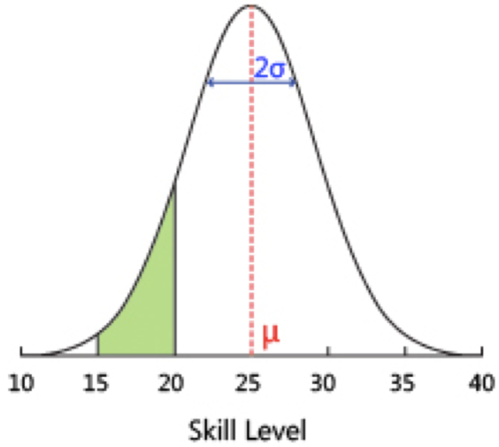
Learning Analytics

Dr. Bowen Hui
Computer Science
University of British Columbia Okanagan

Real World BN Applications: 2005

- MSR TrueSkill Ranking System in Xbox Live
- Identify and track skills of gamers in order to match them into competitive matches
- Update skill estimates using final standings of all teams in a game
- Simple model using:
  - Average skill of the gamer, μ
  - Degree of uncertainty in the gamer's skill, σ



- Wider σ denotes more uncertainty
- Shaded area is the probability that the user's skill is between level 15-20

  - Task: What is the probability that you will draw with another player? High chance means good competitive match!

- See http://www.moserware.com/2010/03/computing-your-skill.html

Real World BN Applications: 2007

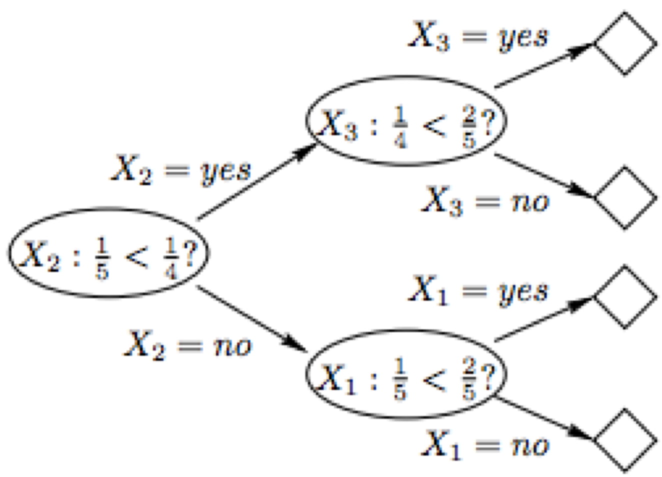
- Bonaparte, forensic software used to identify missing persons
  - Includes functionality for disaster victim identification, familial search, kinship analysis
- Consists of forensic statistical computations based on DNA profiles

- Task: Compute likelihood ratio of two hypotheses
  - Given DNA from family of missing person (MP) and DNA of unidentified individual (UI), is MP=UI?
    Or is MP some unrelated person (UN), is MP=UN?

- See http://www.dnadvi.nl/technology.php

Real World BN Applications: 2004

- Educational testing services (ETS) adaptive testing
  - Replaces a "fixed test" with a fixed sequence of questions covering all testable skills
  - Build an "adaptive test" where the answers of previous questions determines the choice of next question
- Model consists of:
  - A set of testable skills, **S**
  - A set of questions in a test bank, **X**
  - $\Pr(\boldsymbol{S}, \boldsymbol{X}) = \Pr(\boldsymbol{S}) \prod_{X_j \in \boldsymbol{X}} \Pr(Xj | \boldsymbol{S}^j)$



- Right answer goes to harder question
- Wrong answer goes to easier question
- Stop when enough info is gathered, or when all test questions are exhausted
- Algorithm finds a path to max info gain

