistent and
  - the variable binding constraints are consistent

Partially Ordered Plan (POP)

- Plan
  - Steps
  - Ordering constraints
  - Variable binding constraints
  - Causal links
- POP Algorithm
  - Make initial plan
  - Loop until plan is a complete
    - Select a subgoal
    - Choose an operator
    - Resolve threats

Choosing an operator

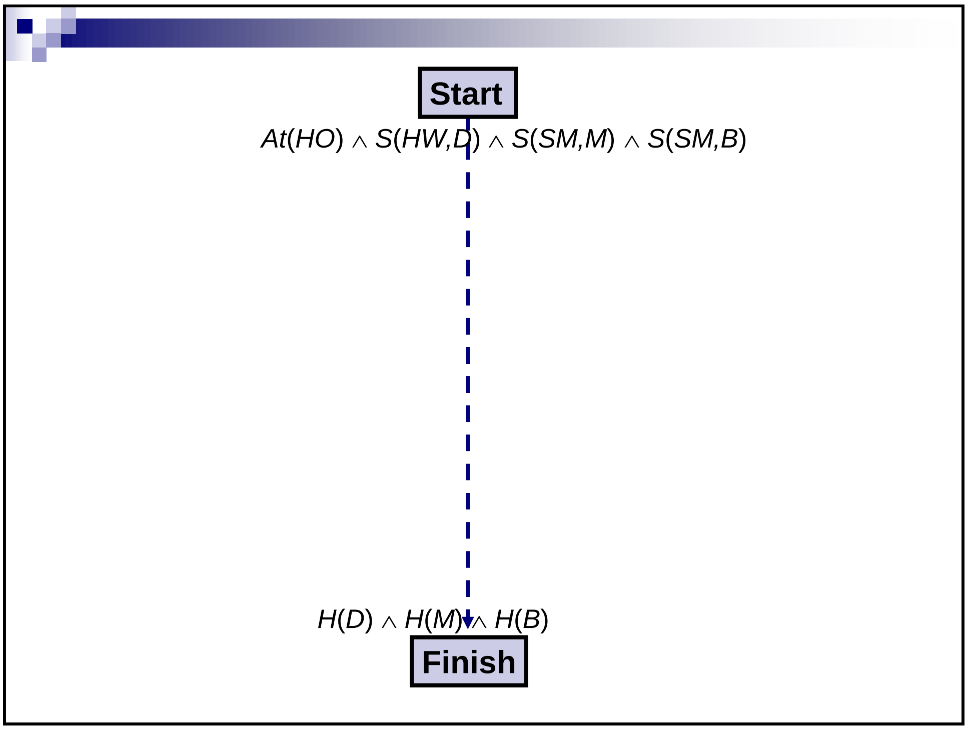
- Choose operator(c, S_{needs})
  - Choose a step S from the plan or a new step S by instantiating an operator that has c as an effect
  - If there's no such step, then fail (backtrack)
  - Add causal link $S \rightarrow_c S_{needs}$
  - Add ordering constraint $S < S_{needs}$
  - Add variable binding constraints if necessary
  - Add S to steps if necessary

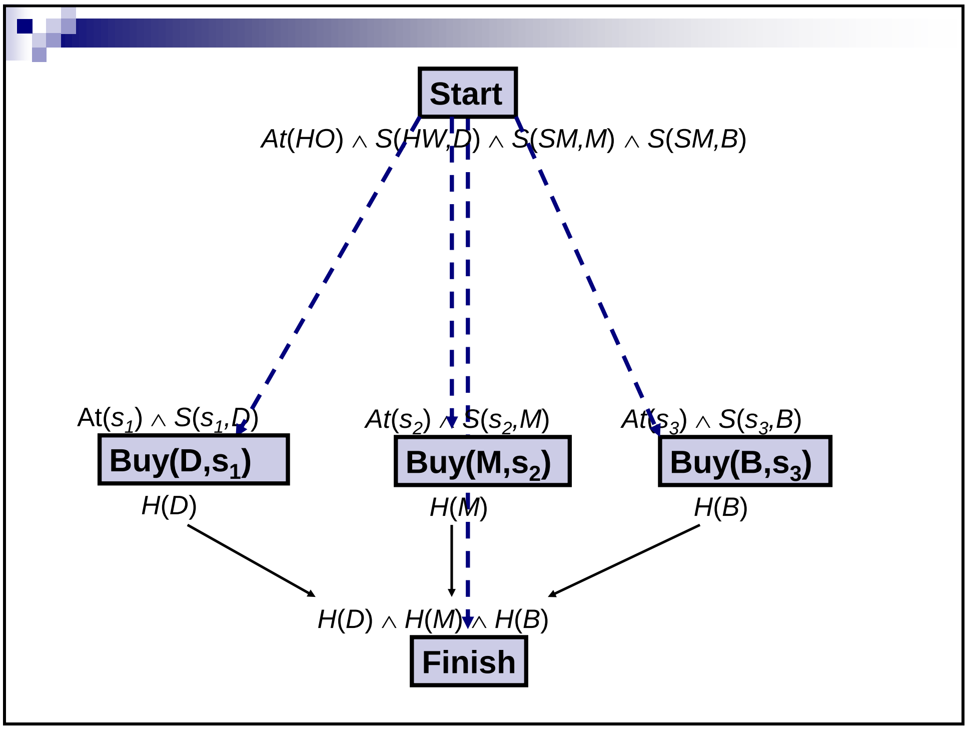
Resolving threats

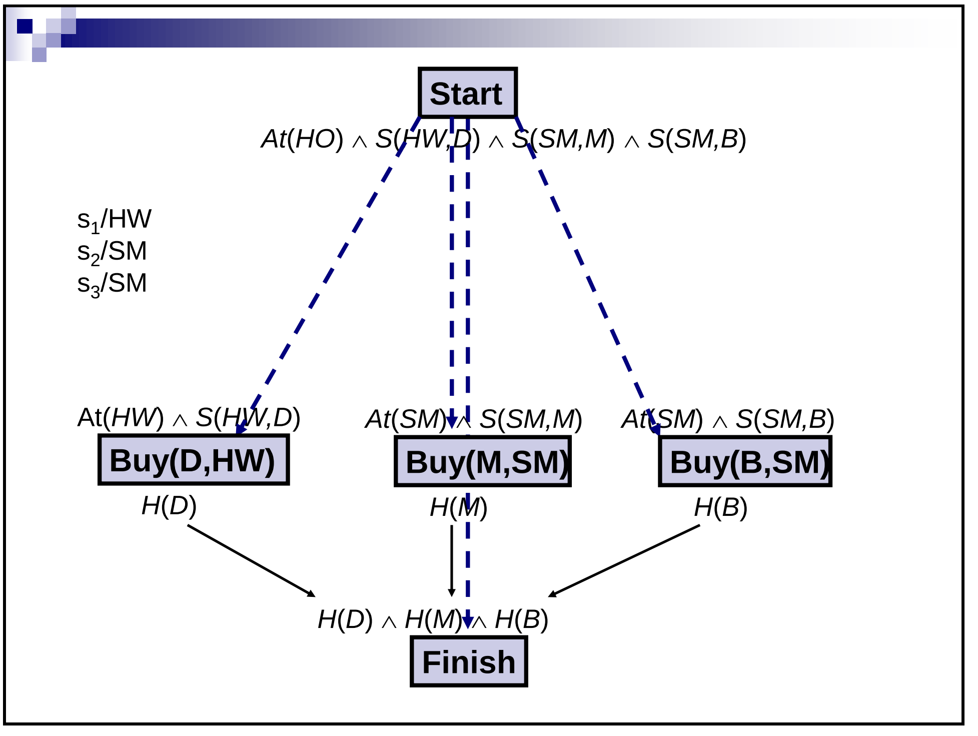
- A step S threatens a causal link $S_i \rightarrow_c S_j$ iff $\neg$ c $\in$ effects(S) and it's possible that $S_i < S < S_j$
- For each threat
  - Choose
    - Promote S : $S < S_i < S_j$
    - Demote S : $S_i < S_j < S$
  - If resulting plan is inconsistent, then Fail (backtrack)
- Threats with variables
  - S is a threat if there is any instantiation of the variables that makes $\neg c \in$ effects(S)
  - Negative binding

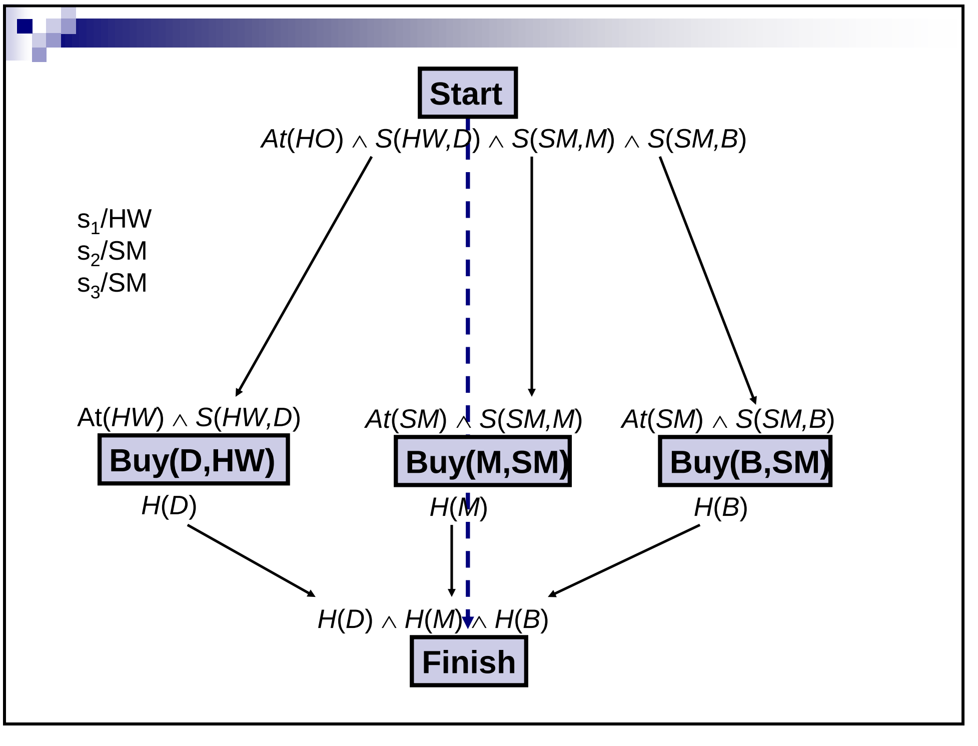
STRIPS example

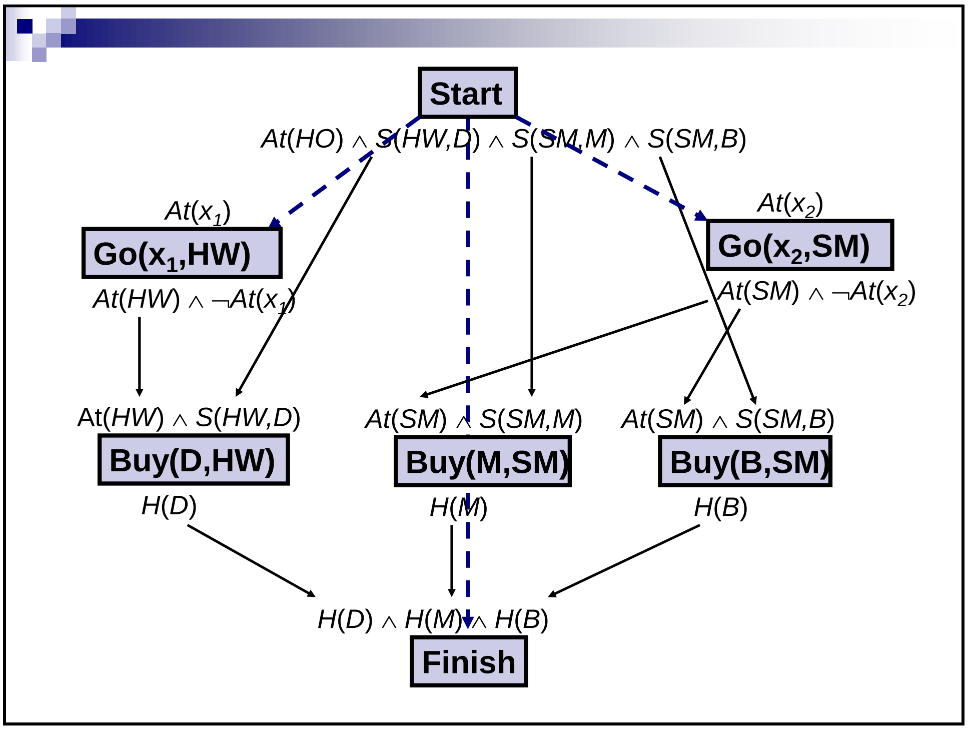
- Action
  - Buy(x, store)
    - Pre: At(store), Sells(store, x)
    - Eff: Have(x)
  - Go(x, y)
    - Pre: At(x)
    - Eff: At(y), ¬At(x)
- Goal
  - Have(Milk) ∧ Have(Banana) ∧ Have(Drill)
- Start
  - At(Home) ∧ Sells(SM, Milk) ∧ Sells(SM, Banana) ∧ Sells(HW, Drill)

