

## **CS 1571 Introduction to AI Review**

### **Course review**

**Milos Hauskrecht**

[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)

5329 Sennott Square

---

CS 1571 Introduction to AI

## **Announcements**

### **Final exam**

**Date: December 10, 2007 at 10:00-11:50am**

**Location: 5129 Sennott Square**

- Closed book
- Cumulative

### **Exam:**

- The structure of the exam similar to the midterm
- Theoretical problems (no programming)

---

CS 1571 Introduction to AI

# Review

# Search

- **Basic definition of the search problem**
  - Search space, operators, initial state, goal condition
- **Formulation of a problem:**
  - We have some control over the complexity of the **search space size**
- **Two types:**
  - Path vs. configuration search
- **Expected skills:**
  - Take a problem and formulate it as a search problem
  - **Define:** search space, states, operators, and the goal

## Search methods

- **Different methods for searching the search space exist**
- **Search trace is captured by the search tree**
- **Search methods properties :**
  - Completeness, Optimality, Space and time complexity.
- **Complexities**
  - measured in terms of a branching factor ( $b$ ), depth of the optimal solution ( $d$ ), maximum depth of the state space ( $m$ )
- **Expected Skills:**
  - Analyze a search algorithm in terms of its properties
  - Propose an algorithm that fits the problem you want to solve the best

## Search methods

- **Uninformed methods:**
    - Breadth first search, Depth first search, Iterative deepening, Bi-directional search, Uniform cost search (for the weighted path search)
  - **Informed methods:**
    - **Heuristic function ( $h$ ):** potential of a state to reach the goal
    - **Evaluation function ( $f$ ) :** desirability of a state to be expanded next
    - **Best first search:**
      - Greedy  $f(n) = h(n)$
      - A\*:  $f(n) = g(n) + h(n)$
- the role of admissible heuristics, optimality

## Constraint satisfaction search

- **Constraint satisfaction problem (CSP)**
  - Variables, constraints on values (reflect the goal)
- **Methods:**
  - Backtracking (DFS)
  - Constraint propagation
- **Heuristics for CSP search expansion:**
  - most constrained variable
  - least constrained value
- **Expected skills:**
  - Formulation of a problem as a CSP search
  - Constraint propagation

## Optimization

### Complex configuration searches

- **rely on iterative algorithms**
  - **Methods:**
    - Hill climbing,
    - Simulated annealing
    - Genetic algorithms
- **Advantage of iterative algorithms: ?**

# Optimization

## Complex configuration searches

- **rely on iterative algorithms:**
  - **Methods:**
    - Hill climbing,
    - Simulated annealing
    - Genetic algorithms
- **Advantage of iterative algorithms: ?**
  - **memory !!**
  - **useful for very large optimization problems.**

# Adversarial search

- **Adversarial search (game playing)**
  - Specifics of a game search, game problem formulation
  - rational opponent
- **Algorithms:**
  - **Minimax algorithm**
    - Complexity bottleneck for large games
  - **Alpha-Beta pruning:** prunes branches not affecting the decision of players
  - **Cutoff** of the search tree and heuristics
- **Expected skills:**
  - Minimax and alpha beta pruning
  - Design of a heuristic evaluation function

## KR and logic

- **Knowledge representation:**
  - **Syntax** (how sentences are build), **Semantics** (meaning of sentences), **Computational aspect** (how sentences are manipulated)
- **Logic:**
  - A formal language for expressing knowledge and ways of reasoning
  - **Three components:**
    - A set of sentences
    - A set of interpretations
    - The valuation (meaning) function

## Propositional logic

- A language for symbolic reasoning
- **Language:**
  - Syntax, Semantics
- **Satisfiability** of a sentence: at least one interpretation under which the sentence can evaluate to **True**.
- **Entailment:**  
 $KB \models \alpha$  is true in all worlds in which KB is true
- **Inference procedure**
  - Soundness      If  $KB \vdash_i \alpha$  then  $KB \models \alpha$
  - Completeness      If  $KB \models \alpha$  then  $KB \vdash_i \alpha$