Inference Task

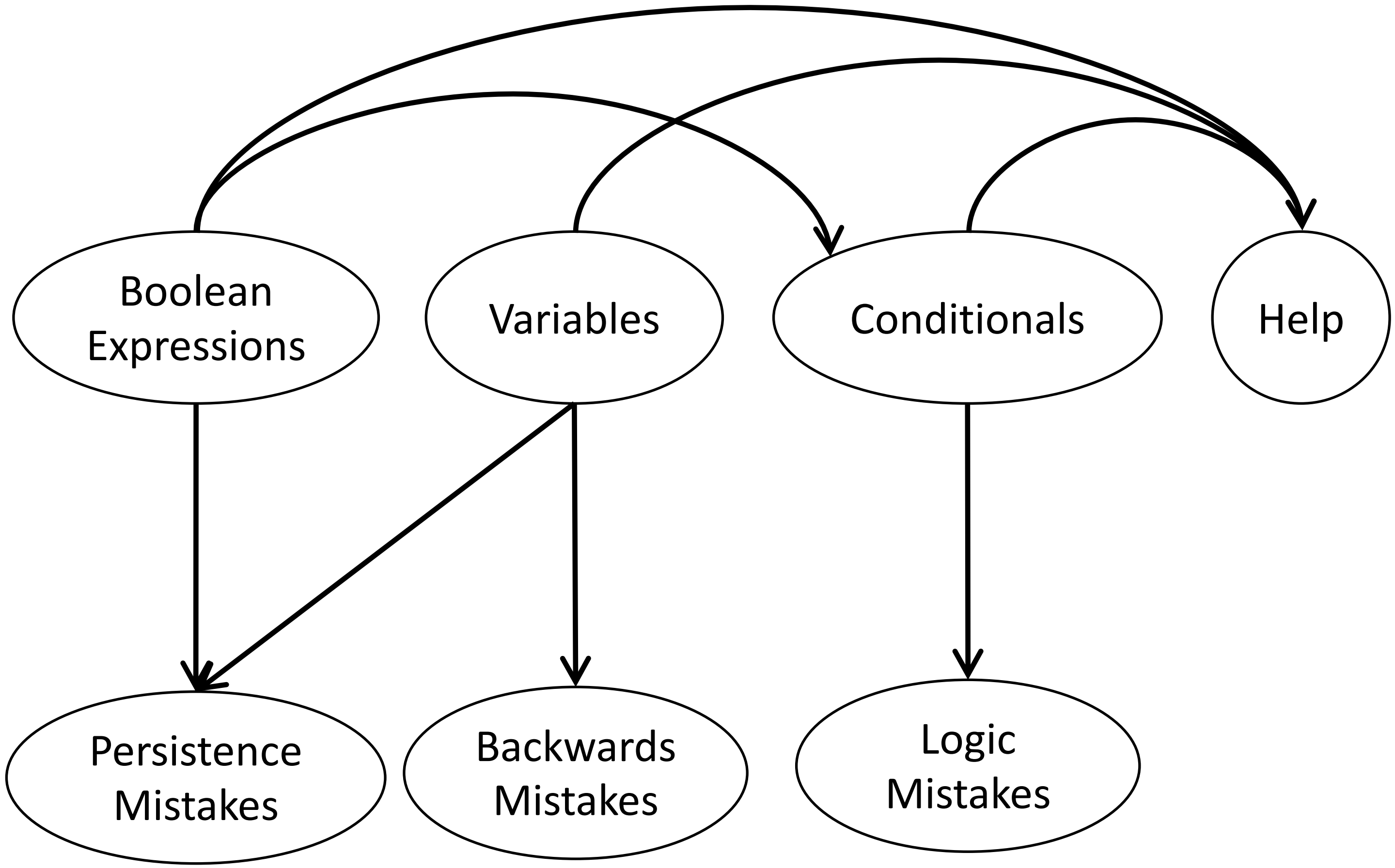
- Given:
  - Query variables: **X**
  - Evidence (observed) variables: **E = e**
  - Unobserved variables: **Y**

- Goal: To calculate useful information about the query variables
  - Updating belief of **X**: Pr(**X**|**e**)
  - Most probable explanation: $\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{e})$

- Recall inference via the full joint distribution:

$$\Pr(\boldsymbol{X}|\boldsymbol{E} = \boldsymbol{e}) = \frac{\Pr(\boldsymbol{X},\boldsymbol{e})}{\Pr(\boldsymbol{e})} \propto \sum_{Y} \Pr(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{e})$$

Which Task Is It?

