

Inference in Belief Networks

CMPUT 366: Intelligent Systems

P&M §8.4

Assignment #1

- Assignment #1 is due **today** at 11:59pm
- Submit via eclass: zipfile containing:
 - All code (yours and provided utility code)
 - PDF of problem set solutions

Lecture Outline

1. Recap
2. Factors
3. Variable Elimination
4. Efficiency

Recap: Belief Networks

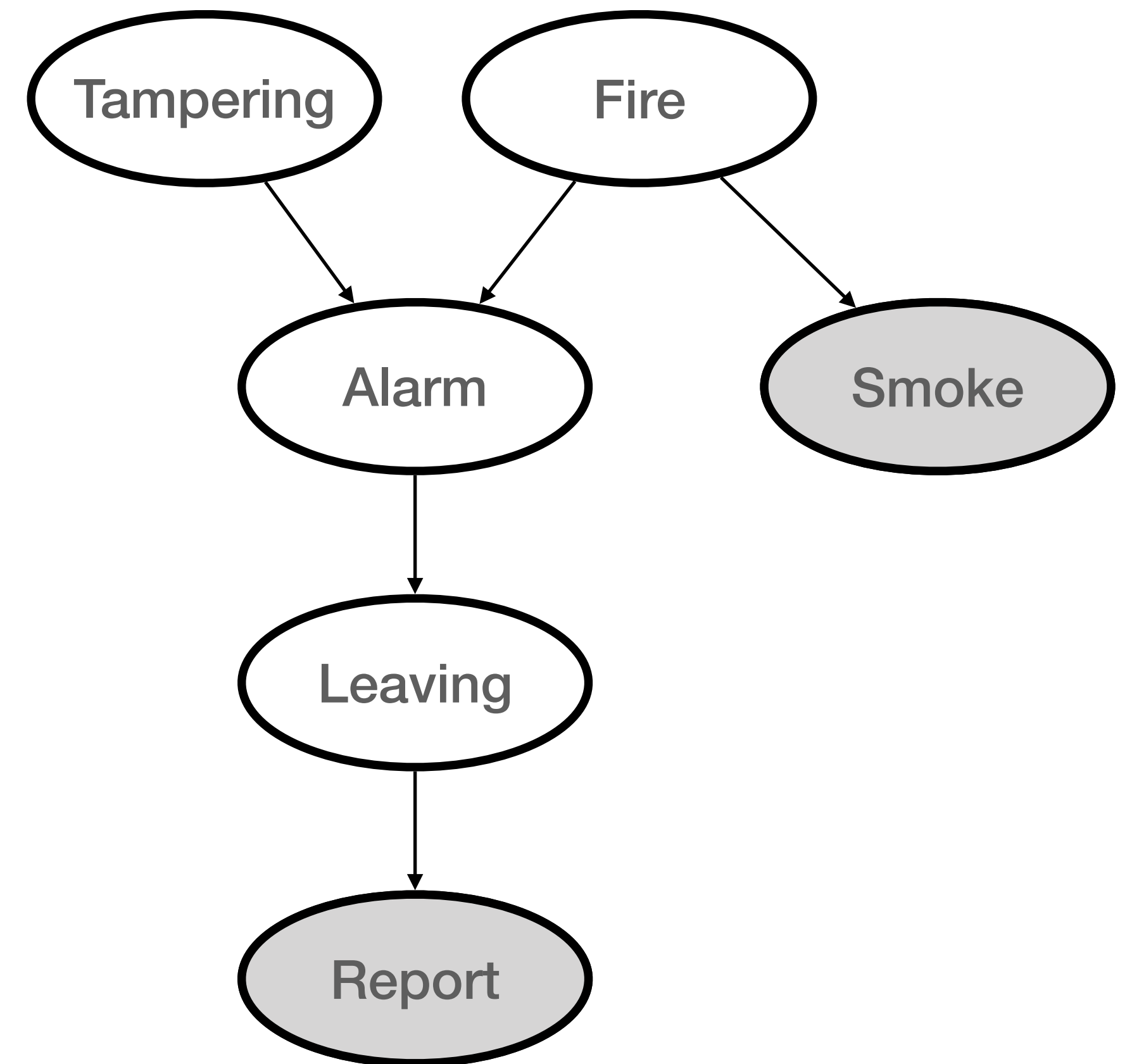
Definition:

A **belief network** (or **Bayesian network**) consists of:

1. A directed acyclic graph, with each node labelled by a **random variable**
 2. A **domain** for each random variable
 3. A **conditional probability table** for each variable given its **parents**
- The graph represents a specific **factorization** of the full **joint distribution**
 - **Semantics:**
Every node is **independent** of its **non-descendants**, **conditional** on its **parents**

Recap: Queries

- The most common task for a belief network is to query **posterior probabilities** given some **observations**
- **Easy cases:**
 - Posteriors of a **single variable** conditional only on **parents**
 - **Joint distributions** of variables early in a **compatible variable ordering**
- Typically, the observations have **no straightforward relationship** to the target
- **This lecture:** mechanical procedure for computing **arbitrary queries**



Factors

- The **Variable Elimination** algorithm exploits the **factorization** of a joint probability distribution encoded by a belief network in order to answer **queries**
- A **factor** is a function $f(X_1, \dots, X_k)$ from **random variables** to a **real number**
- **Input:** factors representing the **conditional probability tables** from the belief network

$P(\text{Leaving} \mid \text{Alarm})P(\text{Smoke} \mid \text{Fire})P(\text{Alarm} \mid \text{Tampering}, \text{Fire})P(\text{Tampering})P(\text{Fire})$

becomes

$f_1(\text{Leaving}, \text{Alarm})f_2(\text{Smoke}, \text{Fire})f_3(\text{Alarm}, \text{Tampering}, \text{Fire})f_4(\text{Tampering})f_5(\text{Fire})$

- **Output:** A **new factor** encoding the target **posterior distribution**

E.g., $f_{12}(\text{Tampering})$.

Conditional Probabilities as Factors

- A **conditional probability** $P(Y | X_1, \dots, X_n)$ is a factor $f(Y, X_1, \dots, X_n)$ that obeys the **constraint**:

$$\forall v_1 \in \text{dom}(X_1), v_2 \in \text{dom}(X_2), \dots, v_n \in \text{dom}(X_n) : \left[\sum_{y \in \text{dom}(Y)} f(y, v_1, \dots, v_n) \right] = 1.$$

- Answer to a query is a factor **constructed by applying operations** to the input factors
 - Operations on factors are not guaranteed to **maintain** this constraint!
 - Solution: **Don't sweat it!**
 - Operate on **unnormalized probabilities** during the computation
 - **Normalize** at the end of the algorithm to re-impose the constraint

Conditioning

Conditioning is an operation on a **single factor**

- Constructs a **new factor** that returns the values of the original factor with some of its inputs fixed

Definition:

For a factor $f_1(X_1, \dots, X_k)$, **conditioning on** $X_i = v_i$ yields a new factor

$$f_2(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_k) = (f_1)_{X_i=v_i}$$

such that for all values $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_k$ in the domain of $X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_k$,

$$f_2(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_k) = f_1(v_1, \dots, v_{i-1}, \mathbf{v}_i, v_{i+1}, \dots, v_k).$$

Conditioning Example

$$f_2(A, B) = f_1(A, B, C)_{C=true}$$

f_1

A	B	C	value
F	F	F	0.1
F	F	T	0.88
F	T	F	0.12
F	T	T	0.45
T	F	F	0.7
T	F	T	0.66
T	T	F	0.1
T	T	T	0.25

f_2

A	B	value
F	F	0.88
F	T	0.45
T	F	0.66
T	T	0.25

Multiplication

Multiplication is an operation on **two factors**

- Constructs a new factor that returns the **product** of the rows selected from each factor by its arguments