## Propositional logic

- **Logical inference problem:**  $KB \models \alpha$  ?
    - Does KB entail the sentence  $\alpha$  ?
  - Logical inference problem for the propositional logic is **decidable**.
    - A procedure (program) that stops in finite time exists
  - **Approaches:**
    - Truth table approach
    - Inference rule approach
    - Resolution refutation
- $KB \models \alpha$  if and only if  
 $(KB \wedge \neg \alpha)$  is **unsatisfiable**
- **Normal forms:** DNF, CNF, Horn NF (conversions)

---

CS 1571 Introduction to AI

## First order logic

- Deficiencies of propositional logic
- **First order logic (FOL):**
  - allows us to represent objects, their properties, relations and statements about them
    - Variables, predicates, functions, quantifiers
    - Syntax and semantics of the sentences in FOL
- **Expected skills:**
  - Translation of English sentences to FOL

---

CS 1571 Introduction to AI

## First order logic

- **Logical inference problem**  $KB \models \alpha$  ?
  - **Undecidable:**
  - No procedure that can decide the entailment for all possible input sentences in a finite number of steps.
- **Inference approaches:**
  - Inference rules
  - Resolution refutation

## First order logic

- **Methods for making inferences work with variables:**
  - **Variable substitutions**
  - **Unification** process that takes two similar sentences and computes the substitution that makes that makes them look the same, if it exists
- **Conversions to CNF** with universally quantified variables
  - Used by resolution refutation
    - The procedure is refutation-complete
- **Expected skills:**
  - Take sentences in FOL
  - Convert them to CNF
  - Use resolution-refutation method to prove the theorems

## Knowledge-based systems with HNF

- **KBs in Horn normal form:**
  - Not all sentences in FOL can be translated to HNF
  - Modus ponens is complete for Horn databases
- **Inferences with KBs in Horn normal form (HNF)**
  - Forward chaining
  - Backward chaining
- **Expected skills:**
  - Inferences (forward and backward) with rules and facts

## Knowledge-based systems based on FOL

- **Production systems**
  - What is the difference from the KB in Horn Normal Form?
  - Conclusions of the rules are actions:

## **Knowledge-based systems based on FOL**

- **Production systems**
  - What is the difference from the KB in Horn Normal Form?
  - Conclusions of the rules are actions:
    - ADD a predicate
    - DELETE a predicate
    - MODIFY a predicate
    - PRINT
    - ASK

## **Knowledge-based systems based on FOL**

- **First-order logic is monotonous**
  - What does it mean?

## Knowledge-based systems based on FOL

- **First-order logic is monotonous**
  - **What does it mean?**
  - **Are production systems monotonous?**
  - **Why or why not?**

## Knowledge-based systems based on FOL

- **First-order logic is monotonous**
  - **What does it mean?**
  - **Are production systems monotonous?**
  - **Why or why not?**
  
  - **What is conflict resolution?**

## Planning

- **Find a sequence of actions** that lead to a goal
  - Much like path search, but for very large domains
  - Need to represent the dynamics of the world
- **Two basic approaches** planning problem representation:
  - **Situation calculus**
    - Explicitly represents situations (extends FOL)
    - **Solving:** theorem proving
  - **STRIPS**
    - Focuses on changes only
    - **Solving:** Search  
(Goal progression, Goal regression)
- **Frame problem**

## STRIPS planning

### Operator schema

- **Move (x,y,z)** ..... moves object x from y to z
- **Definition: ?**
  - **3 components**

## STRIPS planning

### Operator schema

- **Move (x,y,z)** ..... moves object x from y to z
- **Definition:**
  - **Condition:** a conjunction of literals
  - **Add list:** new predicates that hold in the new state
  - **Remove list:** predicates that were true in the previous state but do not hold in the new state

## STRIPS planning

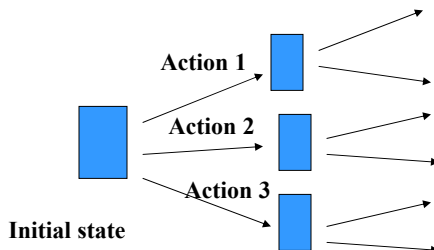
- **Linear planners: ?**

## STRIPS planning

- **Linear planners:**
  - Build sequences of actions one-by-one in the order of their execution either from the beginning or from the end
- **What search method builds the solution from the beginning?**
- 