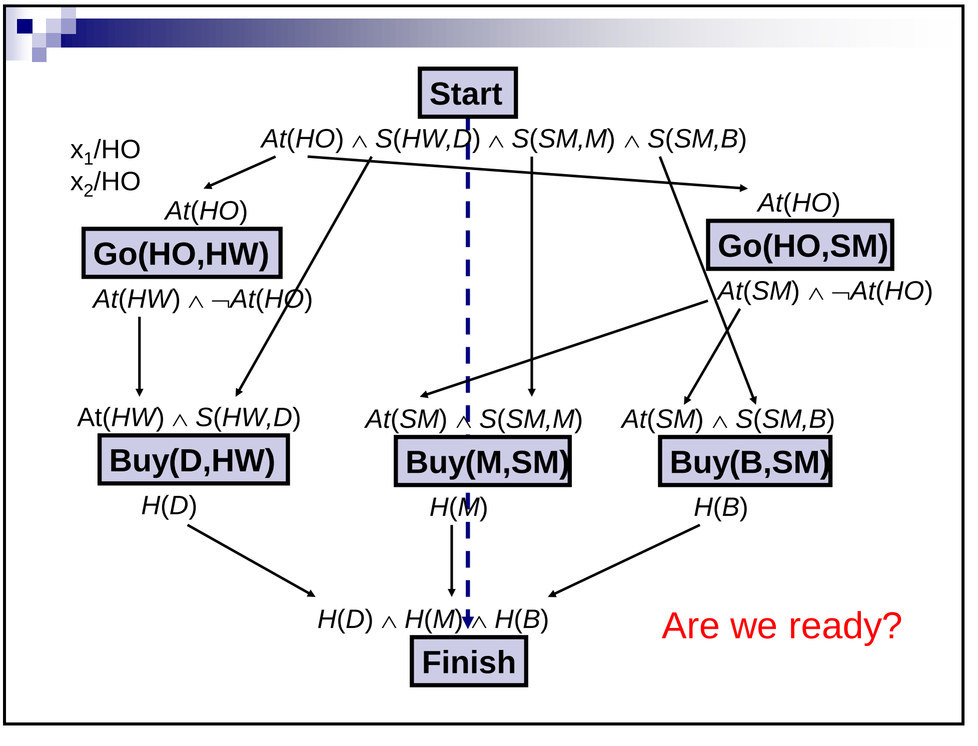

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
H(D) ∧ H(M) ∧ H(B)
Finish

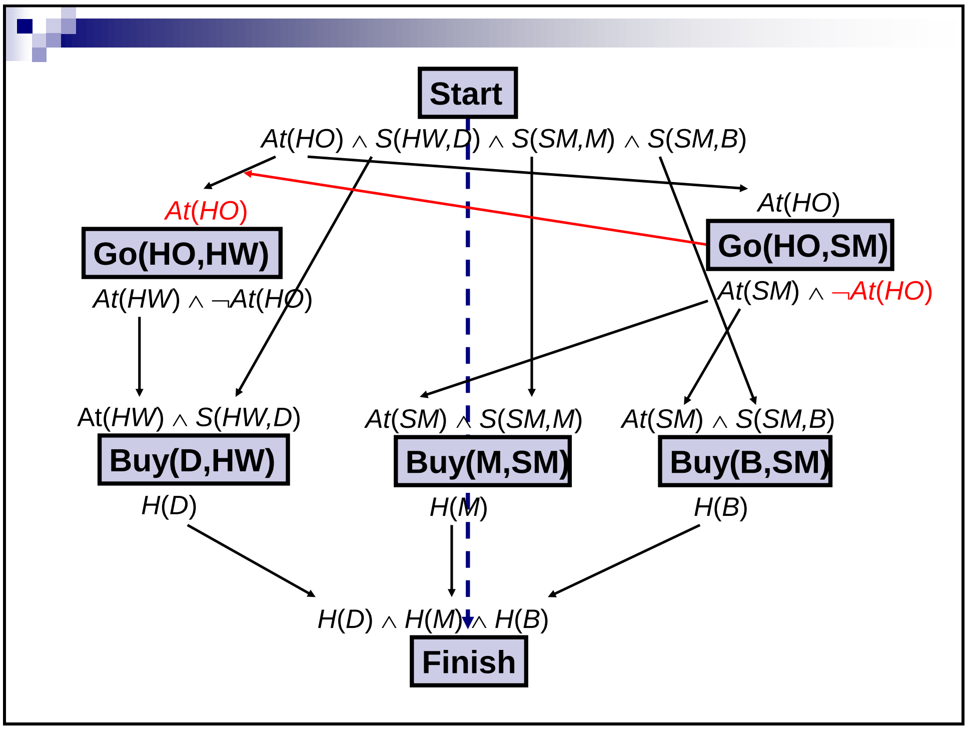

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(s1) ∧ S(s1,D)
Buy(D,s1)
H(D)
At(s2) ∧ S(s2,M)
Buy(M,s2)
H(M)
At(s3) ∧ S(s3,B)
Buy(B,s3)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish

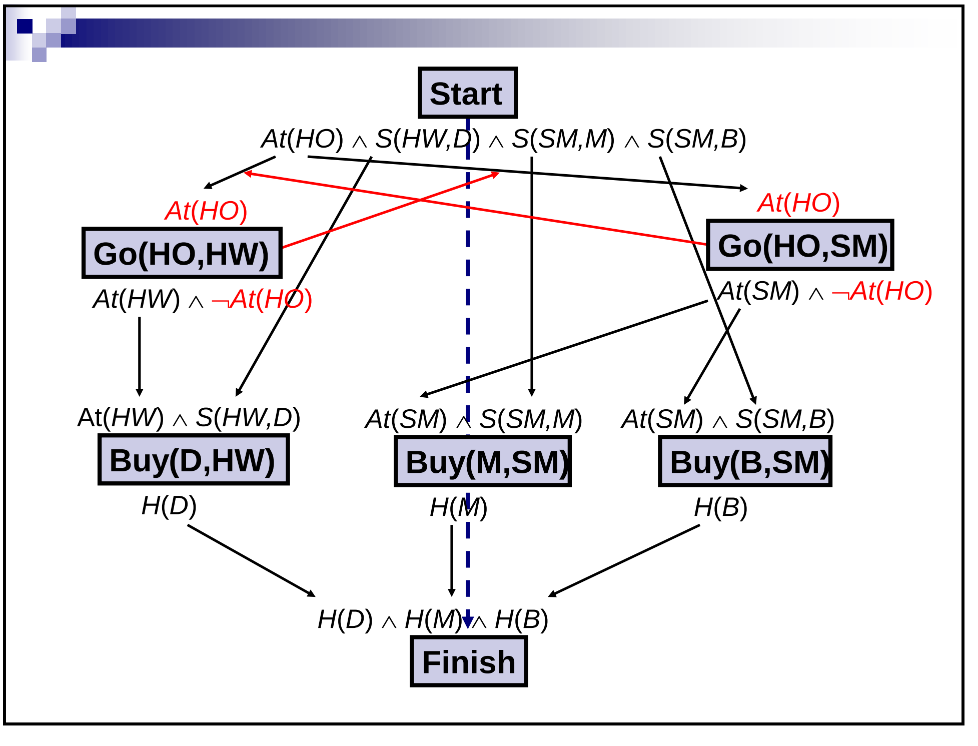

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
s1/HW
s2/SM
s3/SM
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish

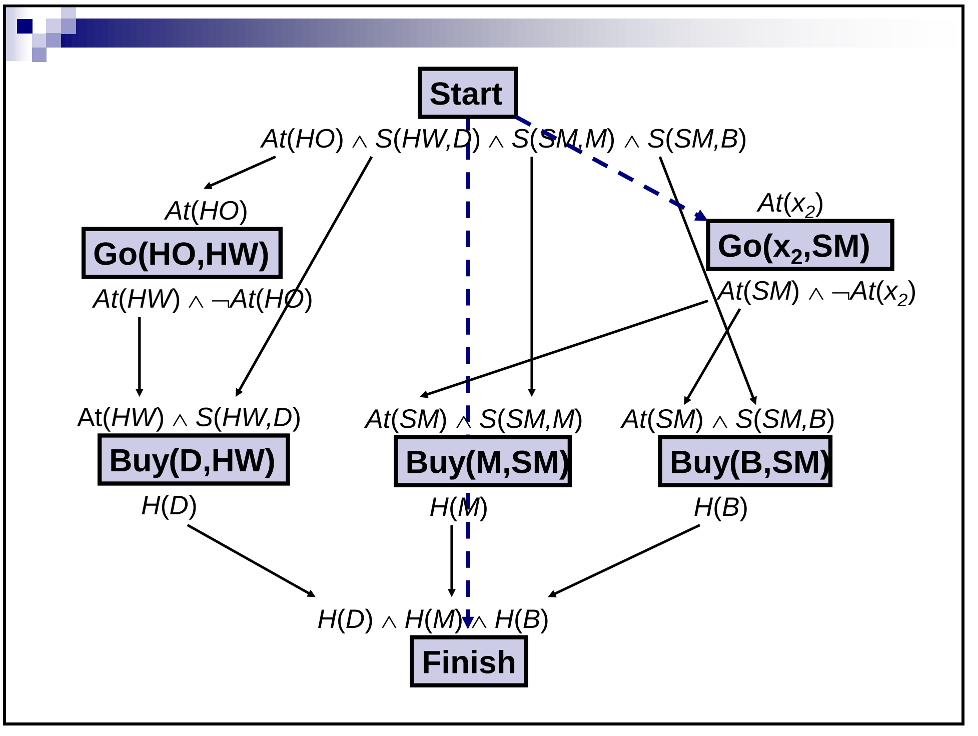

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
s1/HW
s2/SM
s3/SM
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish

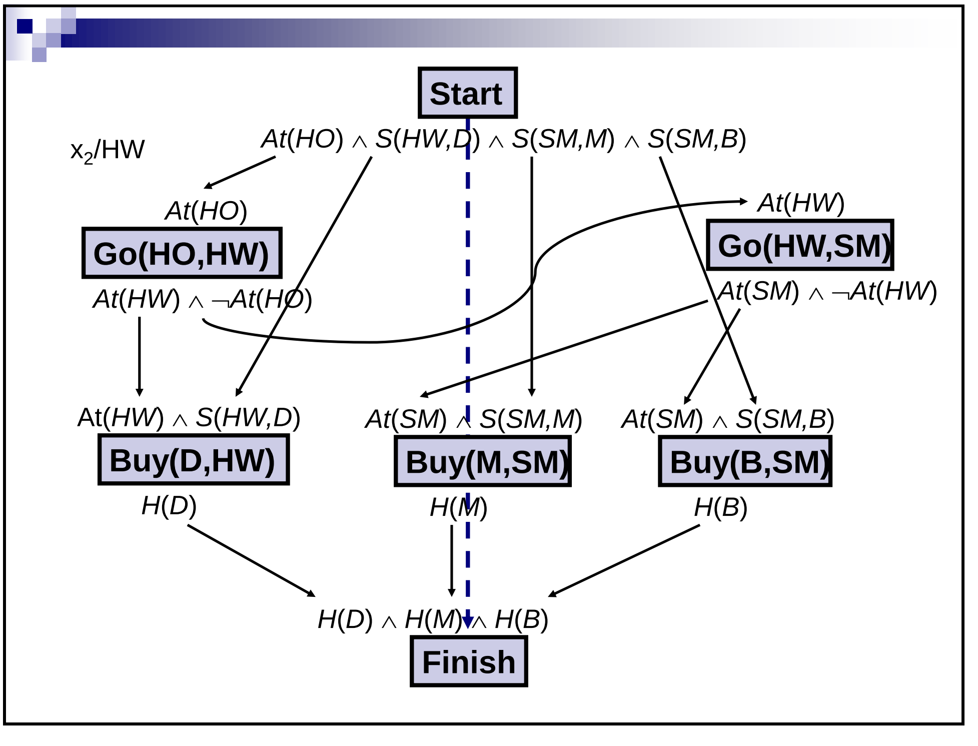

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(x_1)
Go(x_1,HW)
At(HW) ∧ ¬At(x_1)
At(x_2)
Go(x_2,SM)
At(SM) ∧ ¬At(x_2)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish

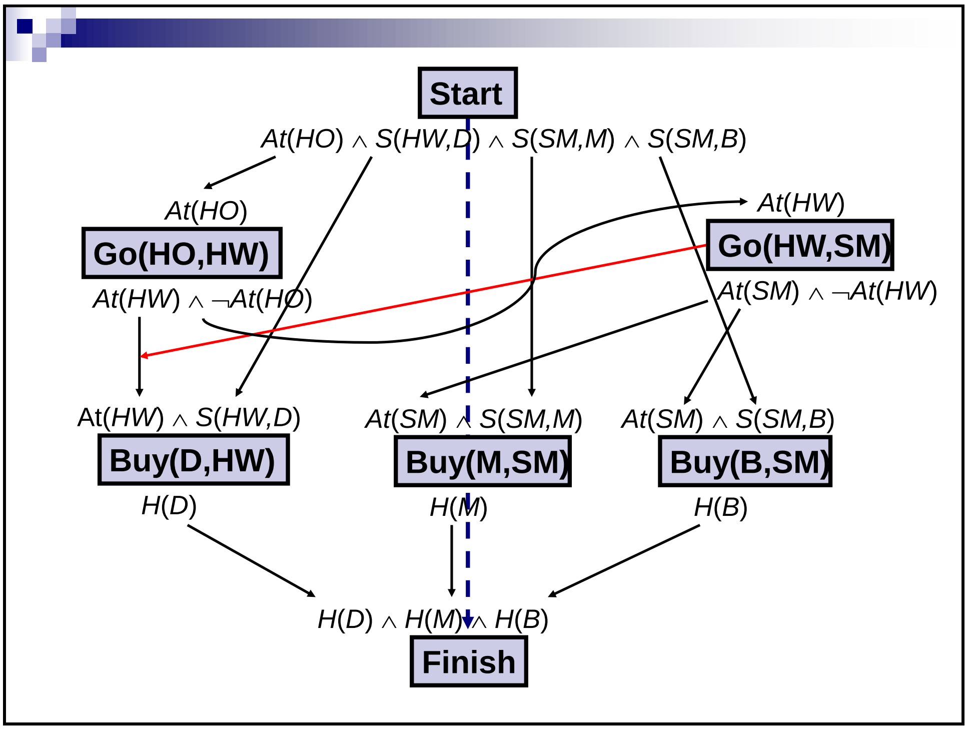

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
x_1/HO
x_2/HO
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(HO)
Go(HO,SM)
At(SM) ∧ ¬At(HO)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish
Are we ready?


Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(HO)
Go(HO,SM)
At(SM) ∧ ¬At(HO)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish

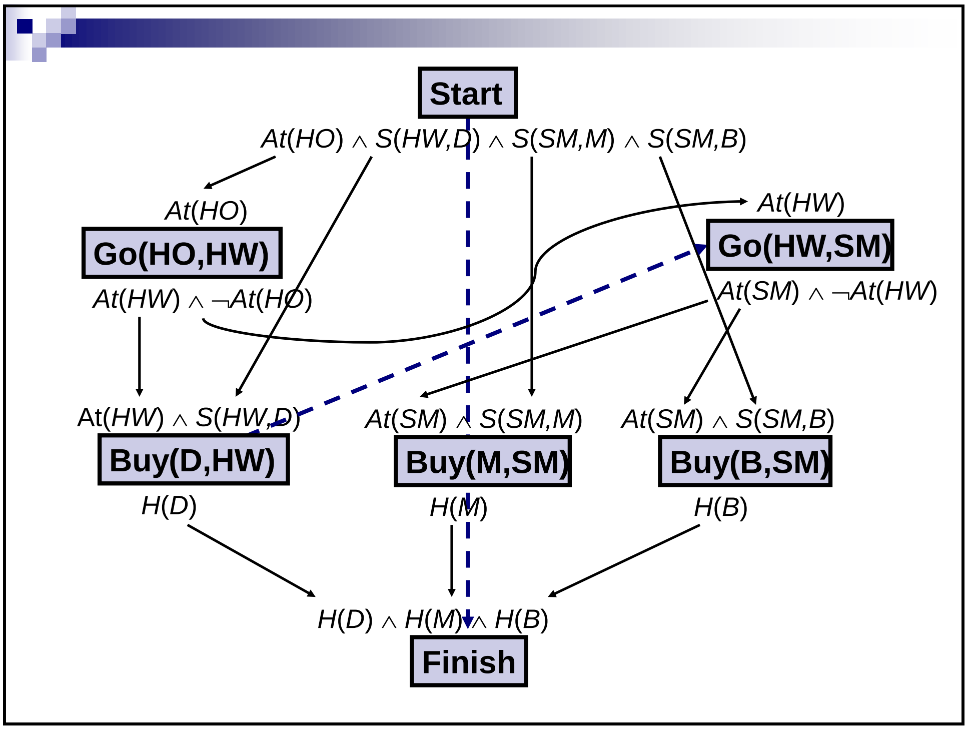

Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(HO)
Go(HO,SM)
At(SM) ∧ ¬At(HO)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish


Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(x2)
Go(x2,SM)
At(SM) ∧ ¬At(x2)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish


Start
x2/HW
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(HW)
Go(HW,SM)
At(SM) ∧ ¬At(HW)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish



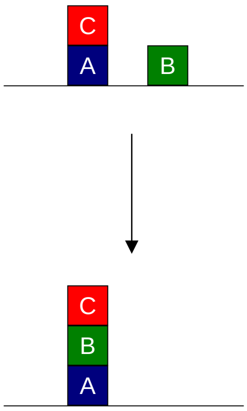
Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(HW)
Go(HW,SM)
At(SM) ∧ ¬At(HW)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish



Start
At(HO) ∧ S(HW,D) ∧ S(SM,M) ∧ S(SM,B)
At(HO)
Go(HO,HW)
At(HW) ∧ ¬At(HO)
At(HW)
Go(HW,SM)
At(SM) ∧ ¬At(HW)
At(HW) ∧ S(HW,D)
Buy(D,HW)
H(D)
At(SM) ∧ S(SM,M)
Buy(M,SM)
H(M)
At(SM) ∧ S(SM,B)
Buy(B,SM)
H(B)
H(D) ∧ H(M) ∧ H(B)
Finish

Sussman anomaly

- Subgoal dependence
  - Goal: on(A,B) $\wedge$ on(B,C)
- Exercise
  - Objects: A, B, C, T
  - Predicates
    - on(x,y), clear(x)
  - Operators
    - move(x,y,z)



Operators

- Move(x,y,z)
  - Pre: on(x,y), clear(x), clear(z)
  - Eff: on(x,z), clear(y), ¬on(x,y), ¬clear(z)
- How do we move to the table?
- Move2T(x,y)
  - Pre: on(x,y), clear(x)
  - Eff: on(x,T), clear(y), ¬on(x,y)

Operators

- Move(x,y,z)
  - Pre: on(x,y), clear(x), clear(z), block(z)
  - Eff: on(x,z), clear(y), ¬on(x,y), ¬clear(z)
- How do we move to the table?
- Move2T(x,y)
  - Pre: on(x,y), clear(x)
  - Eff: on(x,T), clear(y), ¬on(x,y)

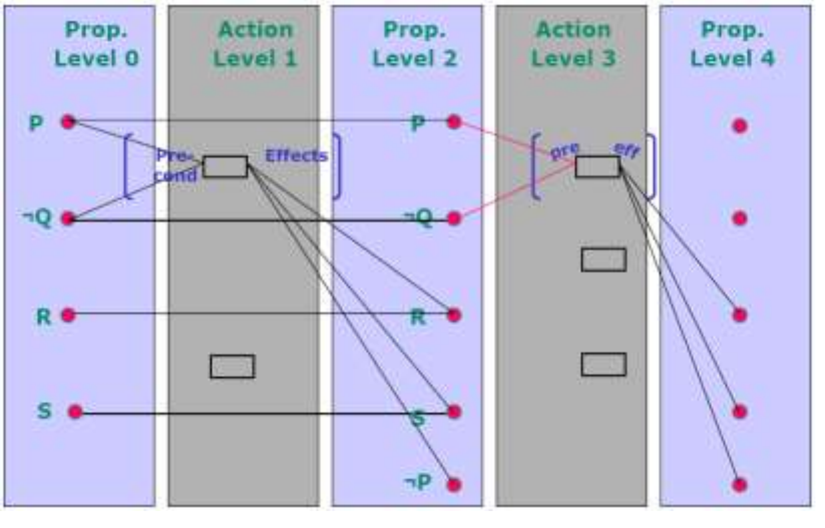
Limitations of the STRIPS language

- Hierarchical planning
  - Generating complex plans often requires abstract planning over increasingly detailed search spaces
- Complex state conditions
  - STRIPS variables are limited in their complexity
  - There is no quantification and no conditional statements
- Representing time
  - The STRIPS framework assumes that everything happens instantly
  - Not possible to represent durations, deadlines, time windows, etc.
- Resource limitations
  - There is no way to represent the amount of available workers, equipment, money, etc. or constraints on them

Graph Plan

- POP
  - "Human-like", but very slow
  - Efficiency hard to evaluate
- Graph Plan
  - Simplified planning model
    - propositional planner (no variables → no matching)
  - Bigger – separate propositions are needed for every combination of arguments
  - Efficient algorithm
  - Complexity between scheduling and planning

Planning graph



Planning graph

- Main idea
  - Construct a graph of possible outcomes

Graph Plan algorithm

- Resembles iterative DFS

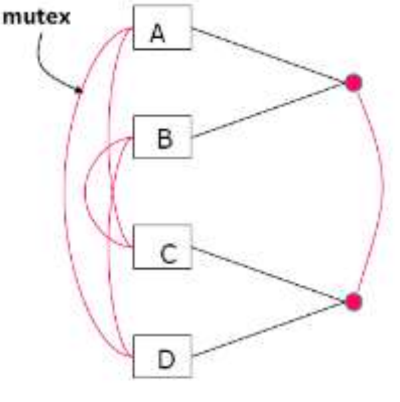
1. Make a plan graph of depth *k*
2. Search for a solution
3. If succeed, return a plan
4. Else *k* := *k* + 1
5. Go to step 1

Mutually exclusive actions

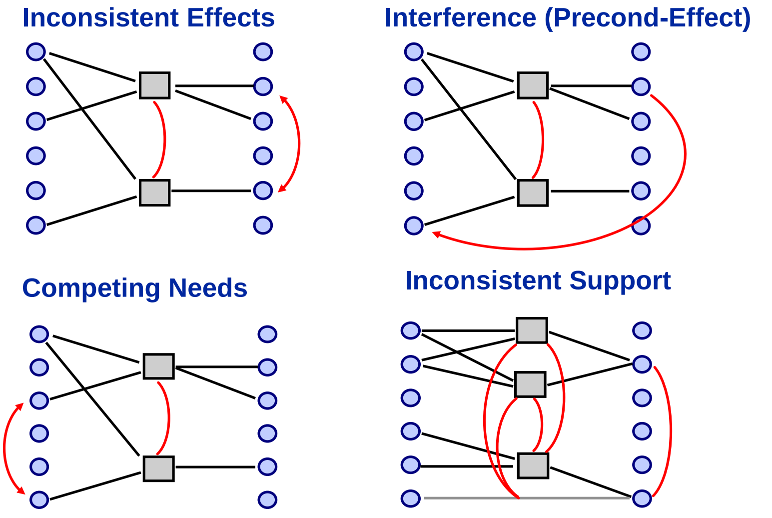
- Two action instances at level *i* are mutex if
  - Inconsistent effects
    - effect of one action is negation of effect of another
  - Interference
    - one action deletes the precondition of the other
  - Competing needs
    - the actions have preconditions that are mutex at level *i* - 1

Mutually exclusive propositions

- Two propositions at level *i* are mutex if
  - Negation
    - they are negations of one another
  - Inconsistent support
    - all ways of achieving the propositions at level *i* - 1 are pairwise mutex

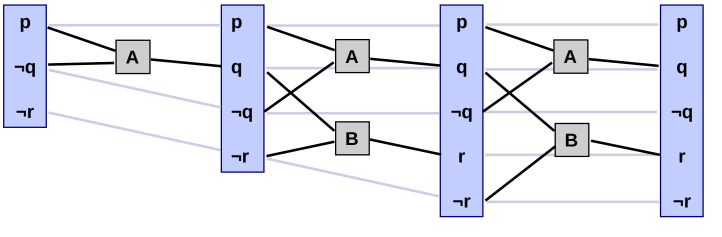


Overview of mutual exclusion classes



Trends with new layers

- Propositions monotonically increase
- Actions monotonically increase
- Proposition mutex relationships monotonically decrease
- Action mutex relationships monotonically decrease



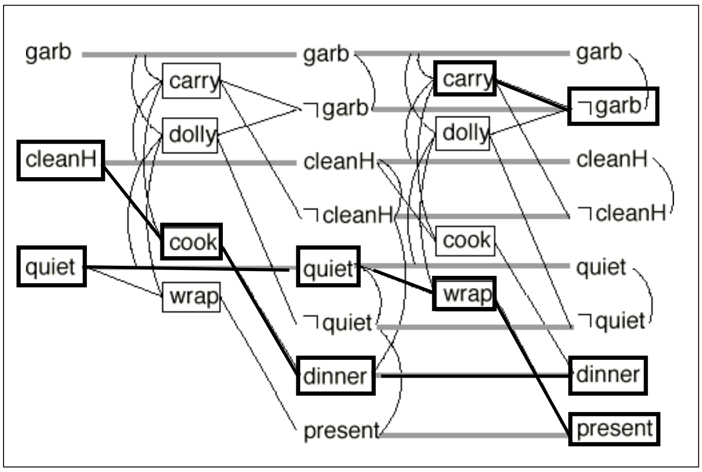
Solution extraction

- If all the literals in the goal appear at the deepest level and not mutex, then search for a solution for each subgoal at level *i*
  - For each subgoal at level *i*
    - Choose an action to achieve it
    - If it's mutex with another action, Fail
  - Repeat for preconditions at level *i* - 2

Example: Dinner date

- Initial conditions: garbage $\wedge$ cleanHands $\wedge$ quiet
- Goal: dinner $\wedge$ present $\wedge$ $\neg$ garbage
- Actions:
  - Cook — precondition: cleanHands; effect: dinner
  - Wrap — precondition: quiet; effect: present
  - Carry — precondition: -; effect: $\neg$ garbage $\wedge$ $\neg$ cleanHands
  - Dolly — precondition; effect: $\neg$ garbage $\wedge$ $\neg$ quiet

Search for a solution plan



Extensions

- Lots of time optimizations
- Disjunctive preconditions
- Universally quantified (almost :) preconditions and effects
- Conditional planning

Other approaches

- Hierarchical planning
- SATPlan
  - Reduces planning problem to satisfiability problem
  - Strongly related to GraphPlan
- FOPL like planning
  - Using structural information and heuristics
- Introducing uncertainty
  - Learning world dynamics
  - Conditional planning
  - Replanning
- Universal planning