Definition:

For two factors $f_1(X_1, \dots, X_j, Y_1, \dots, Y_k)$ and $f_2(Y_1, \dots, Y_k, Z_1, \dots, Z_\ell)$,

multiplication of f_1 and f_2 yields a new factor

$$(f_1 \times f_2) = f_3(X_1, \dots, X_j, Y_1, \dots, Y_k, Z_1, \dots, Z_\ell)$$

such that for all values $x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_\ell$,

$$f_3(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_\ell) = f_1(x_1, \dots, x_j, y_1, \dots, y_k) f_2(y_1, \dots, y_k, z_1, \dots, z_\ell).$$

Multiplication Example

$$f_3(A, B, C) = f_1(A, B) \times f_2(B, C)$$

f_1

A	B	value
F	F	0.1
F	T	0.2
T	F	0.3
T	T	0.4

f_2

B	C	value
F	F	1.0
F	T	0
T	F	0.5
T	T	0.25

f_3

A	B	C	value
F	F	F	0.1
F	F	T	0
F	T	F	0.1
F	T	T	0.05
T	F	F	0.3
T	F	T	0
T	T	F	0.2
T	T	T	0.1

Summing Out

Summing out is an operation on a **single factor**

- Constructs a new factor that returns the **sum over all values** of a random variable of the original factor

Definition:

For a factor $f_1(X_1, \dots, X_k)$, summing out a variable X_i yields a new factor

$$f_2(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_k) = \left(\sum_{X_i} f_1 \right)$$

such that for all values $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_k$ in the domain of $X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_k$

$$f_2(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_k) = \sum_{\mathbf{v}_i \in \text{dom}(X_i)} f_1(v_1, \dots, v_{i-1}, \mathbf{v}_i, v_{i+1}, \dots, v_k).$$

Summing Out Example

$$f_2(B) = \sum_A f_1(A, B)$$

f_1

A	B	value
F	F	0.1
F	T	0.2
T	F	0.3
T	T	0.4

f_2

B	value
F	0.4
T	0.6

Variable Elimination

- Given **observations** $Y_1 = v_1, \dots, Y_k = v_k$ and query variable Q , we want

$$P(Q \mid Y_1 = v_1, \dots, Y_k = v_k) = \frac{P(Q, Y_1 = v_1, \dots, Y_k = v_k)}{\sum_{q \in \text{dom}(Q)} P(Q = q, Y_1 = v_1, \dots, Y_k = v_k)}.$$

- Basic idea of variable elimination:

1. Condition on observations by **conditioning**

2. Construct joint distribution factor by **multiplication**

3. Remove unwanted variables (neither query nor observed) by **summing out**

4. **Normalize** at the end

- Doing these steps in order is **correct** but not **efficient**
- Efficiency comes from **interleaving** the order of operations

Sums of Products

2. Construct joint distribution factor by **multiplication**
3. Remove unwanted variables (neither query nor observed) by **summing out**

The computationally intensive part of variable elimination is computing **sums of products**

Example: multiply factors $f_1(Q, A, B, C)$, $f_2(C, D, E)$; sum out A, E

1. $f_3(Q, A, B, C, D, E) = f_1(Q, A, B, C) \times f_2(C, D, E) : 2^6$ multiplications

2. $f_4(Q, B, C, D) = \sum_{A, E} f_3(Q, A, B, C, D, E) : 3 \times 16$ additions

Total: **112** computations

(*) For all numerical examples, we assume binary domains

Efficient Sums of Products

We can reduce the number of computations required by changing their **order**.

$$\begin{aligned} & \sum_A \sum_E f_1(Q, A, B, C) \times f_2(C, D, E) \\ &= \left(\sum_A f_1(Q, A, B, C) \right) \times \left(\sum_E f_2(C, D, E) \right) \end{aligned}$$

