# Final Review

Shirley Chen. Sajad Darabi

# Content

- We first go over the topics that we didn't cover in past discussions (underlined)
- If there is time, we review the problems we practiced before

# Bayesian Network

- Objective:
  - model conditional dependency => causation
  - probability computation

- A directed acyclic graph (DAG)
- Each edge: a conditional dependency
- Each node: a unique random variable

- Edge (A,B): P(B|A) is a **factor** in the joint probability distribution
  - We must know P(B|A) for all values of B and A in order to conduct inference

# Bayesian Network

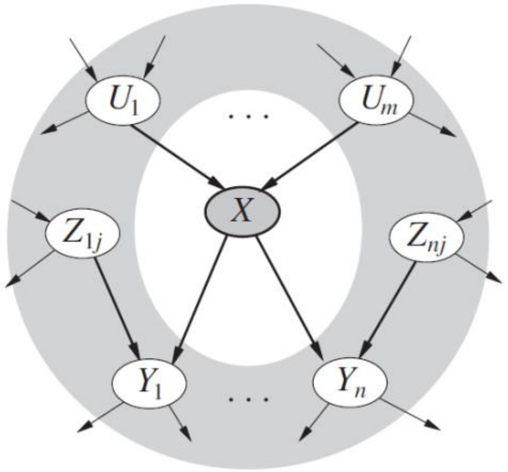
- Edge (A,B): P(B|A) is a **factor** in the joint probability distribution
  - Chain rule of probability
  - P(A0, A1, ... An) = P(A1|A2, ..., An) * P(A2|A3, ... An) * ... *P(An)

- Conditional independency $A \perp \mathrm{B}|\mathrm{C}$
  - P(A,B|C) = P(A|C)*P(B|C)
  - or P(A|B,C) = P(A|C)
  - A and B are independent when the value of C is known and fixed

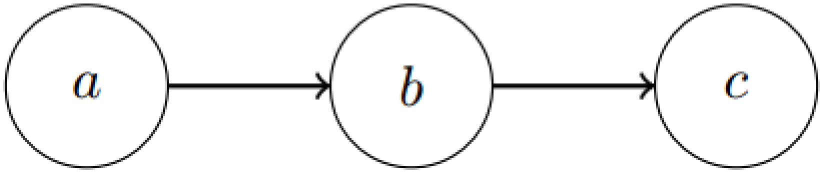
# Bayesian Network

- BN satisfies **local Markov property:**
  - A node is conditionally independent of its non-descendants given its parents. (topological semantics)

- Markov Blanket
  - The node's parents, children and children's parents
  - The node is conditionally independent of all other nodes given this Markov Blanket