## STRIPS planning

- **Linear planners:**
  - Build sequences of actions in the order of their execution either from the beginning or from the end
- **What search method builds the solution from the beginning?**
- Forward state space search



## **STRIPS planning**

- **How does the backward search work?**

## **STRIPS planning**

- **How does the backward search work?**
- **Starts from the goal condition and projects the goals backward using operator**

## **STRIPS planning**

- **What is the divide and conquer approach to solving the problem?**

## **STRIPS planning**

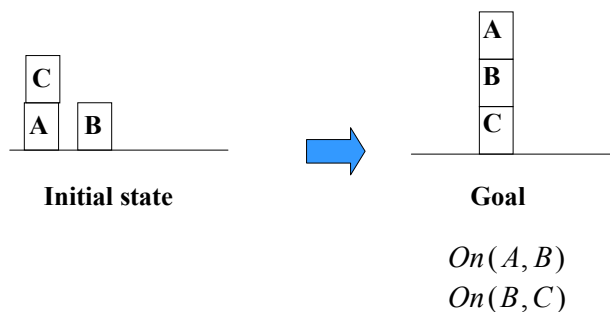
- **What is the divide and conquer approach to solving the problem?**
- Decompose the problem to a set of simpler subproblems, solve the subproblems and merge their solutions.
- **Do the linear planners support the decomposition of the planning problem?**

## STRIPS planning

- **What is the divide and conquer approach to solving the problem?**
- Decompose the problem to a set of simpler subproblems, solve the subproblems and merge their solutions.
- **Do the linear planners support the decomposition of the planning problem?**
- No. The full sequence is always built.

## Sussman's anomaly.

- Is the solution of the planning problem decomposable along goals?



## Partial-order planning

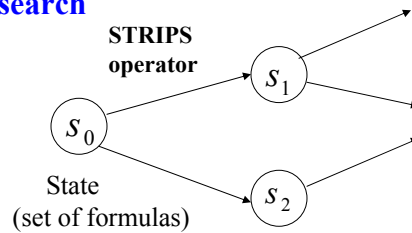
- **Partial order (non-linear) planners:**
  - How they differ from linear planners?

## Partial-order planning

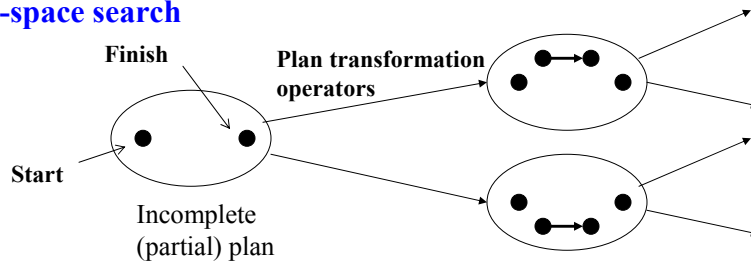
- **Partial order (non-linear) planners:**
  - How they differ from linear planners?
  - They search a space of partial plans

## State-space vs. plan-space search

### State-space search



### Plan-space search



CS 1571 Introduction to AI

## Uncertainty

- **Basics of probability:**
  - A random variable, values, probability distribution
- **Joint probability distribution**
  - Over variables in a set, **full joint** over all variables
  - Marginalization (summing out)

- **Conditional probability distribution**

$$P(A|B) = \frac{P(A,B)}{P(B)} \text{ s.t. } P(B) \neq 0$$

- **Product rule**  $P(A,B) = P(A|B)P(B)$
- **Chain rule**

CS 1571 Introduction to AI

## Uncertainty

- **Bayes rule**

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- **Used often for diagnostic inferences:**

- **From effect to causes**

E.g.  $P(\text{device}=\text{normal} \mid \text{sensor reading}=\text{high})$

- **Probabilities given are typically opposite**

- **From causes to effects**

## Uncertainty

### **Full joint probability distribution**

- the distribution over all variables defining the problem

### **Two important things to remember:**

- Any probabilistic query can be computed from the full joint distribution
- Full joint distribution can be expressed as a product of conditionals via the chain rule