1. $f_3(C, D) = \sum_E f_2(C, D, E) : 2^2$ additions
2. $f_4(Q, B, C) = \sum_A f_1(Q, A, B, C) : 2^3$ additions
3. $f_5(Q, B, C, D) = f_3(Q, B, C) \times f_4(B, C, D) : 2^4$ multiplications

Total: **28** computations

Variable Elimination Algorithm

Input: query variable Q ; set of variables Vs ; observations O ; factors Ps representing conditional probability tables

$Fs := Ps$

for each X in $Vs \setminus \{Q\}$ according to some **elimination ordering**:

$R_s = \{ F \text{ in } F_s \mid F \text{ involves } X \}$

if X is observed:

for each F in R_s :

$F' = F$ **conditioned** on observed value of X

$F_s = F_s \setminus \{F\} \cup \{F'\}$

else:

$T :=$ **product** of factors in R_s

$N :=$ **sum** X out of T

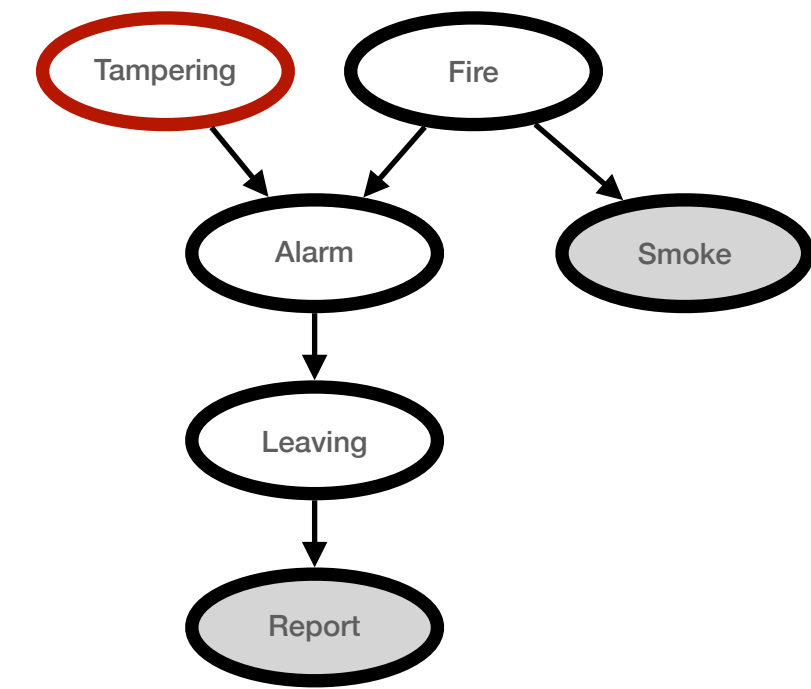
$F_s := F_s \setminus R_s \cup \{N\}$

$T :=$ **product** of factors in F_s

$N :=$ **sum** Q out of T

return T / N

Variable Elimination Example: Conditioning



Query: $P(\text{Tampering} \mid \text{Smoke}=\text{true}, \text{Report}=\text{true})$

Variable ordering: Smoke, Report, Fire, Alarm, Leaving

$P(\text{Tampering}, \text{Fire}, \text{Alarm}, \text{Smoke}, \text{Leaving}, \text{Report}) =$
 $P(\text{Tampering})P(\text{Fire})P(\text{Alarm} \mid \text{Tampering}, \text{Fire})P(\text{Smoke} \mid \text{Fire})P(\text{Leaving} \mid \text{Alarm})P(\text{Report} \mid \text{Leaving})$

Construct **factors** for each table:

$\{ f_0(\text{Tampering}), f_1(\text{Fire}), f_2(\text{Tampering}, \text{Alarm}, \text{Fire}), f_3(\text{Smoke}, \text{Fire}), f_4(\text{Leaving}, \text{Alarm}), f_5(\text{Report}, \text{Leaving}) \}$