Bayesian networks

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- What is intelligence?
- Agent model
- Problem solving by search
  - Non-informed, informed search strategies
  - Search in two player games
- Constraint satisfaction problems
- Planning

Outline

- Uncertainty vs. probability
- Bayes' rule
- Independence…
- Combining evidence
- Bayesian Networks
  - Connections
  - Independence

Motivation

- To calculate every possible probability joint probability distributions are needed
- But: given N propositional variables, there are 2^N joint probabilities

- Solution: exploit independencies in the domain

Bayes' rule

- Commutativity
  - $P(A \wedge B) = P(B \wedge A)$
  - $P(A) * P(B \mid A) = P(B) * P(A \mid B)$

$$P(B \mid A) = P(A \mid B) * P(B) / P(A)$$

- Example
  - P(disease | symptom) = P(symptom | disease) * P(disease) / P(symptom)
  - High fever (HF), diphtheria (D)
  - P(D | HF) = P(HF | D) * P(D) / P(HF)

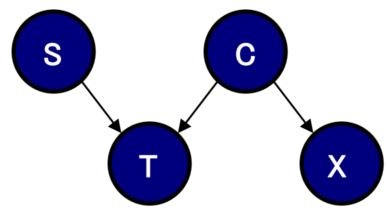
Bayes' rule

- E - evidence
- H_i - hypotheses

$$P(H_i \mid E) = \frac{P(E \mid H_i)P(H_i)}{P(E)} = \frac{P(E \mid H_i)P(H_i)}{\sum_{k=1}^{n} P(E \mid H_k)P(H_k)}$$

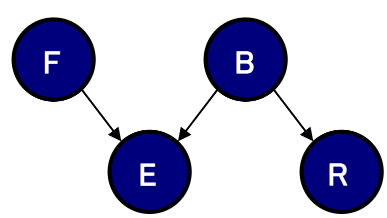
Conditional independence

- A and B are conditionally independent given C iff
  - $P(A \wedge B \mid C) = P(A \mid C) * P(B \mid C)$
  - $P(A \mid B,C) = P(A \mid C)$
  - $P(B \mid A,C) = P(B \mid C)$
- Examples
  - Toothache, spot, cavity



Conditional independence

- A and B are conditionally independent given C iff
  - $P(A \wedge B \mid C) = P(A \mid C) * P(B \mid C)$
  - $P(A \mid B,C) = P(A \mid C)$
  - $P(B \mid A,C) = P(B \mid C)$
- Examples
  - Toothache, spot, cavity
  - Engine, radio, battery



Everyday probability

- *Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations.*

- Which is more likely?
  (1) Linda is a bank teller.
  (2) Linda is a bank teller and is active in the feminist movement.

Conjugation fallacy

- Amos Tversky and Daniel Kahneman, 1983
- "Extension versus intuitive reasoning: The conjunction fallacy in probability judgment"

- 85% of people chose option 2, although $P(A) \geq P(A,B)$

Exercises

- Show that
  (1) P(A) $\geq$ P(A,B)
  (2) P(A | B) + P(¬A | B) = 1

- Write an expression for P(A | B,C) in terms of P(B | A,C)!

Combining evidence

- T: toothache X: spot on X-ray C: cavity

$$P(C \mid T, X) = \frac{P(T, X \mid C)\,P(C)}{P(T, X)}$$

- If *T* and *X* are conditionally independent given *C*, then

$$P(C \mid T, X) = \frac{P(T \mid C)\,P(X \mid C)\,P(C)}{P(T, X)}$$

Normalizing factor

$$P(C \mid T,X) + P(\neg C \mid T,X) = 1$$

$$\frac{P(T \mid C)P(X \mid C)P(C)}{P(T,X)} + \frac{P(T \mid \neg C)P(X \mid \neg C)P(\neg C)}{P(T,X)} = 1$$

$$P(T \mid C)P(X \mid C)P(C) + P(T \mid \neg C)P(X \mid \neg C)P(\neg C) = P(T,X)$$

Combining evidence

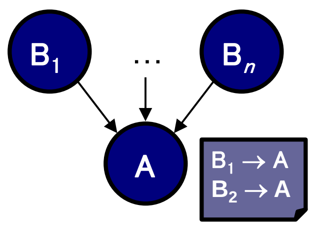
$$P(C \mid T,X) = \frac{P(T,X \mid C)P(C)}{P(T,X)} =$$

$$= \frac{P(T \mid C)P(X \mid C)P(C)}{P(T,X)} =$$

$$= \frac{P(T \mid C)P(X \mid C)P(C)}{P(T \mid C)P(X \mid C)P(C) + P(T \mid \neg C)P(X \mid \neg C)P(\neg C)}$$

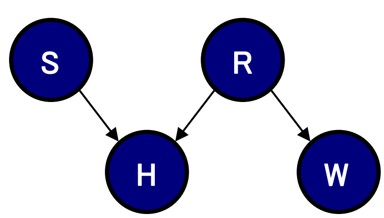
Bayesian networks

- Set of nodes representing random variables
- Set of directed arcs (forming a DAG) expressing direct influence between nodes
- Every node A with parents $B_1, \ldots, B_n$ has the conditional probabilities $P(A \mid B_1, \ldots, B_n)$ specified



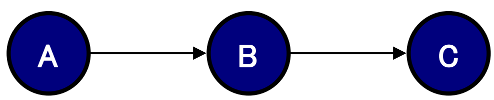
Causal component

- "Sherlock Holmes wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet as well. So, he concludes, it must have rained."



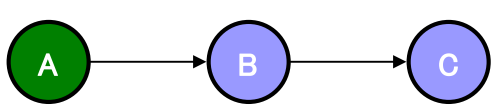
Serial connections

- 1. Forward serial connection
  - Transmit evidence from A to C through unless B is instantiated (its truth value is known)



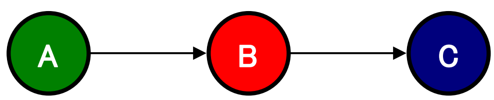
Serial connections

- 1. Forward serial connection
  - Transmit evidence from A to C through unless B is instantiated (its truth value is known)



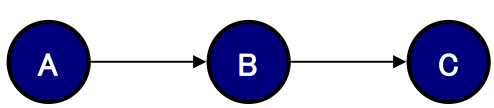
Serial connections

- 1. Forward serial connection
  - Transmit evidence from A to C through unless B is instantiated (its truth value is known)



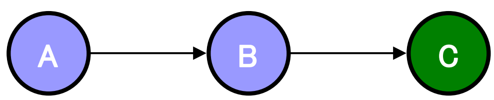
Serial connections

- 1. Forward serial connection
  - Transmit evidence from A to C through unless B is instantiated (its truth value is known)
- 2. Backward serial connection
  - Transmit evidence from C to A through unless B is instantiated (its truth value is known)



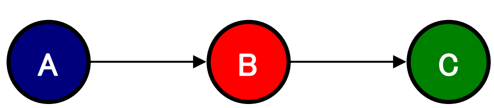
Serial connections

- 1. Forward serial connection
  - Transmit evidence from A to C through unless B is instantiated (its truth value is known)
- 2. Backward serial connection
  - Transmit evidence from C to A through unless B is instantiated (its truth value is known)



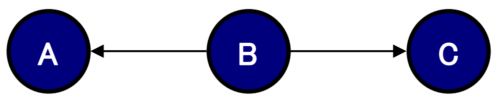
Serial connections

- 1. Forward serial connection
  - Transmit evidence from A to C through unless B is instantiated (its truth value is known)
- 2. Backward serial connection
  - Transmit evidence from C to A through unless B is instantiated (its truth value is known)



Diverging connection

- Transmit evidence through B unless it is instantiated

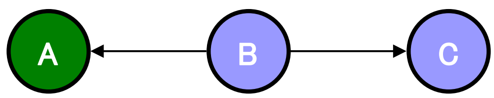


Diverging connection example – Icy roads

- Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, “It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch.” But, his secretary says, “No, the roads are not icy, look at the window.” So, he says, “I guess I better wait for Holmes.”

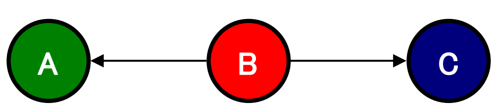
Diverging connection

- Transmit evidence through B unless it is instantiated
- Knowing about A will tell us something about C



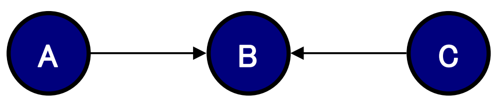
Diverging connection

- Transmit evidence through B unless it is instantiated
- But, if we know B, then knowing about A will not tell us anything new about C, or vice versa



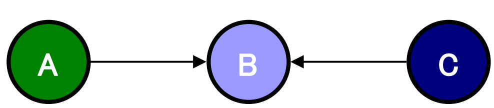
Converging connection

- Tricky case!
- Transmit evidence from A to C only if B or a descendant of B is instantiated



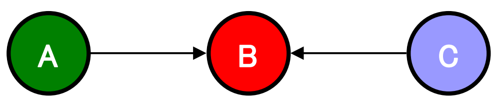
Converging connection

- Transmit evidence from A to C only if B or a descendant of B is instantiated
- Without knowing B, finding A does not tell us anything about C



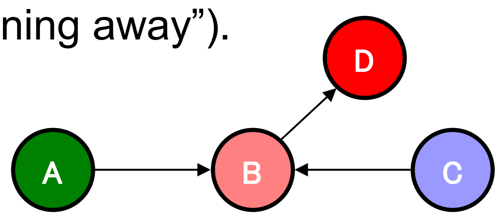
Converging connection

- Transmit evidence from A to C only if B or a descendant of B is instantiated
- If we see evidence for B, then A and C become dependent (potential for "explaining away").



Converging connection

- Transmit evidence from A to C only if B or a descendant of B is instantiated
- If we see evidence for B, then A and C become dependent (potential for "explaining away").

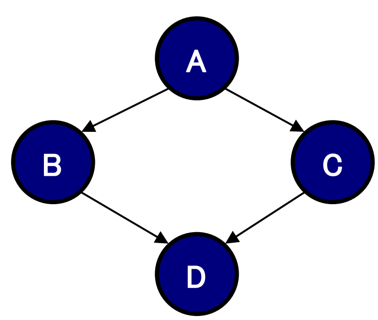


D-separation

- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - the connection is serial or diverging and V is known
  - the connection is converging and neither V nor any of its descendants is instantiated
- Two variables are d-connected iff they are not d-separated

D-separation exercise

- No instantiation
- A instantiated
- A and D instantiated
- B instantiated
- B and C instantiated



Solution

- No instantiation
  - A, D are d-connected (A-B-D connected, A-C-D connected)
  - B, C are d-connected (B-A-C connected, B-D-C blocked)
- A instantiated
  - B, C are d-separated (B-A-C blocked, B-D-C blocked)
- A and D instantiated
  - B, C are d-connected (B-A-C blocked, B-D-C connected)
- B instantiated
  - A, D are d-connected (A-B-D blocked, A-C-D connected)
- B and C instantiated
  - A, D are d-separated (A-B-D blocked, A-C-D blocked)

Outline

- Uncertainty vs. probability
- Bayes' Rule
- Conditional independence
- Combining evidence
- Bayesian Networks
  - Connections
  - D-separation

Inference in Bayesian networks

Artificial intelligence

Kristóf Karacs

PPKE-ITK

Recap

- Bayesian networks
- Combination of evidence
- Type of connections
- d-separation

Outline

- Efficient inference
  - D-separation theorem
  - Chain rule
- Quantitative inference
- Using joint distributions
- Variable elimination
- Multiply connected networks

Theorem

- If A and B are d-separated given an evidence e, then P(A | e) = P(A | B, e)
- Enables efficient inference

Chain rule

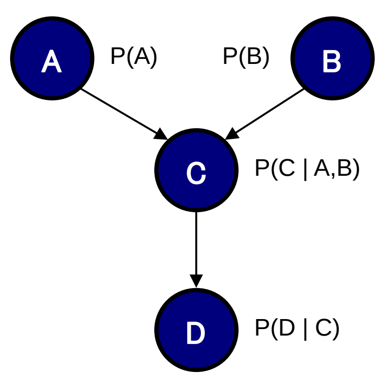
- Variables: $V_1, \ldots, V_n$
- Values: $v_1, \ldots, v_n$

$$\mathrm{P}(V_1 = v_1, V_2 = v_2, \ldots, V_n = v_n) =$$
$$= \prod_{i=1}^{n} P(V_i = v_i | parents(V_i))$$

Using the chain rule

- P(ABCD) = P(A=true, B=true, C=true, D=true)

- P(ABCD) =
  P(D|ABC) P(ABC) =
  P(D|C) P(ABC) =
  P(D|C) P(C|AB) P(AB) =
  P(D|C) P(C|AB) P(A) P(B)



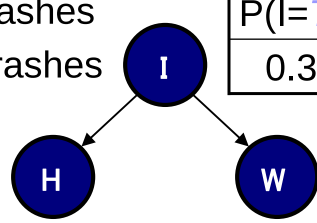
Icy roads

- Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, “It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch.” But, his secretary says, “No, the roads are not icy, look at the window.” So, he says, “I guess I better wait for Holmes.”