## Bayesian belief networks

### Full joint distribution

- over all random variables defining the domain can be very large

### Issues:

- Complexity of a model,
- Complexity of inferences,
- Acquisition cost

### Solution: Bayesian belief networks (BBNs)

- **BBN** build upon conditional independence relations:

$$P(A, B | C) = P(A | C)P(B | C)$$

## Bayesian belief networks

- **Two components of BBNs:**
  - Structure (directed acyclic graph)
  - Parameters (conditional prob. distributions)
- **Full joint probability distribution for the BBN:**
  - Product of local (variable-parents) conditionals

$$\mathbf{P}(X_1, X_2, \dots, X_n) = \prod_{i=1, \dots, n} \mathbf{P}(X_i | pa(X_i))$$

## Bayesian belief networks

- **Benefits for the representation of the full joint distribution:**
- ?

## Bayesian belief networks

- **Benefits for the representation of the full joint distribution:**
- We need a smaller number of parameters to define the full joint

## Parameter complexity problem

Alarm example:

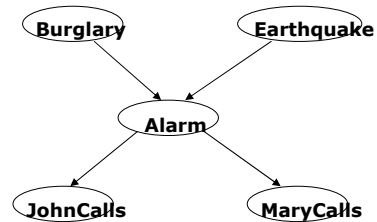
5 binary (True, False) variables

# of parameters of the full joint:

$$2^5 = 32$$

One parameter is for free:

$$2^5 - 1 = 31$$



## Parameter complexity problem

Alarm example:

5 binary (True, False) variables

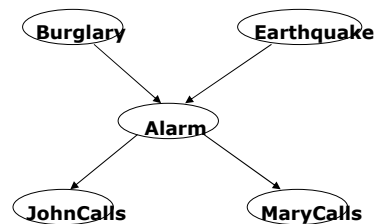
# of parameters of the full joint:

$$2^5 = 32$$

One parameter is for free:

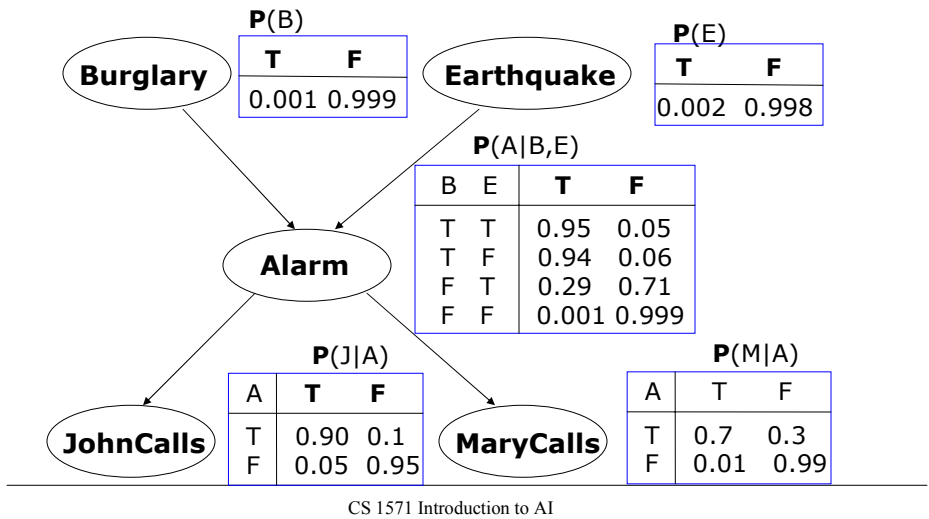
$$2^5 - 1 = 31$$

# of parameters of the BBN: ?



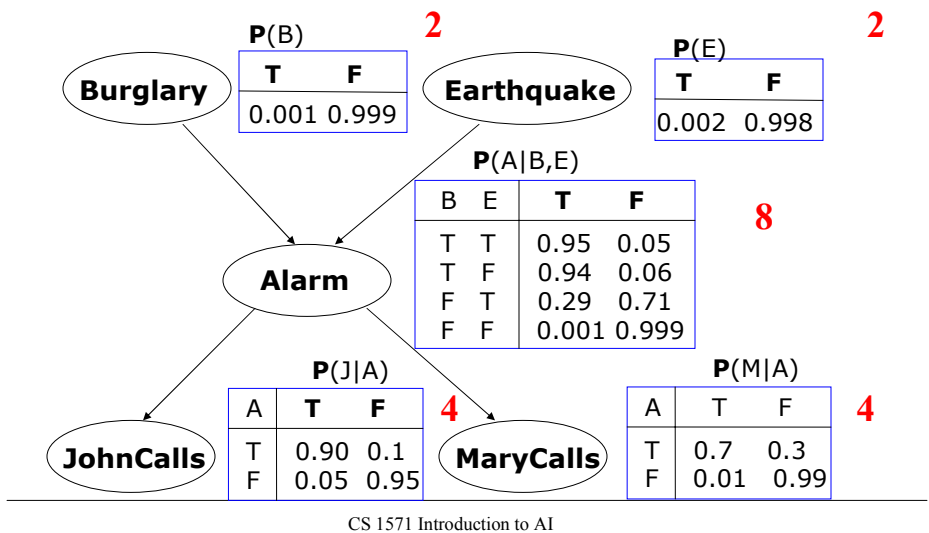
## Bayesian belief network.

- In the BBN the **full joint distribution** is expressed using a set of local conditional distributions



## Bayesian belief network.

- In the BBN the **full joint distribution** is expressed using a set of local conditional distributions



## Parameter complexity problem

Alarm example:

5 binary (True, False) variables

# of parameters of the full joint:

$$2^5 = 32$$

One parameter is for free:

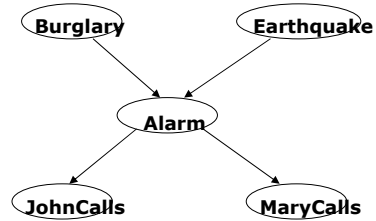
$$2^5 - 1 = 31$$

# of parameters of the BBN:

$$2^3 + 2(2^2) + 2(2) = 20$$

One parameter in every conditional is for free:

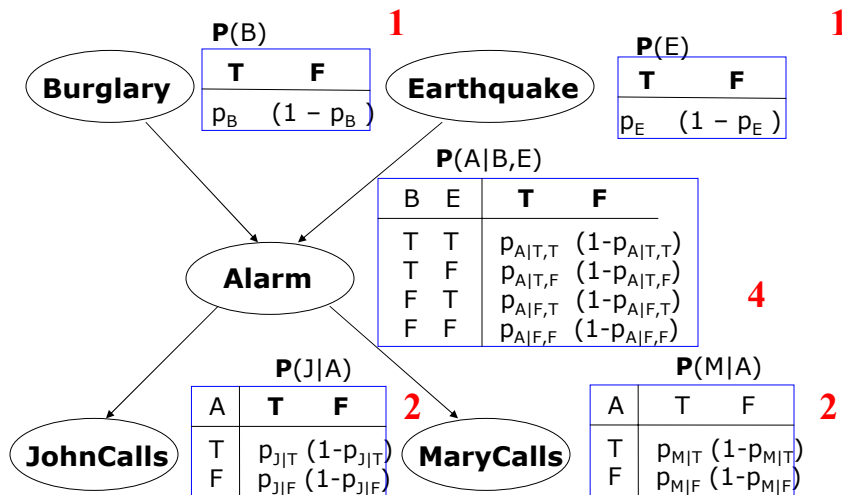
?



CS 1571 Introduction to AI

## Bayesian belief network.

- Number of free parameters



CS 1571 Introduction to AI

## Parameter complexity problem

- In the BBN the **full joint distribution** is defined as:

$$\mathbf{P}(X_1, X_2, \dots, X_n) = \prod_{i=1, \dots, n} \mathbf{P}(X_i \mid pa(X_i))$$

- What did we save?