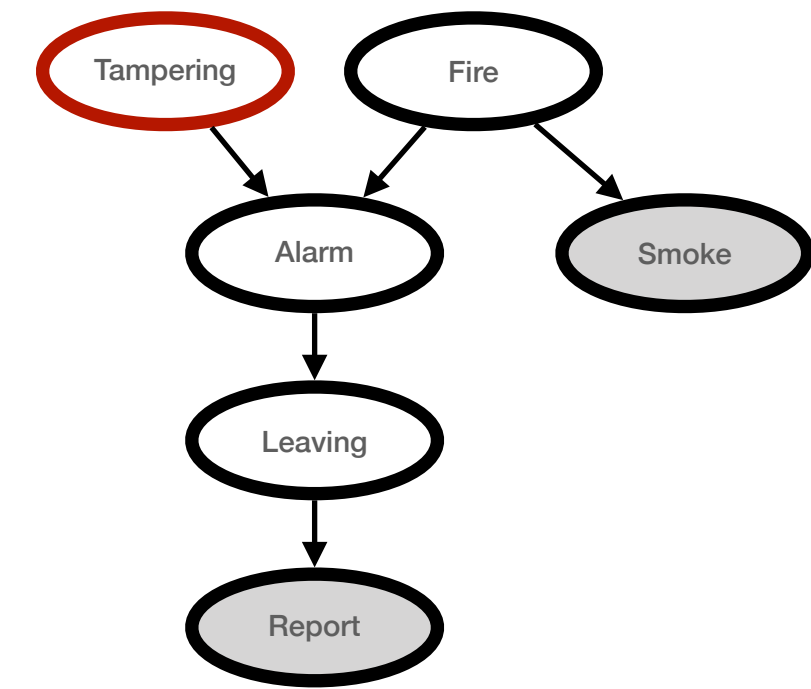
Condition on Smoke: $f_6 = (f_3)_{\text{Smoke}=\text{true}}$

$\{ f_0(\text{Tampering}), f_1(\text{Fire}), f_2(\text{Tampering}, \text{Alarm}, \text{Fire}), f_6(\text{Fire}), f_4(\text{Leaving}, \text{Alarm}), f_5(\text{Report}, \text{Leaving}) \}$

Condition on Report: $f_7 = (f_5)_{\text{Report}=\text{true}}$

$\{ f_0(\text{Tampering}), f_1(\text{Fire}), f_2(\text{Tampering}, \text{Alarm}, \text{Fire}), f_6(\text{Fire}), f_4(\text{Leaving}, \text{Alarm}), f_7(\text{Leaving}) \}$

Variable Elimination Example: Elimination



Query: $P(\text{Tampering} \mid \text{Smoke}=\text{true}, \text{Report}=\text{true})$

Variable ordering: ~~Smoke, Report~~, Fire, Alarm, Leaving

$\{ f_0(\text{Tampering}), f_1(\text{Fire}), f_2(\text{Tampering}, \text{Alarm}, \text{Fire}), f_6(\text{Fire}), f_4(\text{Leaving}, \text{Alarm}), f_7(\text{Leaving}) \}$

Sum out Fire from **product** of f_1, f_2, f_6 : $f_8 = \sum_{\text{Fire}} (f_1 \times f_2 \times f_6)$

$\{ f_0(\text{Tampering}), f_8(\text{Tampering}, \text{Alarm}), f_4(\text{Leaving}, \text{Alarm}), f_7(\text{Leaving}) \}$