Icy roads – Conditional Probability Tables (CPT)

- I: Road is icy
- H: Holmes crashes
- W: Watson crashes

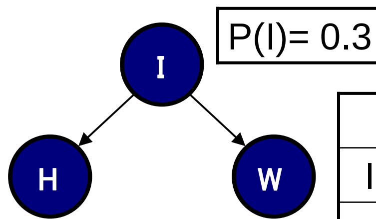
P(I=*T*)	P(I=*F*)
0.3	0.7



	P(H=*T* | I)	P(H=*F* | I)
I=*T*	0.8	0.2
I=*F*	0.1	0.9

	P(W=*T* | I)	P(W=*F* | I)
I=*T*	0.8	0.2
I=*F*	0.1	0.9

Icy roads – with numbers

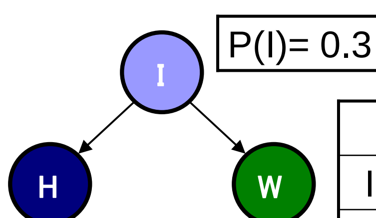


	P(H | I)
I	0.8
¬I	0.1

	P(W | I)
I	0.8
¬I	0.1

- P(W) = P(W | I) P(I) + P(W | ¬I) P(¬ I)
  = 0.8 * 0.3 + 0.1 * 0.7 = 0.31

Icy roads – with numbers



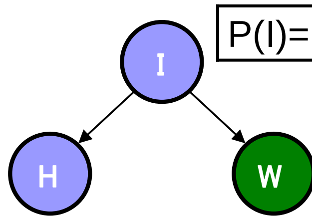
	P(H | I)
I	0.8
¬I	0.1

	P(W | I)
I	0.8
¬I	0.1

- P(I | W) = P(W | I) P(I) / P(W)
  = 0.8 * 0.3 / 0.31 = 0.77

Icy roads – with numbers

P(I)= 0.3



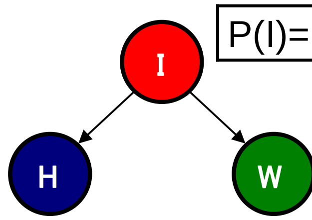
	P(H | I)
I	0.8
¬I	0.1

	P(W | I)
I	0.8
¬I	0.1

- P(H | W) =
  = P(H | W,I) P(I | W) + P(H | W,¬I) P(¬I | W)
  = P(H | I) P(I | W) + P(H | ¬I) P(¬I | W)
  = 0.8 * 0.77 + 0.1 * 0.23 = 0.639

Icy roads – with numbers

P(I)= 0.3



	P(H | I)
I	0.8
¬I	0.1

	P(W | I)
I	0.8
¬I	0.1

- P(H | W, ¬I) = P(H | ¬I)
  = 0.1

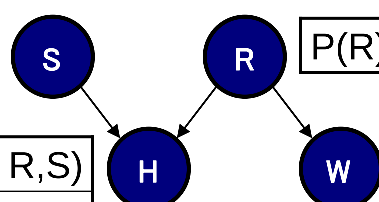
Wet lawns

- "Sherlock Holmes wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet as well. So, he concludes, it must have rained."

Wet lawns

P(S)= 0.1

P(R)= 0.2



	P(H | R,S)
R,S	1.0
R,¬S	1.0
¬R,S	0.9
¬R,¬S	0.1

	P(W | R)
R	1.0
¬R	0.2

- P(R | H), P(S | H), P(W | H)
- P(R | HW), P(S | HW)

Types of inference

- Exact inference
- Approximate inference

Possible queries

- $P(X=x_0 \mid E=e)$
- What value of x maximizes $P(X=x \mid E=e)$?
- Probability distribution $Pr(X \mid E=e)$

Using joint distribution

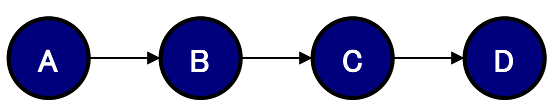
- Summing over variables not involved

$$P(d) = \sum_{ABC} P(a, b, c, d) =$$

$$= \sum_{a \in dom(A)} \sum_{b \in dom(B)} \sum_{c \in dom(C)} P(A = a \wedge B = b \wedge C = c \wedge D = d)$$

$$P(d|b) = \frac{P(b,d)}{P(b)} = \frac{\sum_{AC} P(a,b,c,d)}{\sum_{ACD} P(a,b,c,d)}$$

Variable elimination

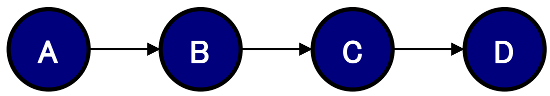


$$P(d) = \sum_{ABC} P(a, b, c, d) = \sum_{ABC} P(d|c)P(c|b)P(b|a)P(a)$$

$$= \sum_{C} \sum_{B} \sum_{A} P(d|c)P(c|b)P(b|a)P(a)$$

$$= \sum_{C} P(d|c) \sum_{B} P(c|b) \sum_{A} P(b|a)P(a)$$

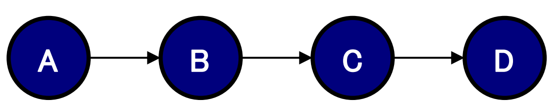
Variable elimination



$$P(d) = \sum_C P(d|c) \sum_B P(c|b) \underbrace{\sum_A P(b|a)P(a)}$$

$$\begin{bmatrix} P(b_1|a_1)P(a_1) & P(b_1|a_2)P(a_2) \\ P(b_2|a_1)P(a_1) & P(b_2|a_2)P(a_2) \end{bmatrix}$$

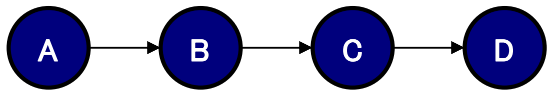
Variable elimination



$$P(d) = \sum_C P(d|c) \sum_B P(c|b) \underbrace{\sum_A P(b|a)P(a)}$$

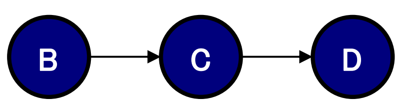
$$\begin{bmatrix} \sum_A P(b_1|a)P(a) \\ \sum_A P(b_2|a)P(a) \end{bmatrix}$$

Variable elimination



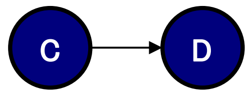
$$P(d) = \sum_{C} P(d|c) \sum_{B} P(c|b) \underbrace{\sum_{A} P(b|a) P(a)}_{f_1(b)}$$

Variable elimination



$$P(d) = \sum_{C} P(d|c) \underbrace{\sum_{B} P(c|b) f_1(b)}_{f_2(c)}$$

Variable elimination



$$P(d) = \sum_{c} P(d|c) f_2(c)$$

Variable elimination algorithm

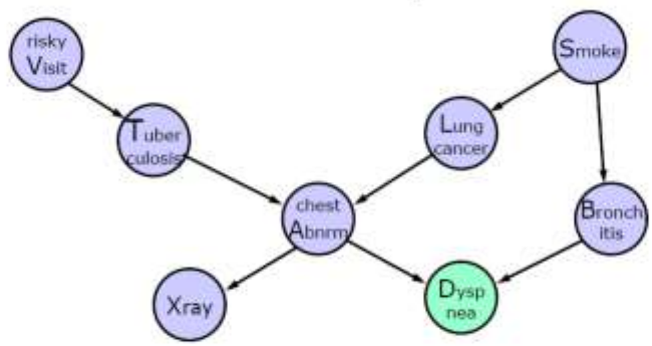
Given a Bayesian network and an elimination order for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} K \sum_{X_m} \prod_{j} P(x_j | parents(x_j))$$

For i = m downto 1

- Remove all factors that mention X_i
- Multiply those factors, getting a value for each combination of mentioned variables
- Sum over X_i
- Put this new factor into the factor set

Example



$$\Pr(d) = \sum_{A,B,L,T,S,V} \Pr(d \mid a,b)\Pr(a \mid t,l)\Pr(b \mid s)\Pr(l \mid s)\Pr(s)\Pr(t \mid v)\Pr(v)$$

Example

$$\Pr(d) = \sum_{A,B,L,T,S,V} \Pr(d \mid a,b)\Pr(a \mid t,l)\Pr(b \mid s)\Pr(l \mid s)\Pr(s)\Pr(t \mid v)\Pr(v)$$

$$= \sum_{A,B} \Pr(d \mid a,b) \sum_{L,T} \Pr(a \mid t,l) \sum_{S} \Pr(b \mid s)\Pr(l \mid s)\Pr(s) \sum_{V} \Pr(t \mid v)\Pr(v)$$

$$= \sum_{A,B} \Pr(d \mid a,b) \sum_{L,T} \Pr(a \mid t,l)\, f_1(t) \sum_{S} \Pr(b \mid s)\Pr(l \mid s)\Pr(s)$$

$$= \sum_{A,B} \Pr(d \mid a,b) \sum_{L} f_2(b,l) \sum_{T} \Pr(a \mid t,l)\, f_1(t)$$

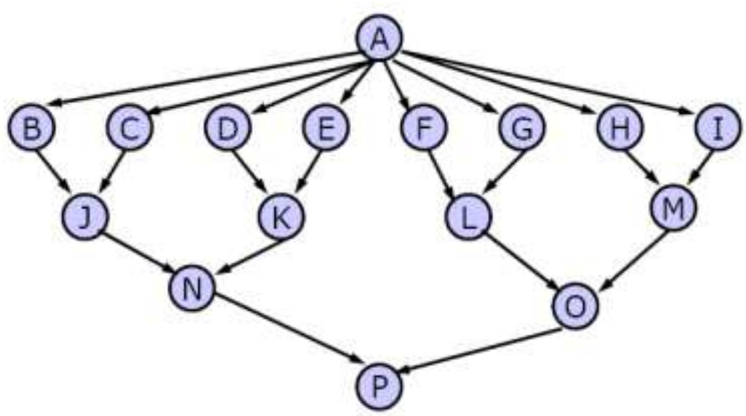
$$= \sum_{A,B} \Pr(d \mid a,b) \sum_{L} f_2(b,l)\, f_3(a,l) = \sum_{A}\sum_{B} \Pr(d \mid a,b)\, f_4(a,b) = \sum_{A} f_5(a)$$

Variable elimination

- Generally requires exponential time ($O(n^2 b^k)$)
- Bad elimination order can generate huge factors
- Finding the best one is NP-hard
  - Heuristic: choose the variable that results in smallest next factor (greedy method)

- Linear time for singly connected networks (polytree)
  - There is only one (undirected) path between any two nodes
  - Always eliminate variables with no parents

Exercise



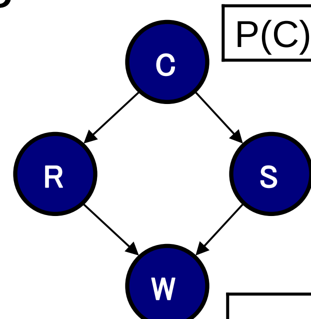
Inference in multiply connected DAGs

- **Clustering**
  - Transforms the network to a probabilistically equivalent polytree by joining certain nodes in the network
  - Useful when computing many a posteriori probabilities
- **Stochastic simulation (Monte Carlo)**
  - Estimates the probabilities by generating samples using the probability distribution defined by the network

Clustering

P(C)= 0.5

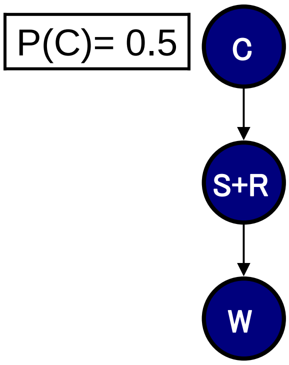
	P(R | C)
C	0.8
¬C	0.2



	P(S | C)
C	0.1
¬C	0.5

	P(W | R,S)
R,S	1.0
R,¬S	1.0
¬R,S	0.9
¬R,¬S	0.1

Clustering



	P(R+S = x | C)			
	R,S	R,¬S	¬R,S	¬R,¬S
C	0.08	0.72	0.02	0.18
¬C	0.1	0.1	0.4	0.4

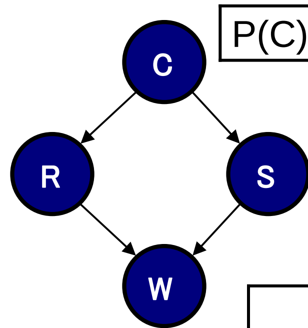
	P(W | R,S)
R,S	1.0
R,¬S	1.0
¬R,S	0.9
¬R,¬S	0.1

Monte Carlo (sampling)

- Iterative sampling by making draws for each variable
  - Draws are based on CPTs
  - Start from root nodes
  - For children use the drawn values of parents
- After many rounds relative frequencies can be calculated

Monte Carlo

P(C)= 0.5



	P(R | C)
C	0.8
¬C	0.2

	P(S | C)
C	0.1
¬C	0.5

	P(W | R,S)
R,S	1.0
R,¬S	1.0
¬R,S	0.9
¬R,¬S	0.1

$P(W \mid C) = P(C,W) / P(C)$

$P^*(W \mid C) = \#C,W / \#C$

Importance sampling

- Problem
  - Rare events will not be well represented
- Solution: Importance sampling
  - Considers biased distributions (towards rare values)
  - Output is weighted to correct for the bias
  - Weights are determined by likelihood ratios
  - Fast convergence
  - Can handle huge networks

Exercises

- Wet Lawns with numbers
- Variable elimination
- Monte Carlo

Outline

- Efficient inference
  - D-separation theorem
  - Chain rule
- Quantitative inference
- Using joint distributions
- Variable elimination
- Multiply connected networks

Machine Learning

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Recap

- Concepts related to intelligence
- Agent model
- Problem solving by search
- Strategies in games
- Inference in First Order Predicate Logic

Outline

- Supervised vs. unsupervised vs. …
- Logical inference schemes
- Inductive learning
  - ID3 algorithm
  - Version spaces
  - ~~Inductive logic programming~~
- Learning theory

Machine Learning and AI

- Improve task performance through observation, teaching
- Acquire knowledge automatically for use in a task
- Learning is a key component in intelligence

Learning

- Supervised, unsupervised, semi-supervised, reinforcement
- Representation trade-off
  - Efficiency vs. expressive power
- There exist learning methods for several representation schemes

Applications

- Data mining
  - Big data, web mining
- Language/speech
  - Machine translation, text summarization, grammars
- Medical
  - Assessment of illness severity
- Vision
  - Face recognition, digit recognition, outdoor scene recognition
- Security
  - Intrusion detection, network traffic, credit fraud
- Social networks
  - Email traffic

Logical inference schemes

- *Deduction*: formal logical reasoning
  - Premises: 1. All men are mortal. 2. Aristotle is a man.
  - Conclusion: Aristotle is mortal.
- *Induction*: generalization
  - Premise: The sun has risen in the east every morning up until now.
  - Conclusion: The sun will also rise in the east tomorrow.
- *Abduction*: choosing an explanation
  - Premise: 1. Flu causes fever. 2. Peter has fever.
  - Conclusion: Peter has flu.