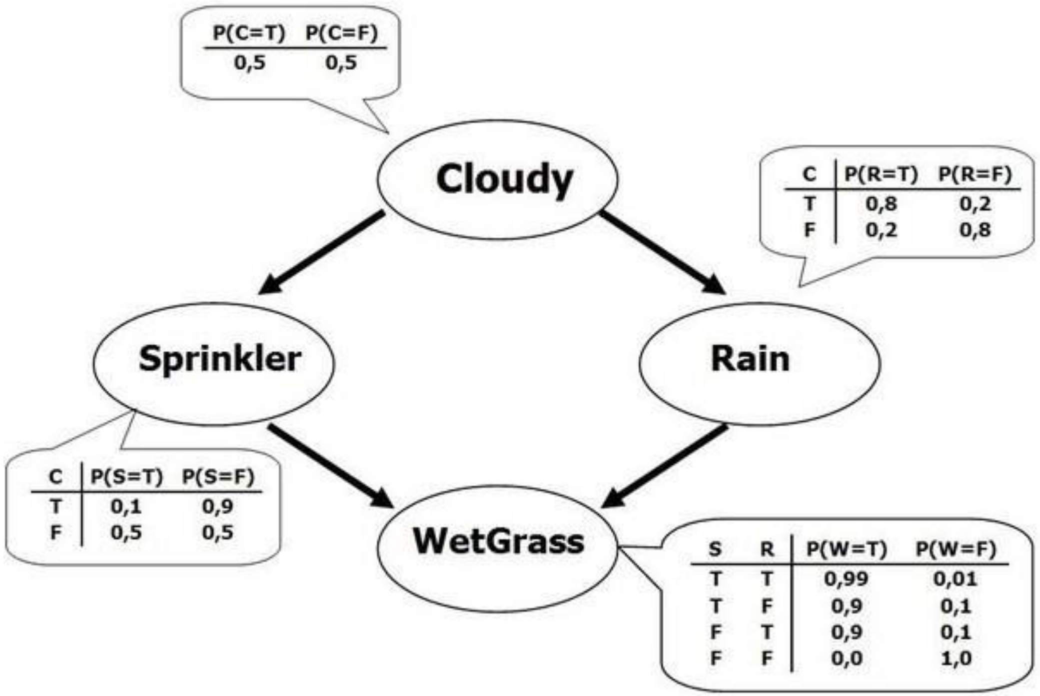


# Exercise – conditional independency



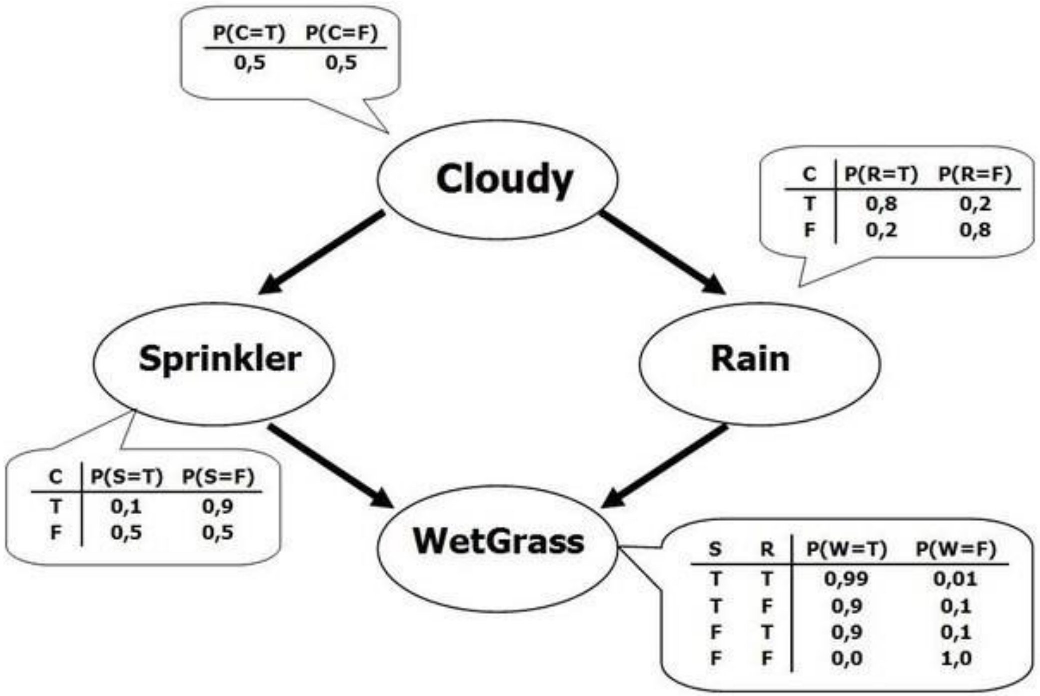

Give the topological semantics encoded in the BN.

# Exercise – conditional independency




- Given Cloudy, what variables is Sprinkler conditional independent of?

# Exercise – conditional independency




- Given Cloudy, what variables is Sprinkler conditional independent of?

Rain

# Bayesian Network

- BN satisfies **local Markov property**
  - A node is conditionally independent of its non-descendants given its parents.

  - The joint probability computation is simplified!
  - $\mathrm{P}(\mathrm{A}_1, \dots, \mathrm{A}_\mathrm{n}) = \prod_{i=1}^{n} P(A_i|\mathrm{Parents}(A_i))$

# Bayesian Network

**Inference over Bayesian network**

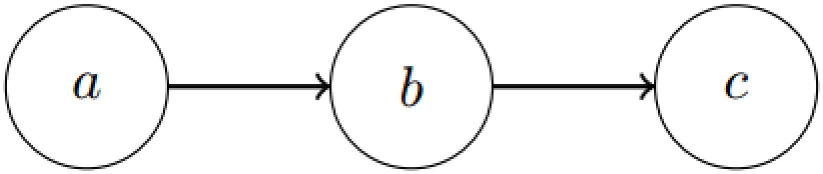
- Compute joint probability of a particular assignment
  - $\mathrm{P}(\mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{n}) = \prod_{i=1}^{n} P(A_i|\mathrm{Parents}(A_i))$
  - Greatly reduce the amount of required computation
- Compute P(x|e)

# Bayesian Network

Two main methods for inference

- By enumeration
  - Compute sums of products of conditional probabilities
  - (A lot repeated calculations)
- By variable elimination (using factors)
  - Store intermediate results to avoid repeated calculations
  - summing out variables (right to left) from pointwise products of factors to produce new factors