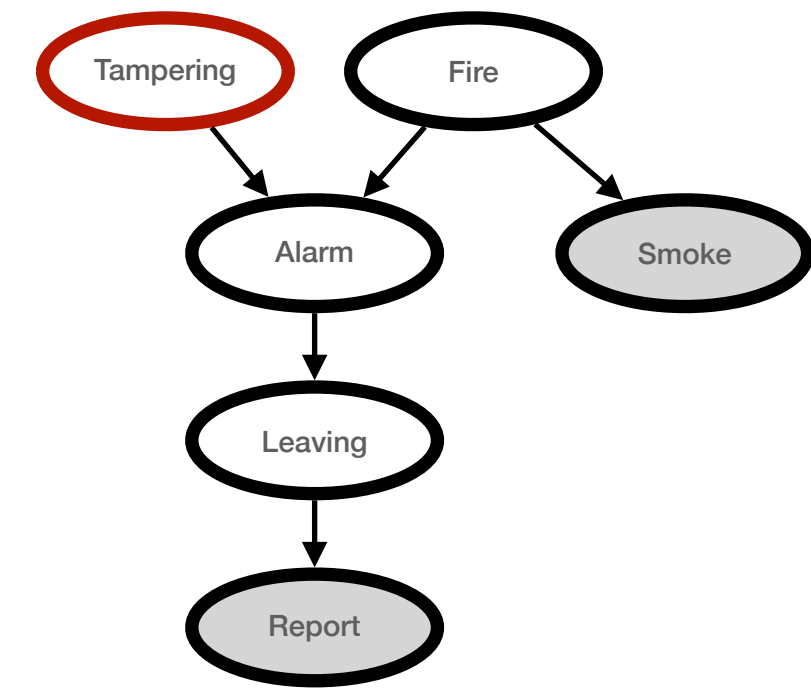
Sum out Alarm from **product** of f_8, f_4 : $f_9 = \sum_{\text{Alarm}} (f_8 \times f_4)$

$\{ f_0(\text{Tampering}), f_9(\text{Tampering}, \text{Leaving}), f_7(\text{Leaving}) \}$

Sum out Leaving from **product** of f_9, f_7 : $f_{10} = \sum_{\text{Leaving}} (f_9 \times f_7)$

$\{ f_0(\text{Tampering}), f_{10}(\text{Tampering}) \}$

Variable Elimination Example: Normalization



Query: $P(\text{Tampering} \mid \text{Smoke}=\text{true}, \text{Report}=\text{true})$

Variable ordering: ~~Smoke, Report, Fire, Alarm, Leaving~~
{ $f_0(\text{Tampering})$, $f_{10}(\text{Tampering})$ }

Product of remaining factors: $f_{11} = f_0 \times f_{10}$
{ $f_{11}(\text{Tampering})$ }

Normalize by division:

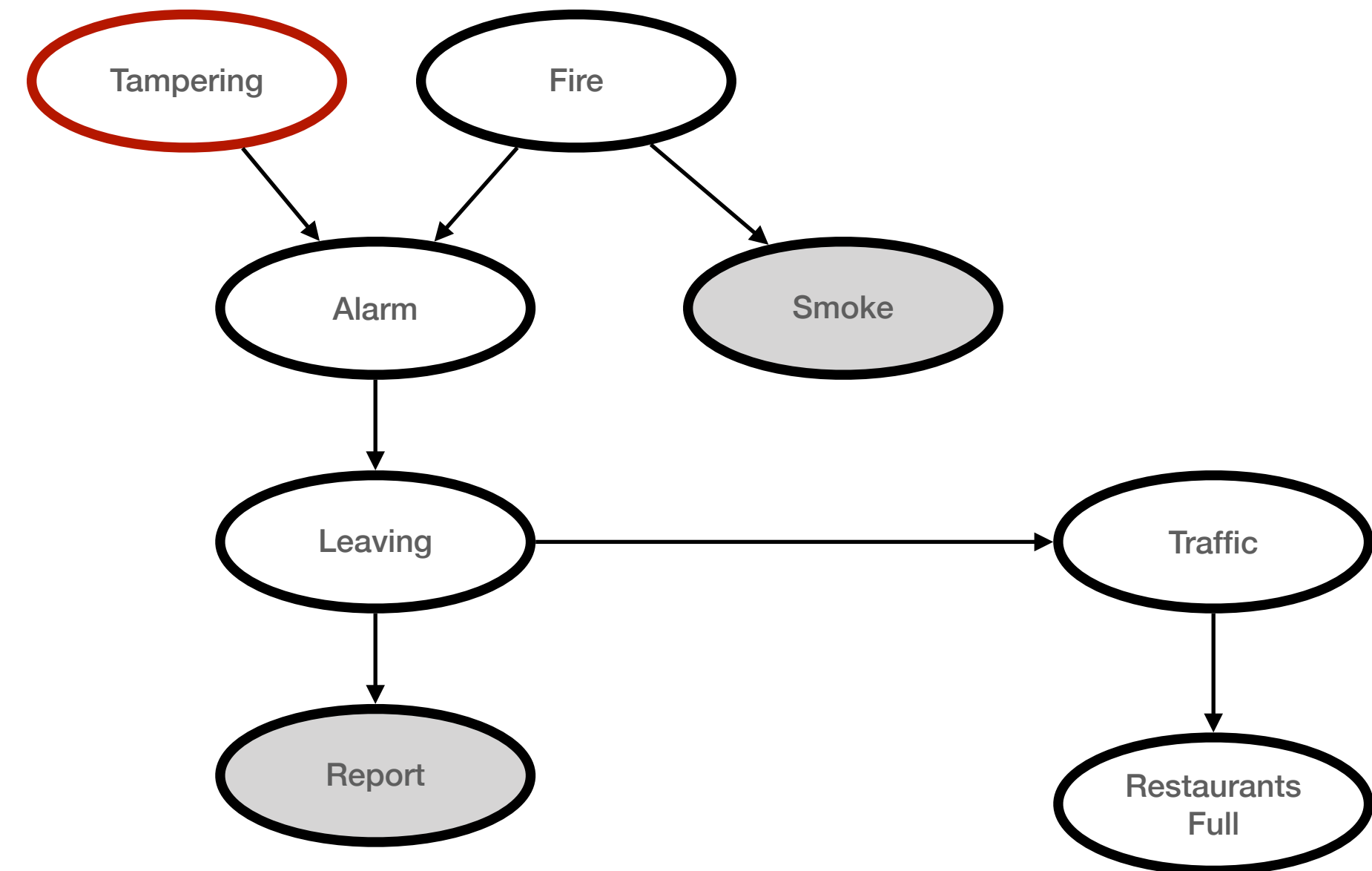
$\text{query}(\text{Tampering}) = f_{11}(\text{Tampering}) / (\sum_{\text{Tampering}} f_{11}(\text{Tampering}))$

Optimizing Elimination Order

- Variable elimination exploits **efficient sums of products** on a **factored joint distribution**
- The **elimination order** of the variables affects the **efficiency** of the algorithm
- Finding an **optimal** elimination ordering is **NP-hard**
- **Heuristics** (rules of thumb) for good orderings:
 - **Observations first:** Condition on all of the observed variables first
 - **Min-factor:** At every stage, select the variable that constructs the **smallest new factor**
 - Problem-specific heuristics

Optimization: Pruning

- The structure of the graph can allow us to **drop leaf nodes** that are **neither observed nor queried**
 - Summing them out for **free**
- We can **repeat** this process:



Optimization: Preprocessing

Finally, if we know that we are always going to be observing and/or querying the same variables, we can **preprocess** our graph; e.g.:

1. Precompute the **joint distribution** of all the variables we will observe and/or query
2. Precompute **conditional distributions** for our exact queries

Summary

- **Variable elimination** is an algorithm for answering **queries** based on a **belief network**
- Operates by using three **operations** on **factors** to reduce graph to a single posterior distribution
 1. Conditioning
 2. Multiplication
 3. Summing out
- **Distributes** operations more efficiently than taking full product and then summing out
 - **Optimal** order of operations is **NP-hard** to compute
- Additional **optimization** techniques: heuristic ordering, pruning, precomputation