Approaches

- Inductive learning
  - Discovering general concepts from a limited set of examples (experience)
  - From a formal point of view the obtained knowledge is invalid
  - Supervised
    - Given input data as pairs of (x_i, $f(x_i)$)
    - Generate a hypothesis for $f()$
  - Unsupervised
- Analytic or deductive learning
  - Based on explanations

Inductive learning

- Decision trees
- Version-spaces
    - More general, but less efficient
- Inductive logic programming

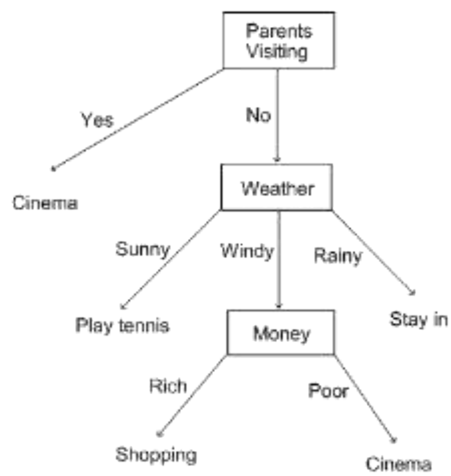
General Approach

- Formulate task
- Prior model (parameters, structure)
- Obtain data
- What representation should be used? (attribute/value pairs)
- Annotate data
- Learn/refine model with data (training)
- Use model for classification or prediction on unseen data (testing)
- Measure accuracy

Decision trees

- Measurements
- Nodes
- Node selection: ID3 algorithm
  - Maximizing information gain
  - Measure for information gain of an attribute: expected value of the information given by it
- Pruning

Sample decision tree



Information theory

- S: set of measurements
- A: an attribute with domain $V = \{v_1, \dots, v_i, \dots, v_n\}$
- S_v: set of measurements for which $A = v$
- Entropy:

$$H(S) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

- Gain:

$$G(S,A) = H(S) - \sum_{v \in V} \frac{|S_v|}{|S|} H(S_v)$$

Example

	weather	got HW?	friend comes?	excursion
S_1	nice	Yes	Yes	Yes
S_2	cloudy	No	No	No
S_3	rainy	No	Yes	Yes
S_4	cloudy	Yes	Yes	Yes

- 3 input attributes
- 4 measurements

Information gain for attribute 'weather'

- $H(S) = -3/4 \log_2(3/4) - 1/4 \log_2(1/4) = 0.811$
- $|S_{nice}|/|S| * H(S_{nice}) = 1/4 * (-1/1 \log_2(1/1) - 0/1 \log_2(0/1)) = 1/4 * (1*0 - 0) = 0$
- $|S_{cloudy}|/|S| * H(S_{cloudy}) = 2/4*(-1/2 \log_2(1/2) - 1/2 \log_2(1/2)) = 1/2 * (-1/2 * (-1) - 1/2 * (-1)) = 1/2$
- $|S_{rainy}|/|S| * H(S_{rainy}) = 0$
- $G(S,A) = 0.811 - (0+0.5+0) = 0.311$

The ID3 algorithm

- Given a set of examples, S
  - Described by a set of attributes A_i
  - Categorised into categories c_j

1. Put the attribute the has the highest information gain in the root node (attribute A_1)
2. For each value v_i that A can take
   - Draw a branch and label each with corresponding v_i

The ID3 algorithm

3. For each branch with value v_i
   - If S_{vi} only contains examples in category c, then put that category as a leaf node in the tree
   - If S_{vi} is empty, then put the default category (which contains the most examples from S) as a leaf node in the tree
   - Otherwise construct subtree_i by recursively calling decision tree with S_v, all attributes

Text Classification

- Is text_i a new finance article?

Positive

Negative

Dow Closes Down 38, Nasdaq Gains 3

General Sense of Caution Pushes Stocks Mostly Lower; Dow Closes Down 38, Nasdaq Gains 3

The Associated Press

April 8, 8:56 pm ET — Investors concerned about the deteriorating situation in Iraq looked past solid earnings from General Electric Co. and Yahoo! Inc. Thursday, sending stocks mostly lower and leaving Wall Street with a loss for the holiday-shortened week.

The Dow Jones industrial average was down 38.12, or 0.4 percent, at 10,442.03. The Dow had gained more than 60 points in early trading before falling back. Broader stock indicators were narrowly mixed. The Standard & Poor's 500 index was down 1.21, or 0.1 percent, at 1,139.32, while the Nasdaq composite index gained 2.64, or 0.1 percent, to 2,052.88.

While GE, which posted earnings in line with Wall Street estimates, is seen as a gauge of the overall economy due to the conglomerate's diverse businesses, the threat of increased violence in Iraq and possible consequences from terrorism kept investors from making large bets.

"The market is now in a duel between good economic numbers, good earnings, and the situation in Iraq," said Peter Cardillo, chief strategist and senior vice president at S.W. Bach & Co. "With the market trading at the upper end of its trading range for the year, it induces people to take some money off the table."

Trading was quiet and volume light, with many investors and traders taking time off for the holidays. The stock market was scheduled to close for Good Friday.

For the week, the Dow lost 0.3 percent, while the S&P 500 and Nasdaq both dropped

20 attributes

- Investors 2
- Dow 2
- Jones 2
- Industrial 1
- Average 3
- Percent 5
- Gain 6
- Trading 8
- Broader 5
- stock 5
- Indicators 6
- Standard 2
- Rolling 1
- Nasdaq 3
- Early 10
- Rest 12
- More 13
- first 11
- Same 12
- The 30

UConn defense knocks out Georgia Tech for second title

By Jack Carey, USA TODAY

SAN ANTONIO — Dominating inside, outside and especially defensively, Connecticut roared to its second men's basketball national championship in six years Monday night, rolling past Georgia Tech 82-73.

Jim Calhoun, center, and the rest of the UConn Huskies celebrate the school's second national title.

By Sue Ogrocki, AP

With the UConn women playing Tennessee for the national crown Tuesday night in New Orleans, Connecticut has a chance to become the first school to hold both basketball titles in the same year.

In its unexpected run to the final, Georgia Tech (28-10) had five games decided in the closing seconds, but Monday's contest was over early. (**Related item:** Box score)

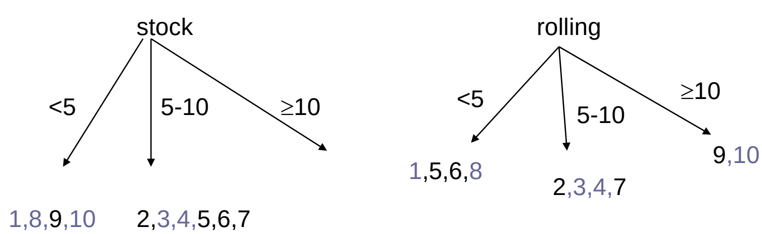
With All-America center Emeka Okafor (24 points, 15 rebounds) virtually unstoppable down low and smooth guard Ben Gordon scoring 21, the Huskies (33-6) were too much for Tech, which couldn't find the range until late against UConn's aggressive defenders.

20 attributes

- Men's
- Basketball
- Championship
- UConn Huskies
- Georgia Tech
- Women
- Playing
- Crown
- Titles
- Games
- Rebounds
- All-America
- early
- rolling
- Celebrates
- Rest
- More
- First
- The
- same

Example

	stock	rolling	the	class
1	0	3	40	other
2	6	8	35	finance
3	7	7	25	other
4	5	7	14	other
5	8	2	20	finance
6	9	4	25	finance
7	5	6	20	finance
8	0	2	35	other
9	0	11	25	finance
10	0	15	28	other



Gain(stock) = 1 - [0.4 * H(0.1, 0.3) + 0.6 * H(0.4, 0.2)] =
= 1 - [0.4 ((-0.1 * -3.32) - (0.3 * -1.74)) +
+ 0.6 ((-0.4 * -1.32) - (0.2 * -2.32))] =
= 1-[0.303 + 0.5952] = 0.105

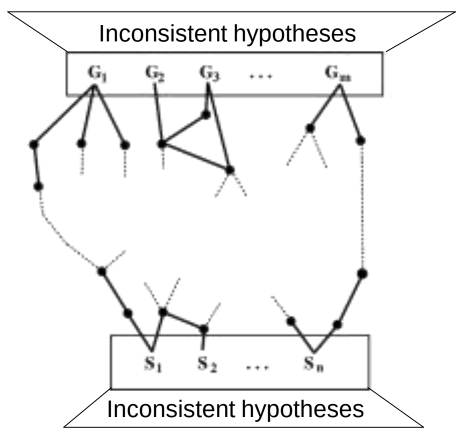
Gain(rolling) = 1 - [0.4 * H(0.5, 0.5) + 0.4 * H(0.5, 0.5) +
+ 0.2 * H(0.5, 0.5)] = 0

Issues

- Representation
  - How to map from a representation in the domain to a representation used for learning?
- Training data
  - How can training data be acquired?
- Amount of training data
  - How well does the algorithm do as we vary the amount of data?
- Which attributes influence learning most?
- Does the learning algorithm provide insight into the generalizations made?

Version space learning

- A technique for learning concepts
- Continuously maintains the set of consistent hypotheses

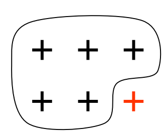


Version space learning

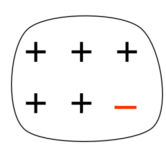
- Least commitment principle
  - G: most general set
  - S: most specific set
- Initialization
  - G: True
  - S: False
- A hypothesis H is **consistent** if H is
  - more specific than some element of G
  - and more general than some element of S

Handling inconsistence

- If S_i is false negative
  - replace S_i by all direct generalizations that classifies e as positive and is more specific than some element of G

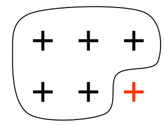


- If S_i is false positive
  - remove S_i from S

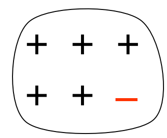


Handling inconsistence

- If G_i is false negative
  - remove G_i from G



- If G_i is false positive
  - replace G_i by all direct specializations that classifies e as negative and is more general than some element of S



Example

- Training set
  - S_1: {Japan, Honda, blue, 1980, economy} +
  - S_2: {Japan, Toyota, green, 1970, sports} –
  - S_3: {Japan, Toyota, blue, 1990, economy} +
  - S_4: {USA, Chrysler, red, 1980, economy} –
  - S_5: {Japan, Honda, white, 1980, economy +

Example

S_1: {Japan, Honda, blue, 1980, economy}	+
S_2: {Japan, Toyota, green, 1970, sports}	–
S_3: {Japan, Toyota, blue, 1990, economy}	+
S_4: {USA, Chrysler, red, 1980, economy}	–
S_5: {Japan, Honda, white, 1980, economy}	+

- S1: G = {G1:(*,*,*,*,*)}
  S = {S1:(Japan, Honda, blue, 1980, economy)}
- S2: G = {G1:(*,Honda,*,*,*), G2:(*,*, blue,*,*), G3:(*,*,*, 1980,*), G4:(*,*,*,*, economy)}
  S = {S1:(Japan, Honda, blue, 1980, economy)}
- S3: G = {G2:(*,*,blue,*,*), G4:(*,*,*,*, economy)}
  S = {S1:(Japan,*, blue,*, economy)}
- S4: G = {G2:(*,*, blue,*,*), G4:(Japan,*,*,*, economy)}
  S = {S1:(Japan,*, blue,*, economy)}
- S5: G = {G4:(Japan,*,*,*, economy)}
  S = {S1:(Japan,*,*,*, economy)}
- The concept is "Japanese economy car"

Termination

- G ≠ S, but no more samples
- G = S → a single winner hypothesis
- S or G is empty
  - Inconsistent samples (may be due to noise)
  - Insufficient attributes
  - Chosen language is incapable to express the concept

Problems of VS learning

- Cannot handle noisy data
  - Version spaces collapses
- Unlimited disjunction
  - Wave fronts will not meet

Learning summary

What is the space of hypotheses to be considered when learning?