# Exercise – Inference by enumeration




| $a$ | $\Pr(a)$ |
|---|---|
| 1 | 1/2 |
| 0 | 1/2 |

| $a$ | $b$ | $\Pr(b \mid a)$ |
|---|---|---|
| 1 | 1 | 1/8 |
| 1 | 0 | 7/8 |
| 0 | 1 | 1/4 |
| 0 | 0 | 3/4 |

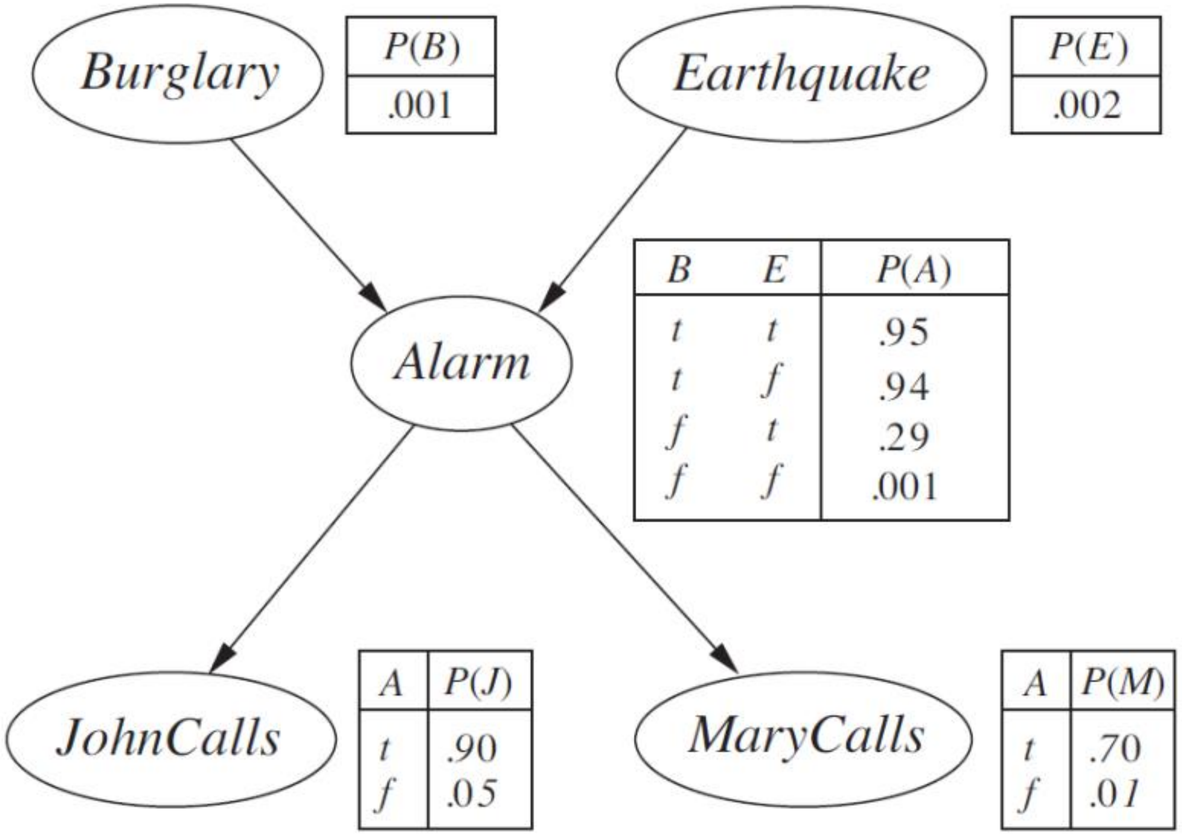
| $b$ | $c$ | $\Pr(c \mid b)$ |
|---|---|---|
| 1 | 1 | 4/5 |
| 1 | 0 | 1/5 |
| 0 | 1 | 1/4 |
| 0 | 0 | 3/4 |

compute Pr(a=T|b=T)

# Exercise – Inference over BN

$$
\begin{aligned}
\Pr(a = \texttt{true} \mid b = \texttt{true}) &= \frac{Pr(a = \texttt{true}, b = \texttt{true})}{Pr(b = \texttt{true})} \\
&= \frac{\frac{1}{2} \cdot \frac{1}{8}}{\frac{1}{2} \cdot \frac{1}{8} + \frac{1}{2} \cdot \frac{1}{4}} \\
&= \frac{\frac{1}{16}}{\frac{1}{16} + \frac{1}{8}} \\
&= \frac{1}{3}
\end{aligned}
$$