Alarm example: 5 binary (True, False) variables

# of parameters of the full joint:

$$2^5 = 32$$

One parameter is for free:

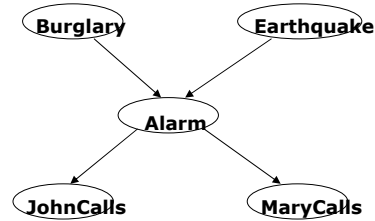
$$2^5 - 1 = 31$$

# of parameters of the BBN:

$$2^3 + 2(2^2) + 2(2) = 20$$

One parameter in every conditional is for free:

$$2^2 + 2(2) + 2(1) = 10$$



## Bayesian belief networks

Advantage of BBNs for inferences:

- Smart way to do inferences:
  - Interleave sums and products
  - Based on:

$$\sum_x a f(x) = a \sum_x f(x)$$

## Inference in Bayesian networks

**Computing:**  $P(J = T)$

**Approach 1. Blind approach.**

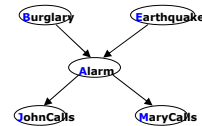
- Sum out all un-instantiated variables from the full joint,
- express the joint distribution as a product of conditionals

$$\begin{aligned}
 P(J = T) &= \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(B = b, E = e, A = a, J = T, M = m) \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(A = a | B = b, E = e) P(B = b) P(E = e)
 \end{aligned}$$

**Computational cost:**

Number of additions: ?

Number of products: ?



## Inference in Bayesian networks

**Computing:**  $P(J = T)$

**Approach 1. Blind approach.**

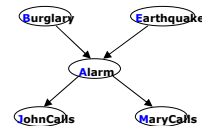
- Sum out all un-instantiated variables from the full joint,
- express the joint distribution as a product of conditionals

$$\begin{aligned}
 P(J = T) &= \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(B = b, E = e, A = a, J = T, M = m) \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(A = a | B = b, E = e) P(B = b) P(E = e)
 \end{aligned}$$

**Computational cost:**

Number of additions: 15

Number of products: ?



## Inference in Bayesian networks

**Computing:**  $P(J = T)$

### Approach 1. Blind approach.

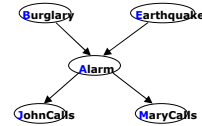
- Sum out all un-instantiated variables from the full joint,
- express the joint distribution as a product of conditionals

$$\begin{aligned}
 P(J = T) &= \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(B = b, E = e, A = a, J = T, M = m) \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(A = a | B = b, E = e) P(B = b) P(E = e)
 \end{aligned}$$

**Computational cost:**

Number of additions: 15

Number of products:  $16 * 4 = 64$



## Inference in Bayesian networks

### Approach 2. Interleave sums and products

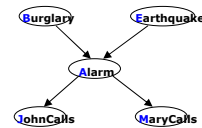
- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$\begin{aligned}
 P(J = T) &= \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(A = a | B = b, E = e) P(B = b) P(E = e) \\
 &= \sum_{b \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(B = b) \left[ \sum_{e \in T, F} P(A = a | B = b, E = e) P(E = e) \right] \\
 &= \sum_{a \in T, F} P(J = T | A = a) \left[ \sum_{m \in T, F} P(M = m | A = a) \right] \left[ \sum_{b \in T, F} P(B = b) \right] \left[ \sum_{e \in T, F} P(A = a | B = b, E = e) P(E = e) \right]
 \end{aligned}$$

**Computational cost:**

Number of additions:  $1 + 2 * [1 + 1 + 2 * 1] = ?$

Number of products:  $2 * [2 + 2 * (1 + 2 * 1)] = ?$



## Inference in Bayesian networks

### Approach 2. Interleave sums and products

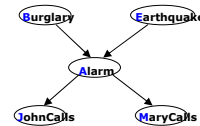
- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$\begin{aligned}
 P(J = T) &= \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(A = a | B = b, E = e) P(B = b) P(E = e) \\
 &= \sum_{b \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(B = b) \left[ \sum_{e \in T, F} P(A = a | B = b, E = e) P(E = e) \right] \\
 &= \sum_{a \in T, F} P(J = T | A = a) \left[ \sum_{m \in T, F} P(M = m | A = a) \right] \left[ \sum_{b \in T, F} P(B = b) \right] \left[ \sum_{e \in T, F} P(A = a | B = b, E = e) P(E = e) \right]
 \end{aligned}$$