- If too large: no actual knowledge can be gained (generalization is not possible)
- If too small: it might not include the target function

Learning theory

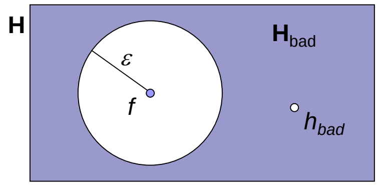
- Correctness of hypothesis *h*() has to be evaluated without knowing *f*(), the function to be learned
- A sufficiently large dataset can ensure an approximately good result with a high probability
- How many samples are needed to evaluate the correctness of a hypothesis?
  - *Probably Approximately Correct* learning (PAC learning)

Probably approximately correct (PAC) learning

- **X**: set of all possible examples
- *D*: distribution of examples
- **H**: set of possible hypotheses
- *m*: # of examples in training set

- Looking for an *h*()∈**H** being close to *f*()∈**H**

Hypothesis space



- error(h) = $P(h(x) \neq f(x) \mid x$ is drawn from D)
- Hypothesis h is approximately correct if error(h) $\leq \varepsilon$
- P(h_{bad} is consistent with m examples) $\leq (1 - \varepsilon)^m$

PAC-learning

- P($\mathbf{H}_{bad}$ contains a consistent hypothesis) $\leq |\mathbf{H}_{bad}|(1 - \varepsilon)^m \leq |\mathbf{H}|(1 - \varepsilon)^m$
- Let δ be an upper bound for this
  - Sample complexity function of the hypothesis space

$$m(\epsilon, \delta) \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathbf{H}|\right)$$

  - If the complexity of the hypothesis space is less than exponential, then the function is learnable

PAC-learning

- For Boole functions: $|\mathbf{H}| = 2^{2^n}$
- Sample complexity grows as
  For $\varepsilon = \delta = 10^{-4}$
  $n = 2$ $m \geq 5{,}000$
  $n = 10$ $m \geq 3{,}000{,}000$
- Solution
  - Searching in space of simple solutions
  - Restricting the language of hypotheses

Summary

- Supervised vs. unsupervised vs. …
- Logical inference schemes
- Inductive learning
  - ID3 algorithm
  - Version spaces
  - Inductive logic programming
- Learning theory

Learning Bayesian Networks

Artificial intelligence
Kristóf Karacs
PPKE-ITK

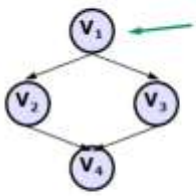
Learning Bayesian Networks

- Sources for Bayesian Nets
  - Human experts
  - Data (measurement)
- What can be learned?
  - Structure
  - Probabilities
- Typical case: structure from expert, probability from data

Learning probabilities

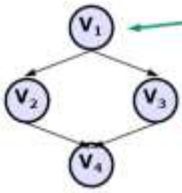
- Given a data set $D = \{<v_{11}, \ldots, v_{m1}>, \ldots, <v_{1k}, \ldots, v_{mk}>\}$
- m: # of nodes k: # of samples
- Elements are assumed to be independent given M
- Maximum likelihood estimate
  - Find model M that maximizes $\Pr(D|M)$

Estimating Conditional Probabilities



$$P(V_1) \approx \frac{\#(V_1 = true)}{k}$$

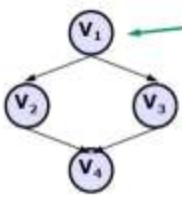
Estimating Conditional Probabilities



$$P(V_1) \approx \frac{\#(V_1{=}true)}{k}$$

$$P(V_3|V_1) \approx \frac{\#(V_3{=}true \wedge V_1{=}true)}{\#(V_1{=}true)}$$

Estimating Conditional Probabilities

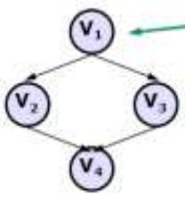


$$P(V_1) \approx \frac{\#(V_1{=}true)}{k}$$

$$P(V_3|V_1) \approx \frac{\#(V_3{=}true \wedge V_1{=}true)}{\#(V_1{=}true)}$$

$$P(V_3|\neg V_1) \approx \frac{\#(V_3{=}true \wedge V_1{=}false)}{\#(V_1{=}false)}$$

Estimating Conditional Probabilities



$$P(V_1) \approx \frac{\#(V_1=true)+1}{k+2}$$

$$P(V_3|V_1) \approx \frac{\#(V_3=true \wedge V_1=true)+1}{\#(V_1=true)+2}$$

$$P(V_3|\neg V_1) \approx \frac{\#(V_3=true \wedge V_1=false)+1}{\#(V_1=false)+2}$$

Measuring goodness of fit

- Calculating Pr(*D*|*M*)

$$\Pr(D|M) = \prod_j \Pr(v^j|M)$$
$$= \prod_j \prod_i \Pr(N_i = v_i^j | Parents(N_i), M)$$

- Log likelihood

$$\log \Pr(D|M) = \log \prod_j \prod_i \Pr(N_i = v_i^j | Parents(N_i), M)$$
$$= \sum_j \sum_i \log \Pr(N_i = v_i^j | Parents(N_i), M)$$

Learning the structure

- For a fixed structure, counting estimates of the CPT converge to the maximum likelihood model
- What if we get to pick the structure as well?
- In general, the best model will have no conditional independence relationships
- Undesirable, for reasons of overfitting

Scoring metric

- What if we want to vary the structure?
- A network with conflicting properties
  - good fit to data: log likelihood
  - low complexity: total number of parameters
- Try to maximize scoring metric, by varying M (structure and parameters) given D

$$\log \Pr(D|M) - \alpha \, \#(M)$$

Search in structure space

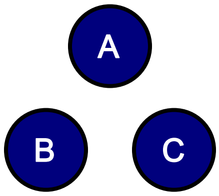
- Brute-force method cannot be applied
- Local search in structure space
  - Starting with some initial structure
  - Operators: add, delete, or reverse an arc
  - Choosing the next state
    - Evaluation of candidates using maximum likelihood parameters
    - Hill climbing, simulated annealing
    - No directed cycles should occur in the structure

Initialization possibilities

- No arcs
- With a random ordering $V_1 \ldots V_n$
  - variable V_i has all parents $V_1 \ldots V_{i-1}$
  - variable V_i has parents randomly chosen from $V_1 \ldots V_{n-1}$
- Best tree-network
  - maximum-weight spanning tree based on pairwise mutual information between every pair of variables
  - polynomial time algorithm

Bayesian net structure example

- Domain with 3 binary nodes
- Measurement data
  - {(0,1,1),(0,1,1),(1,0,0)}
- Consider three structure candidates
  - M_1 {}
  - M_2 {A→B, A→C}
  - M_3 {B→A, C→A}



- 1. Calculate parameter estimates for the CPTs!
- 2. Calculate $\Pr(D|M_i)$ for each structure with Bayesian correction!

Fuzzy logic

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Fuzzy logic

- Lotfi Zadeh
- Concept: truth values may apply partially
- Not able to express uncertainty (rather truthfulness)
- Logical statements are derived from natural language statements

Fuzzy sets

- Sets are identified by linguistic identifiers
  - “tall”, “young”, “bigger”
- Grade of membership

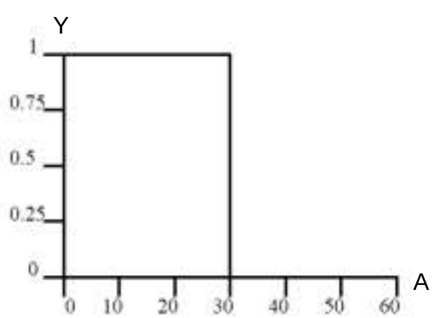
$$\mu(x) \quad , \quad x \in U \quad , \quad 0 \leq \mu(x) \leq 1$$

- Fuzzy set A

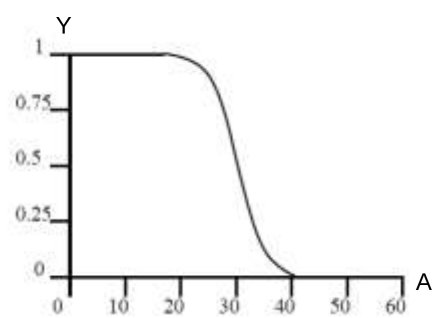
$$A = \{\langle x, \mu(x)\rangle\} = \{\langle x, \mu(x_1)\rangle, \langle x, \mu(x_2)\rangle, \ldots, \langle x, \mu(x_n)\rangle\}$$

Membership function

- Age – “young”



Crisp



Fuzzy

Types of membership functions

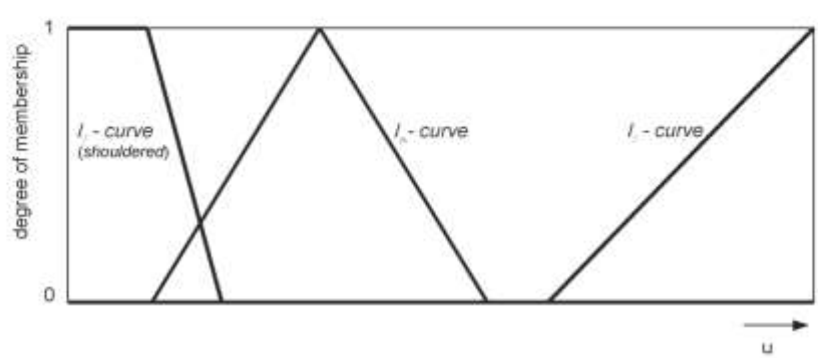
- Piecewise linear
  - Straight lines (increasing, decreasing)

$$\ell_{/}(x_\ell, x_r, x) = \begin{cases} 0, & x < x_\ell \\ \frac{x - x_\ell}{x_r - x_\ell}, & x_\ell \le x \le x_r \\ 1, & x > x_r \end{cases}$$

  - Triangular

$$\ell_{\wedge}(x_\ell, x_c, x_r, x) = \begin{cases} 0, & x < x_\ell \\ \frac{x - x_\ell}{x_c - x_\ell}, & x_\ell \le x \le x_c \\ 1 - \frac{x - x_c}{x_r - x_c}, & x_c \le x \le x_r \\ 0, & x > x_r \end{cases}$$

Linear curves



Types of membership functions

- Smooth curves
  - s-curve

$$s(x_0,b,x) = \begin{cases} 0, & x < x_0 - b \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{x-x_0}{b}\pi), & x_0 - b \le x \le x_0 \\ 1, & x > x_0 \end{cases}$$

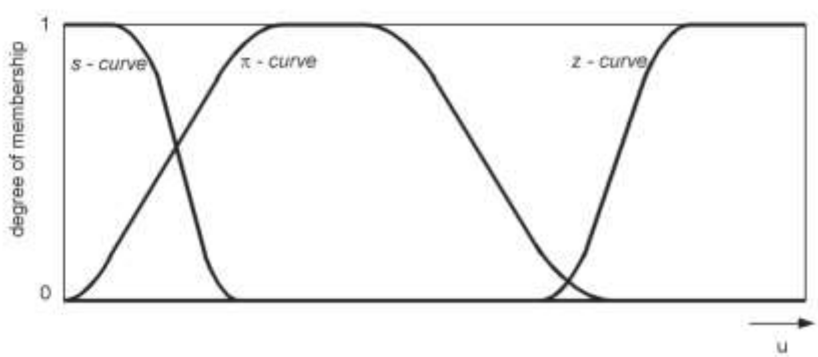
  - z-curve

$$z(x_0,b,x) = \begin{cases} 0, & x < x_0 \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{x-x_0}{b}\pi), & x_0 \le x \le x_0 + b \\ 1, & x > x_0 + b \end{cases}$$

  - π-curve

$$\pi(x_1,x_2,b,x) = min(s(x_1,b,x), z(x_2,b,x))$$

Smooth curves



Operations

Given $A = \{\langle x, \mu_A(x)\rangle\}$ and $B = \{\langle y, \mu_B(y)\rangle\}$ on a joint universe U.