# Example - Inference by Enumeration




**Compute** $\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true)$.

# Example - Inference by Enumeration

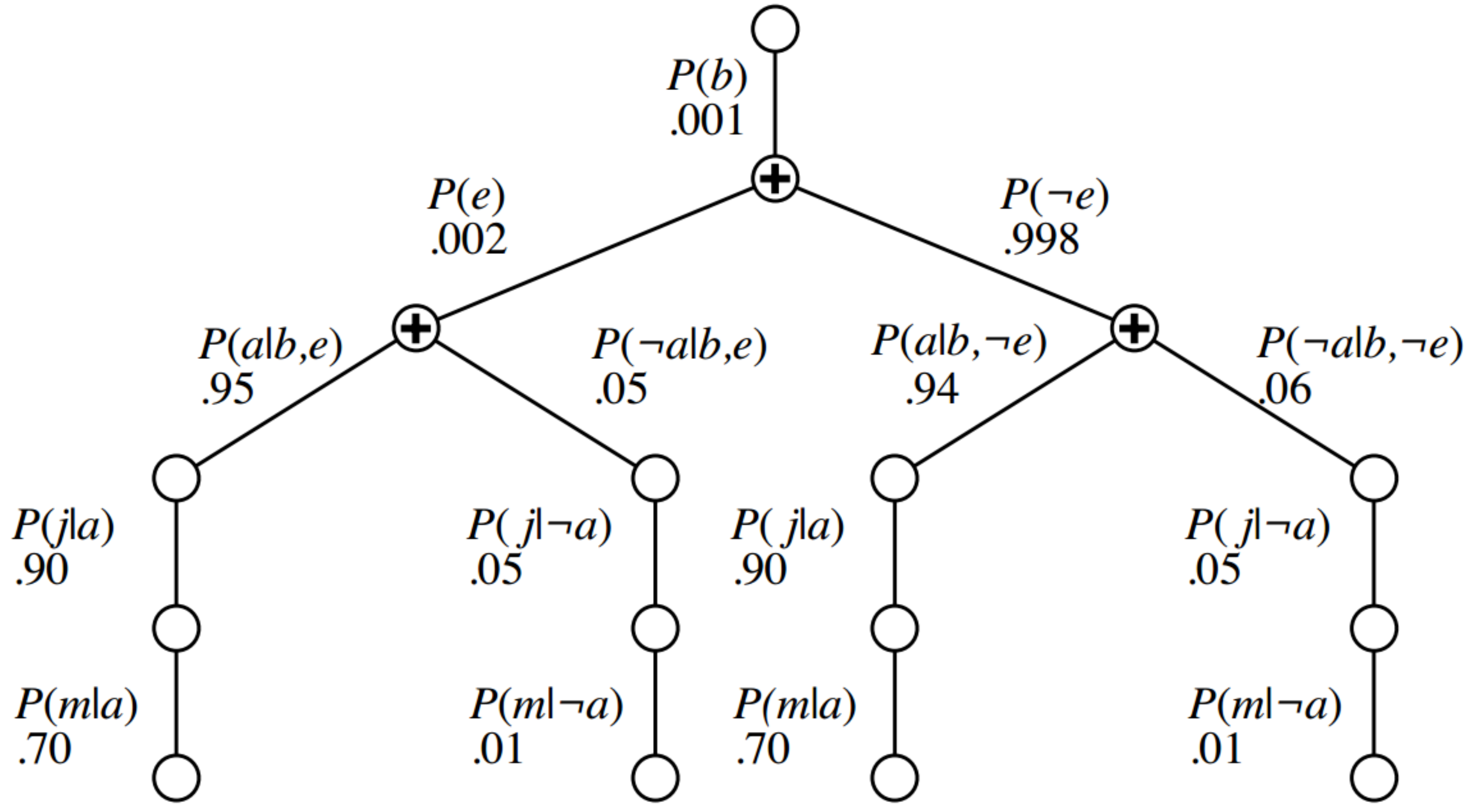



**Figure 14.8** The structure of the expression shown in Equation (14.4). The evaluation proceeds top down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for $j$ and $m$.