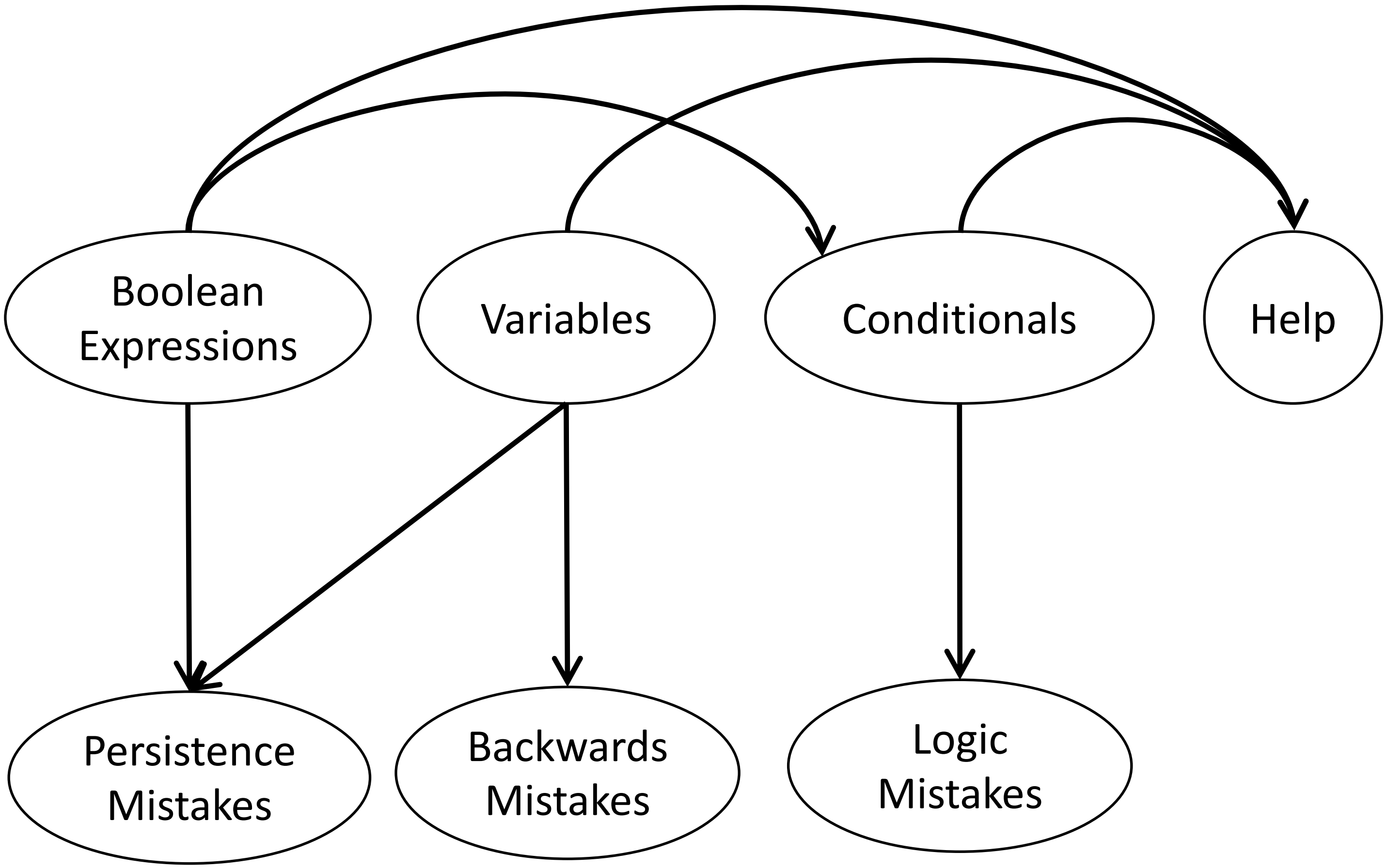


How likely is the student to need help?

a) Belief update: $\Pr(\mathbf{X}|\mathbf{e})$

b) Most probable explanation: $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{e})$

Which Task Is It?