#### Computational cost:

Number of additions:  $1 + 2 * [1 + 1 + 2 * 1] = 9$

Number of products:  $2 * [2 + 2 * (1 + 2 * 1)] = ?$



## Inference in Bayesian networks

### Approach 2. Interleave sums and products

- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$\begin{aligned}
 P(J = T) &= \\
 &= \sum_{b \in T, F} \sum_{e \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(A = a | B = b, E = e) P(B = b) P(E = e) \\
 &= \sum_{b \in T, F} \sum_{a \in T, F} \sum_{m \in T, F} P(J = T | A = a) P(M = m | A = a) P(B = b) \left[ \sum_{e \in T, F} P(A = a | B = b, E = e) P(E = e) \right] \\
 &= \sum_{a \in T, F} P(J = T | A = a) \left[ \sum_{m \in T, F} P(M = m | A = a) \right] \left[ \sum_{b \in T, F} P(B = b) \right] \left[ \sum_{e \in T, F} P(A = a | B = b, E = e) P(E = e) \right]
 \end{aligned}$$

#### Computational cost:

Number of additions:  $1 + 2 * [1 + 1 + 2 * 1] = 9$

Number of products:  $2 * [2 + 2 * (1 + 2 * 1)] = 16$

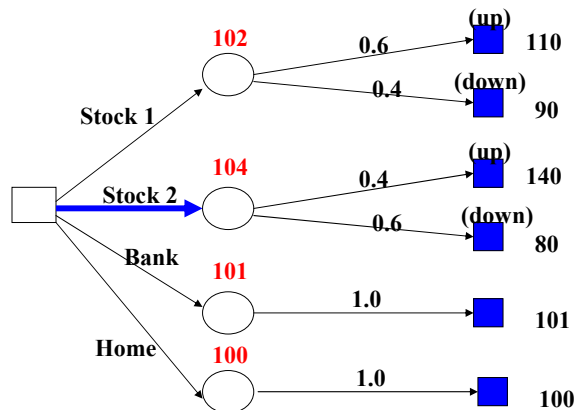
## Decision-making in the presence of uncertainty

- **Decision tree:**
  - Decision nodes (choices are made)
  - Chance nodes (reflect stochastic outcome)
  - Outcomes (value) nodes (value of the end-situation)
- **Rational choice (for a risk neutral decision maker):**
  - Decision-maker tries to optimize the expected value
- **Utility theory:**
  - Utilities express preferences in terms of numeric quantities
  - Monetary values may not be equal to utility values
  - Choices based on expected utility (and its maximization)
- **Value of information**
  - Additional information (via test) can reduce the expected utility

CS 1571 Introduction to AI

## Decision making

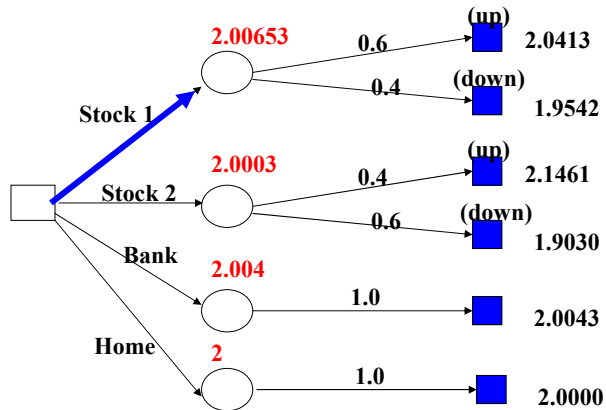
- **Monetary outcomes**
- **Goal: optimize the expected value of the investment**



CS 1571 Introduction to AI

## Decision making with the utility function

- Preferences of a decision maker captured by a utility function
- Utility function  $\log(x)$



CS 1571 Introduction to AI

## Learning

- **Supervised learning**
  - Learning mapping between inputs  $x$  and desired outputs  $y$
  - Teacher gives me  $y$ 's for the learning purposes
- **Unsupervised learning**
  - Learning relations between data components
  - No specific outputs given by a teacher
- **Reinforcement learning**
  - Learning mapping between inputs  $x$  and desired outputs  $y$
  - Critic does not give me  $y$ 's but instead a signal (reinforcement) of how good my answer was
- **Other types of learning:**
  - **Concept learning, explanation-based learning, etc.**

CS 1571 Introduction to AI