# Inference by Variable Elimination

- Write probabilities as factor multiplication

$$\mathbf{P}(B \mid j, m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathbf{f}_1(B)} \sum_e \underbrace{P(e)}_{\mathbf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a \mid B, e)}_{\mathbf{f}_3(A,B,E)} \underbrace{P(j \mid a)}_{\mathbf{f}_4(A)} \underbrace{P(m \mid a)}_{\mathbf{f}_5(A)} .$$

$$\mathbf{P}(B \mid j, m) = \alpha \, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

X is pointwise multiplication, not ordinary matrix multiplication

# Inference by Variable Elimination

- Factor multiplication

The pointwise product of two factors **f1** and **f2** yields a new factor **f** whose variables are the *union* of the variables in **f1** and **f2** and whose elements are given by the product of the corresponding elements in the two factors.

| $A$ | $B$ | $\mathbf{f}_1(A,B)$ | $B$ | $C$ | $\mathbf{f}_2(B,C)$ | $A$ | $B$ | $C$ | $\mathbf{f}_3(A,B,C)$ |
|---|---|---|---|---|---|---|---|---|---|
| T | T | .3 | T | T | .2 | T | T | T | $.3 \times .2 = .06$ |
| T | F | .7 | T | F | .8 | T | T | F | $.3 \times .8 = .24$ |
| F | T | .9 | F | T | .6 | T | F | T | $.7 \times .6 = .42$ |
| F | F | .1 | F | F | .4 | T | F | F | $.7 \times .4 = .28$ |
| | | | | | | F | T | T | $.9 \times .2 = .18$ |
| | | | | | | F | T | F | $.9 \times .8 = .72$ |
| | | | | | | F | F | T | $.1 \times .6 = .06$ |
| | | | | | | F | F | F | $.1 \times .4 = .04$ |

**Figure 14.10** Illustrating pointwise multiplication: $\mathbf{f}_1(A,B) \times \mathbf{f}_2(B,C) = \mathbf{f}_3(A,B,C)$.

# Example – Inference by Variable Elimination

- Sum out variables

To sum out A out of f3(A,B,C):

$$\begin{aligned}\mathbf{f}(B,C) &= \sum_a \mathbf{f}_3(A,B,C) = \mathbf{f}_3(a,B,C) + \mathbf{f}_3(\neg a,B,C) \\ &= \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix} .\end{aligned}$$

# Other topics

***9.*** *Convert a propositional or first-order logic sentence to CNF. Perform* <u>*Skolemization*</u>*. Apply standard logical rewritings.*

- Skolemization in FOL resolution
  - How to remove existential quantifier

$$\forall x \;\; [\forall y \;\; Animal(y) \;\Rightarrow\; Loves(x,y)] \;\Rightarrow\; [\exists y \;\; Loves(y,x)] \,.$$

$$\forall x \;\; [\exists y \;\; Animal(y) \land \neg Loves(x,y)] \lor [\exists z \;\; Loves(z,x)] \,.$$

$$\forall x \;\; [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(z),x) \,.$$

Skolem functions

# Example – Conversion to CNF

$$\forall x \;\; [\forall y \;\; Animal(y) \;\Rightarrow\; Loves(x,y)] \;\Rightarrow\; [\exists y \;\; Loves(y,x)] \; .$$

# Example – Conversion to CNF

$$\forall x\ [\forall y\ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ \ Loves(y,x)]\,.$$

1. Eliminate implications: $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
2. Move $\neg$ inwards
   - $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$
   - $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$ (De Morgan)
   - $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$ (double negation)
3. Standardize variables: $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$
4. Skolemization: $\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$
5. Drop universal quantifiers: $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$
6. Distribute $\vee$ over $\wedge$: $[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

# Example - resolution

$$\frac{[Animal(F(x)) \vee \boxed{Loves(G(x), x)}] \quad [\boxed{\neg Loves(u, v)} \vee \neg Kills(u, v)]}{Animal(F(x)) \vee \neg Kills(G(x), x)}$$

$\theta = \{u/G(x), v/x\}$

# Final review: one by one

- Underlined topics are not covered in previous discussion
- Otherwise you can find examples in previous discussion

1. A simple LISP programming exercise (one recursive function).

See Discussion 1 slides.

**2.** Formalize a real-world problem as a search or constraint satisfaction problem. Come up with an admissible heuristic. Determine branching factors and solution depths.

Midterm Q4.

**3.** Label nodes in a search tree according to the order in which they will be expanded/generated for any of the search algorithms.

Midterm Q1.

**4.** Determine completeness, optimality, time, and space complexity for any of the search algorithms.

Midterm Q3.