What is the most likely cause for observing a lot of persistence mistakes?

a) Belief update: $\Pr(\mathbf{X}|\mathbf{e})$
b) Most probable explanation: $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{e})$

Inference Task

- Given:
  - Query variables: **X**
  - Evidence (observed) variables: **E = e**
  - Unobserved variables: **Y**

- Goal: To calculate useful information about the query variables
  - Updating belief of **X**: Pr(**X**|**e**)
  - Most probable explanation: $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{e})$

- Recall inference via the full joint distribution:

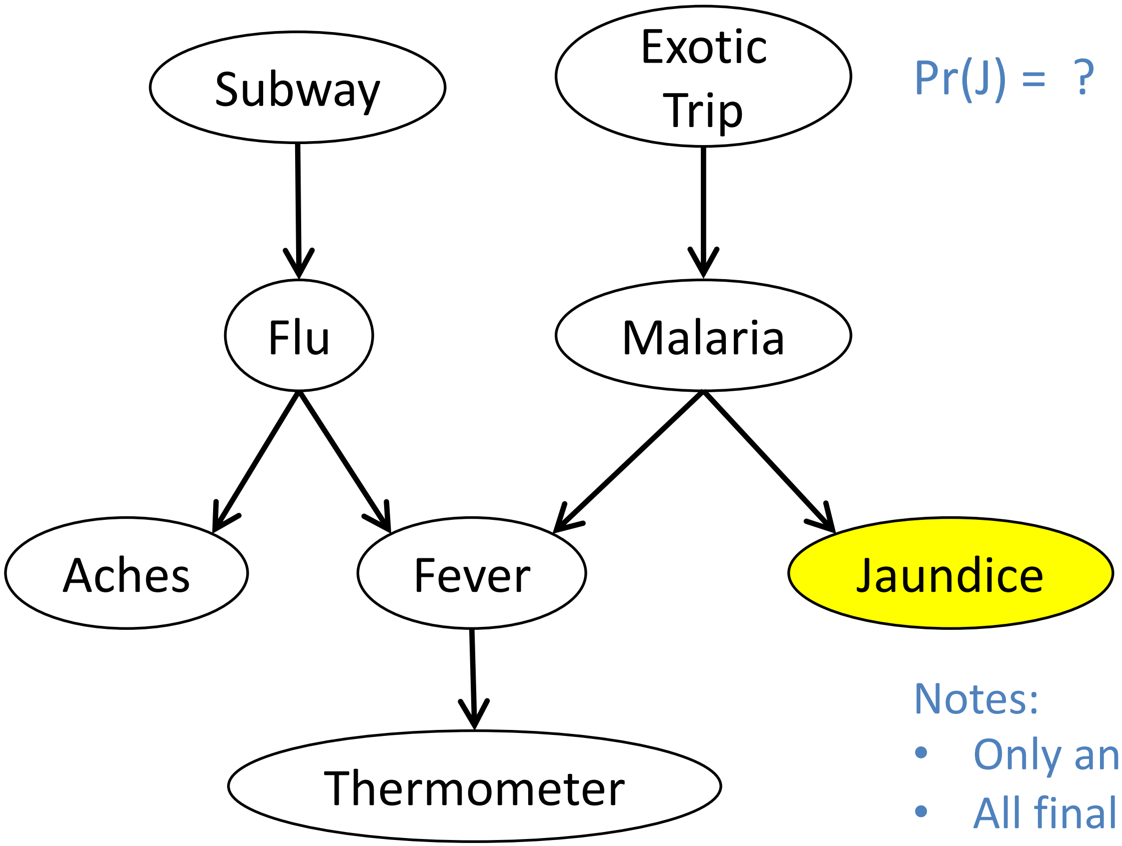
$$\Pr(\boldsymbol{X}|\boldsymbol{E}=\boldsymbol{e}) = \frac{\Pr(\boldsymbol{X},\boldsymbol{e})}{\Pr(\boldsymbol{e})} \propto \sum_{Y} \Pr(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{e})$$