- **fuzzy union**: $A \cup B = A \text{ or } B \triangleq A \max B$
  $\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$ for all $x \in U$
- **fuzzy intersection**: $A \cap B = A \text{ and } B \triangleq A \min B$
  $\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$ for all $x \in U$
- **fuzzy complement**: $\neg A = \text{not} A \triangleq 1 - A$
  $\mu_{\neg A}(x) = 1 - \mu_A(x)$ for all $x \in U$

Fuzzy intersection

- Universe (cylinder capacity): $U = \{1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$
  - **low consumption (LC)**

U	1.0	1.2	1.4	1.6	1.8	2.0
μ_{LC}	1.0	0.9	0.7	0.5	0.2	0.0

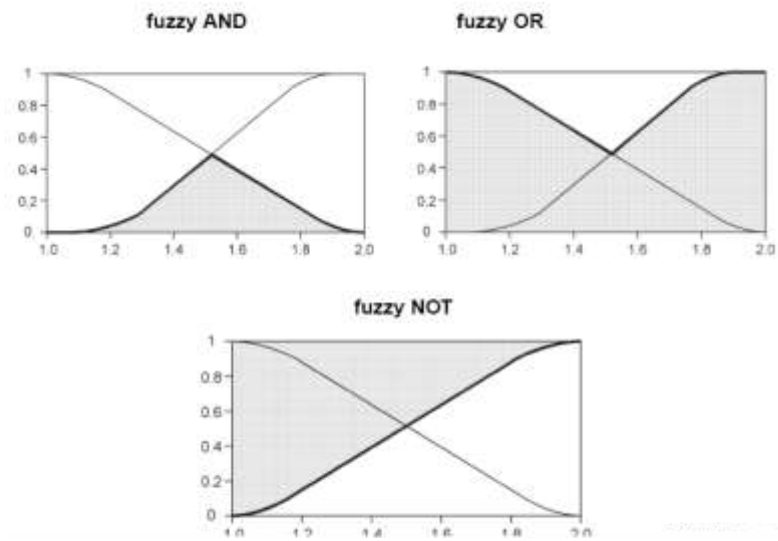
  - **high acceleration (HA)**

U	1.0	1.2	1.4	1.6	1.8	2.0
μ_{HA}	0.0	0.1	0.4	0.5	0.8	1.0

low consumption and *high acceleration*

U	1.0	1.2	1.4	1.6	1.8	2.0
μ_{LC}	1.0	0.9	0.7	0.5	0.2	0.0
μ_{HA}	0.0	0.1	0.4	0.5	0.8	1.0
$\min(\mu_{LC}, \mu_{HA})$	0.0	0.1	0.4	0.5	0.2	0.0

Operations on fuzzy sets



Algebraic properties

- Commutativity
  - a or b = b or a
  - a and b = b and a
- Associativity
  - (a or b) or c = a or (b or c)
  - (a and b) and c = a and (b and c)
- Distributivity
  - a or (b and c) = (a or b) and (b or c)
  - a and (b or c) = (a and b) or (b and c)
- DeMorgan rules
  - not (a and b) = (not a) or (not b)
  - not (a or b) = (not a) and (not b)

Algebraic properties

- Absorption
  - (a and b) or a = a
  - (a or b) and a = a
- Idempotency
  - a or a = a
  - a and a = a
- Exclusion (not satisfied)
  - a or ¬a ≠ 1
  - a and ¬a ≠ ∅

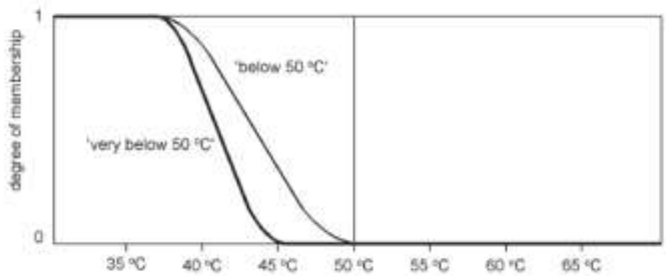
Linguistic modifiers

- **Approximation of Fuzzy Sets**: scalar → fuzzy set, modifying the "base" of a fuzzy set
  - about, around, near and close to
- **Restriction of Fuzzy Sets**: modifying the shape
  - below and above
- **Intensification and Dilution of Fuzzy Sets**
  - *intensification*: very (n = 2) and extremely (n = 3)
  - *dilution*: somewhat (n = 1/2) and greatly (n = 5/7)

$$\mathrm{op}\ \mu(x) = \mu^n(x)$$

Linguistic modifiers

- Graphical representation



Inference in fuzzy systems

Artificial intelligence
Kristóf Karacs
PPKE-ITK

Fuzzy relation

- Given two universes $\mathcal{X}$ and $\mathcal{Y}$, a fuzzy relation $\mathcal{R}$ is

$$\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$$

where $\subset$ denotes a fuzzy subset

- $\mathcal{R}$ is defined by $\mu_{\mathcal{R}}(x, y)$

Composition

- Given two fuzzy relations

$$\mathcal{R}\colon \mathcal{X} \times \mathcal{Y} \to [0,1] \qquad \mathcal{S}\colon \mathcal{Y} \times \mathcal{Z} \to [0,1]$$

their composition is defined by

$$\mathcal{T} = \mathcal{R} \circ \mathcal{S}\colon \mathcal{X} \times \mathcal{Z} \to [0,1]$$

$$\mu_{\mathcal{R}\circ\mathcal{S}}(x,z) = \max_{y\in\mathcal{Y}}\{\min\{\mu_{\mathcal{R}}(x,y), \mu_{\mathcal{S}}(y,z)\}\}$$

- Called an inner or - and product

Composition example

$$R(X,Y) = \begin{bmatrix} 0.8 & 1 & 0.1 & 0.7 \\ 0 & 0.8 & 0 & 0 \end{bmatrix} \qquad S(Y,Z) = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0 & 0.4 & 0 \\ 0.3 & 0.5 & 0.8 \\ 0.6 & 0.7 & 0.5 \end{bmatrix}$$

$$(x_i, z_j) = \max_{y_k}\left(\min\left((x_i, y_k), (y_k, z_j)\right)\right)$$

$$R \circ S(X,Z) = \begin{bmatrix} 0.6 & 0.7 & 0.5 \\ 0 & 0.4 & 0 \end{bmatrix}$$

Form of reasoning

- Fuzzy version of generalized modus ponens
- Antecedent (premise): *x* is A'
  - Implication: if *x* is A then *y* is B
  - Consequence: *y* is B'

$$\frac{A', \quad A \longrightarrow B}{B'} \qquad A' \circ R_{A \to B} = B'$$

Implication

- Let A and B be two fuzzy sets in U_1 and U_2, respectively
- Implication is a relation defined by

$$A \rightarrow B \triangleq A \otimes B,$$

  - where $\otimes$ is the tensor (outer) product of the vectors using the logical operator and ($\wedge$)

- Implication functions
  - I(x,y) = min(x, y) Mamdani
  - I(x,y) = max(1-x, y) Dilne, Zadeh
  - I(x,y) = xy Larsen

Implication example

- Rule
  - "If temperature is high, then humidity is fairly high."
- Fuzzy variables
  - $t \in U_t = \{20,30,40\}$ $h \in U_h = \{20,50,70,90\}$
- Fuzzy sets
  - $HT \subseteq U_t$ $\mu_{HT}(t) = [0.1, 0.5, 0.9]^T$
  - $FHH \subseteq U_h$ $\mu_{FHH}(h) = [0.2, 0.6, 0.7\ 1]^T$
- Fuzzy rule
  - $R(t,h)$: if t is HT then h is FHH

Implication example

- $\mu_{HT}(t) = [0.1, 0.5, 0.9]^T$
- $\mu_{FHH}(h) = [0.2, 0.6, 0.7\ 1]^T$
- $R_{HT \to FHH} = \mathrm{HT} \otimes \mathrm{FHH}$
- $R_{HT \to FHH} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.5 & 0.5 & 0.5 \\ 0.2 & 0.6 & 0.7 & 0.9 \end{bmatrix}$

Implication example

- According to the rule what is the humidity if temperature is fairly high?
  - t = FHT, FHT $\subseteq U_t$

- $\mu_{FHT}(t) = \mu_{HT}^2(t) = [0.01, 0.25, 0.81]^T$

Implication example

- $R(h) = R(t) \circ R_{HT \to FHH} = FHT \circ R_{HT \to FHH}$

$$= [0.01 \quad 0.25 \quad 0.81] \circ \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.5 & 0.5 & 0.5 \\ 0.2 & 0.6 & 0.7 & 0.9 \end{bmatrix}$$

$$= [0.2 \quad 0.6 \quad 0.7 \quad 0.81]$$

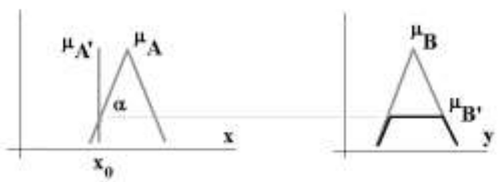
Single rule

General fuzzification

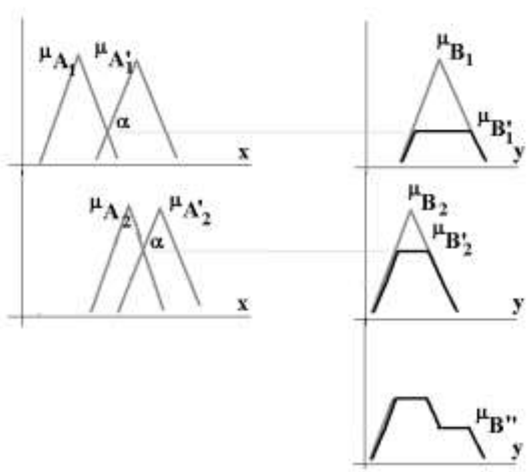
$A \to B$
A'

B'

Singleton fuzzification



Superposition of rules



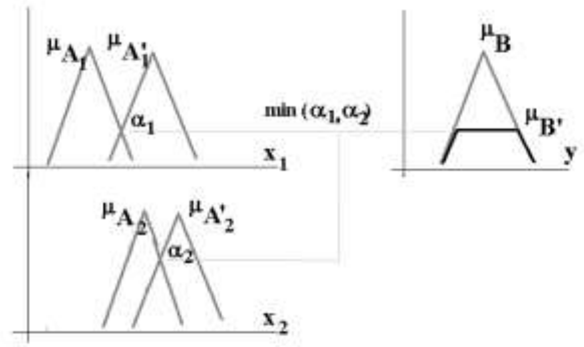
$A_1 \to B_1$
A'_1

$A_2 \to B_2$
A'_2

B'_1, B'_2

B''

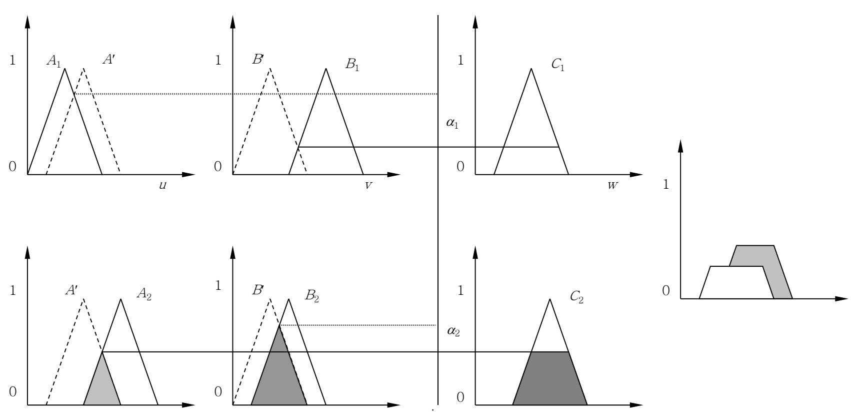
Multiple antecedents



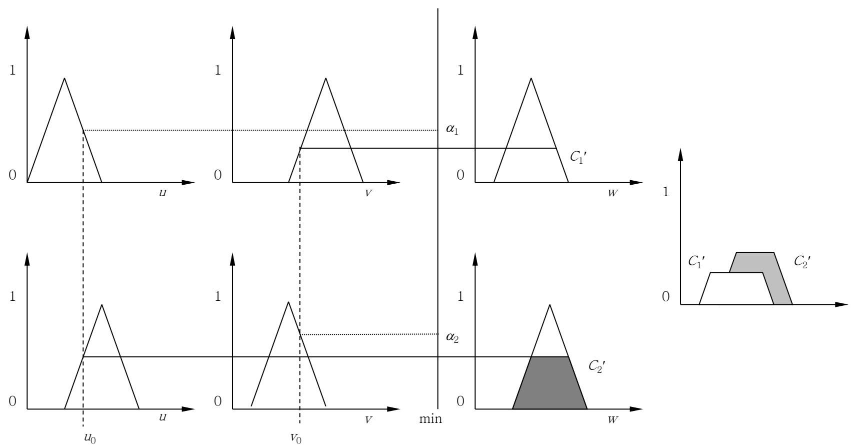
$A_1 \wedge A_2 \rightarrow B$
A'_1, A'_2

B'

Multiple Rules – Fuzzy-Fuzzy



Multiple Rules – Crisp-Fuzzy



Defuzzification

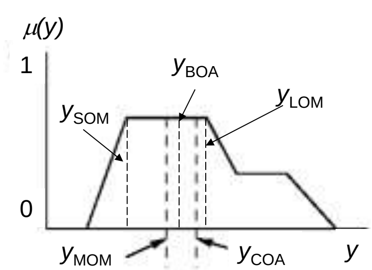
- Converting fuzzy set to crisp data
  - Mean of maxima (MOM)
    - y_{mj}: set of points with maximum membership value

$$y_{MOM} = \frac{\sum_{j=1}^{l} y_{mj}}{l}$$

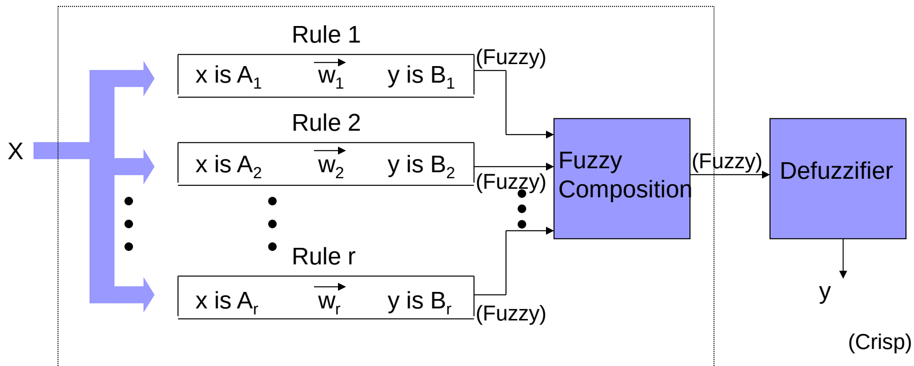
  - Center of area (COA)

$$y_{COA} = \frac{\sum_{j=1}^{l} \mu(y_{mj}) y_{mj}}{\sum_{j=1}^{l} \mu(y_{mj})}$$

  - Bisector of Area (BOA)
  - Smallest of Maximum (SOM)
  - Largest of Maximum (LOM)



Model of a fuzzy system



Ingredients of a fuzzy system

- Normalization of universes
- Fuzzification of crisp input data
- Fuzzy inference
- Defuzzification
- Denormalization