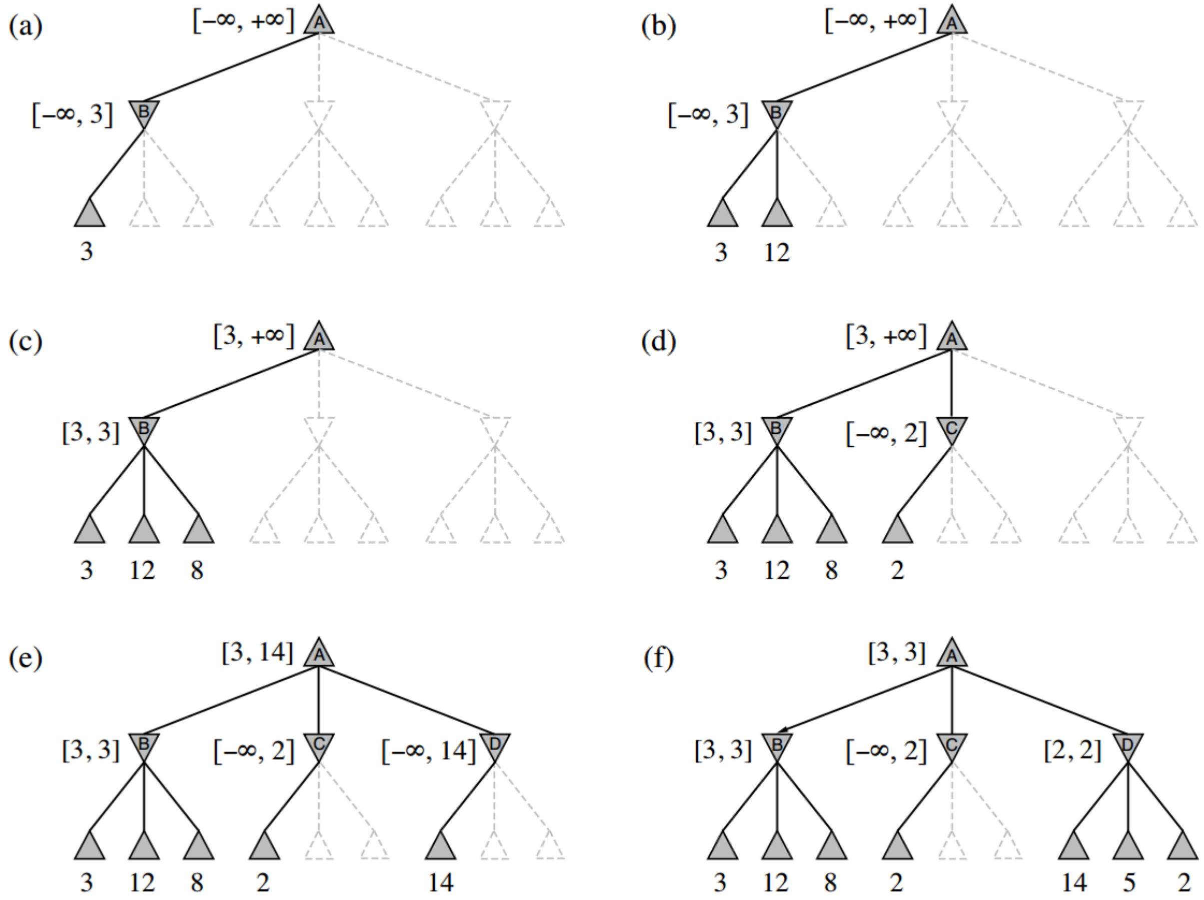
**5.** Perform steps of constraint satisfaction backtracking search, for various choices of variable order, value selection, and constraint propagation.
Midterm Q5.

Heuristics: MRV, Least restraining value, degree, ...

Forward checking and MAC (AC-3 algorithms)

**6.** Compute minimax or expectiminimax values to solve a game.
**7.** Perform $\alpha$-$\beta$ pruning on a given game tree.

Midterm Q2.



**8.** Model a problem as a propositional or first-order knowledge base, or as a Bayesian network.

**9.** Convert a propositional or first-order logic sentence to CNF. Perform Skolemization. Apply standard logical rewritings.

**10.** Reason using possible worlds/models (decide satisfiability, validity, compute probabilities, etc.).

**11.** Perform propositional or first-order resolution, unification, apply deductive inference rules, and perform simple DPLL, forward, or backward chaining.

# Discussion 9 - Resolution

- $\alpha$:

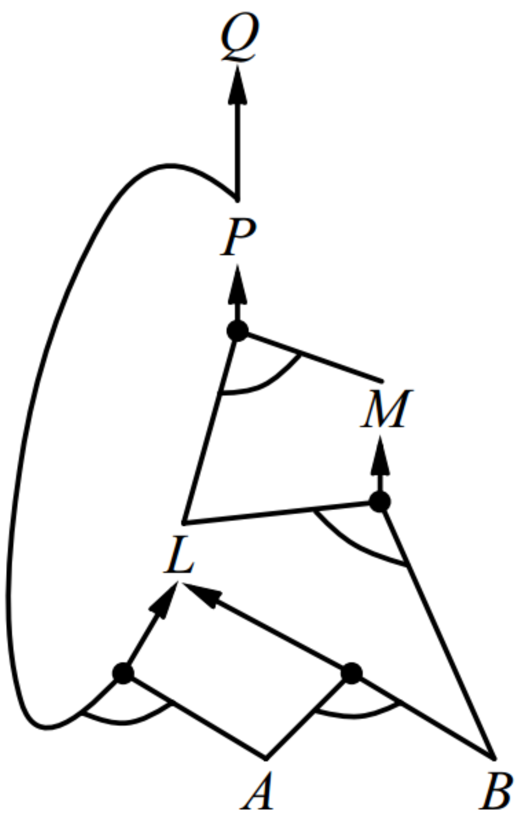
$\forall \mathrm{x}\ \mathrm{King}(\mathrm{x}) \Rightarrow \mathrm{Person}(\mathrm{x})$
$\forall \mathrm{x}, \mathrm{y}\ \mathrm{Person}(\mathrm{x}) \wedge \mathrm{Brother}(\mathrm{x}, \mathrm{y}) \Rightarrow \mathrm{Person}(\mathrm{y})$
$\mathrm{King}(\mathrm{Richard})$
$\mathrm{Brother}(\mathrm{Richard}, \mathrm{John})$

- $\beta$: $\mathrm{Person}(\mathrm{John})$

# Discussion 9 – Forward, Backward Chaining

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

(a)




(b)

**Figure 7.16** (a) A set of Horn clauses. (b) The corresponding AND–OR graph.

# DPLL

The SAT problem:

- Given a propositional formula
- Either find a satisfying assignment or show it's impossible

NP-complete!

DPLL:

- A complete backtracking algorithm

# DPLL

General idea

- Start from a ground CNF formula
- Try to build an assignment (partially, incrementally), verify
- Backtrack when it fails

Pay attention to:

- Pure-literal
- One-literal
- splitting

# DPLL

## Example (Example I)

$$S = (P \vee Q \vee \neg R) \wedge (P \vee \neg Q) \wedge \neg P \wedge R \wedge U$$

| | | |
|---|---|---|
| $\{\}$ | $(P \vee Q \vee \neg R) \wedge (P \wedge \neg Q) \wedge \neg P \wedge R \wedge U$ | One-Literal on $\neg P$ |
| $\{\neg P\}$ | $(Q \vee \neg R) \wedge \neg Q \wedge R \wedge U$ | One-Literal on $\neg Q$ |
| $\{\neg P, \neg Q\}$ | $\neg R \wedge R \wedge U$ | One-Literal on $R$ |
| $\{\neg P, \neg Q\}$ | $\neg R \wedge R \wedge U$ | One-Literal on $R$ |
| $\{\neg P, \neg Q, R\}$ | $\square \wedge U$ | unsatisfiable |

# DPLL

## Example (Example II)

$$S = (P \vee Q) \wedge \neg Q \wedge (\neg P \vee Q \vee \neg R)$$

| | | |
|---|---|---|
| $\{\}$ | $S = (P \vee Q) \wedge \neg Q \wedge (\neg P \vee Q \vee \neg R)$ | One-Literal on $\neg Q$ |
| $\{\neg Q\}$ | $P \wedge (\neg P \vee \neg R)$ | One-Literal on $P$ |
| $\{\neg Q, P\}$ | $\neg R$ | One-Literal on $\neg R$ |
| $\{\neg Q, P, \neg R\}$ | $\{\}$ | Satisfiable |

# DPLL - rules

- One-literal

$$S = \{P \vee Q \vee \neg R, P \vee \neg Q, \neg P, R, U\}$$

$$S' = \{P \vee Q \vee \neg R, P \vee \neg Q, R, U\}$$

- Pure-literal

$$S = \{P \vee Q, P \vee \neg Q, R \vee Q, R \vee \neg Q\}$$

$$S' = \{R \vee Q, R \vee \neg Q\}$$

# DPLL - rules

- Splitting

$S = \{P \vee \neg Q \vee R, \neg P \vee Q, Q \vee \neg R, \neg Q \vee \neg R\}$

Apply Splitting on $P$

$S' = \{\neg Q \vee R, Q \vee \neg R, \neg Q \vee \neg R\},\ P = \bot$

$S'' = \{Q, Q \vee \neg R, \neg Q \vee \neg R\},\ P = \top$

# DPLL - rules

$$S = \{P \vee Q \vee \neg R, P \vee \neg Q, \neg P, R, U\}$$

$$S' = \{P \vee Q \vee \neg R, P \vee \neg Q, R, U\}$$

$$S = \{P \vee Q, P \vee \neg Q, R \vee Q, R \vee \neg Q\}$$

$$S' = \{R \vee Q, R \vee \neg Q\}$$

# DPLL

## Example (Example III)

$$S = (P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg R)$$

| | | |
|---|---|---|
| $\{\,\}$ | $(P \vee Q) \wedge (P \wedge \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg R)$ | Split on $P$ |
| $S'$ $\{\neg P\}$ | $Q \wedge \neg Q$ | One-Literal on $Q$ |
| $S'$ $\{\neg P, Q\}$ | $\square$ | $S'$ Unsat |
| | Backtrack | |

# DPLL

## Example (Example III)

$$S = (P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg R)$$

| | | |
|---|---|---|
| $\{\}$ | $(P \vee Q) \wedge (P \wedge \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg R)$ | Split on $P$ |
| $S'$ $\{\neg P\}$ | $Q \wedge \neg Q$ | One-Literal on $Q$ |
| $S'$ $\{\neg P, Q\}$ | $\square$ | $S'$ Unsat |
| | Backtrack | |

# DPLL

## Example (Example III)

$$S = (P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg R)$$

| | | |
|---|---|---|
| $\{\}$ | $(P \vee Q) \wedge (P \wedge \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg R)$ | Split on $P$ |
| $S''$ $\{P\}$ | $Q \wedge \neg R$ | One-Literal on $Q$ |
| $S''$ $\{P, Q\}$ | $\neg R$ | One-Literal on $\neg R$ |
| $S''$ $\{P, Q, \neg R\}$ | $\{\}$ | Satisfiable! |

**function** DPLL-SATISFIABLE?($s$) **returns** *true* or *false*
  **inputs**: $s$, a sentence in propositional logic

  $clauses \leftarrow$ the set of clauses in the CNF representation of $s$
  $symbols \leftarrow$ a list of the proposition symbols in $s$
  **return** DPLL($clauses, symbols, \{\ \}$)

---

**function** DPLL($clauses, symbols, model$) **returns** *true* or *false*

  **if** every clause in $clauses$ is true in $model$ **then return** *true*
  **if** some clause in $clauses$ is false in $model$ **then return** *false*
  $P, value \leftarrow$ FIND-PURE-SYMBOL($symbols, clauses, model$)
  **if** $P$ is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P, value \leftarrow$ FIND-UNIT-CLAUSE($clauses, model$)
  **if** $P$ is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)
  **return** DPLL($clauses, rest, model \cup \{P{=}true\}$) **or**
      DPLL($clauses, rest, model \cup \{P{=}false\}$))

**Figure 7.17** The DPLL algorithm for checking satisfiability of a sentence in propositional logic. The ideas behind FIND-PURE-SYMBOL and FIND-UNIT-CLAUSE are described in the text; each returns a symbol (or null) and the truth value to assign to that symbol. Like TT-ENTAILS?, DPLL operates over partial models.

**12.** Basic probabilistic reasoning (inclusion-exclusion, marginalization, conditioning, Bayes rule) and checking properties (conditional independence).

**13.** Identify conditional independence assumptions and joint distribution encoded by a Bayesian network (its semantics).

**14.** Perform Bayesian network inference by enumeration. Multiply factors and sum out a variable from a factor.

**15.** Compute the <u>size of a hypothesis space</u>.

**16.** Learn a decision tree from data and identify optimal tests.

# Size of Hypothesis space

A hypothesis is a function
h: $\mathcal{X} \longrightarrow \mathcal{Y}$

$\mathcal{X}$ : feature space (set of all possible inputs)
$\mathcal{Y}$ : label space

- Occam's razor: maximize a combination of consistency and simplicity

# Exercise - Size of Hypothesis space

Each datapoint has 2 binary features. Each feature can take on 2 values, either a 0 or a 1.

2 possible labels: y can either be a 0 or a 1.

What's the size of hypothesis space?

# Size of Hypothesis space

$$\mathcal{X} = \{0,1\}^2 = \{(0,0),(0,1),(1,0),(1,1)\},$$
$$\mathcal{Y} = \{0,1\}.$$

For each x in $\chi$, two possible labels
4 possible inputs in $\chi$

Size of hypothesis space: 2^4