This lecture: Inference Algorithms

- Purpose:
  - Understand the intuition behind the computation steps involved in an inference algorithm
- Need to know:
  - Given a problem, know how to formulate the query for inference task
    - Which are your query variable(s)?
    - Which are your evidence variable(s)?
  - Given a problem, know if you are doing belief update or MPE
- Explore inference algorithms further if desired

Simple Forward Inference

- Computing marginal requires simple forward "propagation" of probabilities



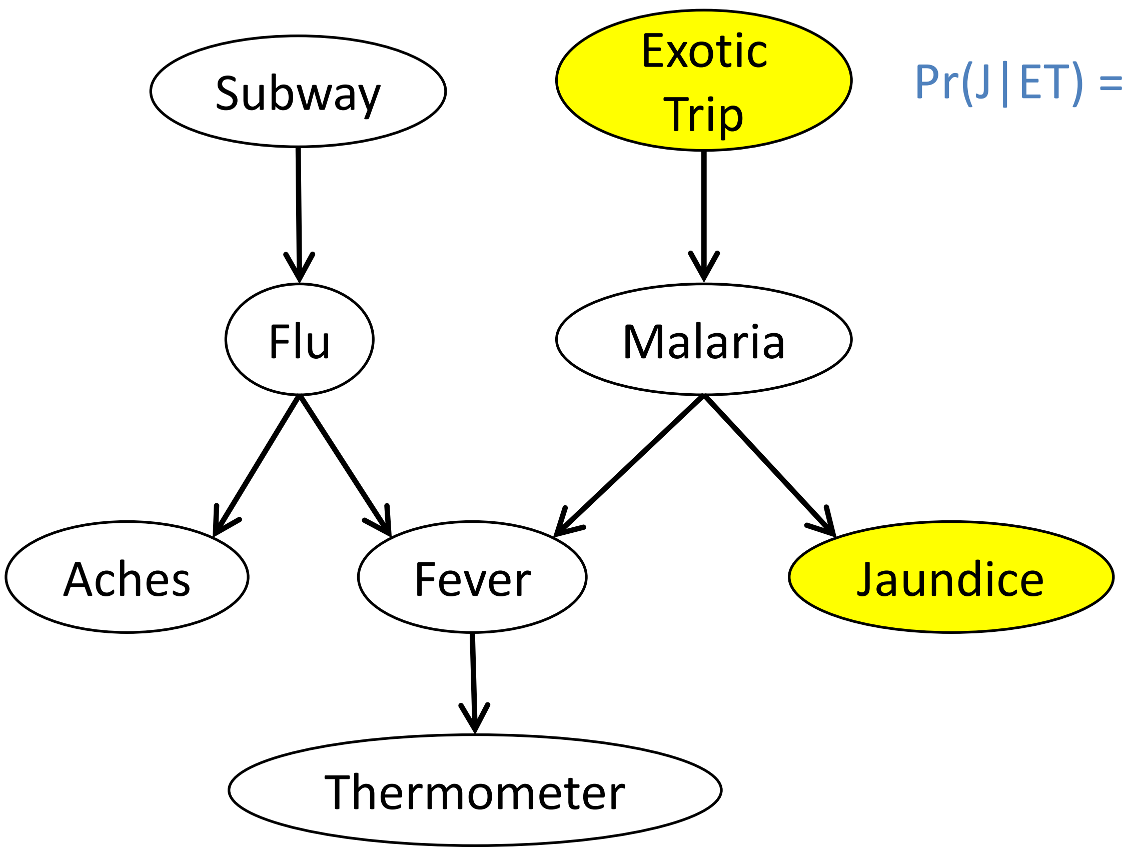
Pr(J) = ?

Notes:

- Only ancestors of J considered
- All final terms are CPTs in the BN

Simple Forward Inference

- Same idea applies when we have upstream evidence

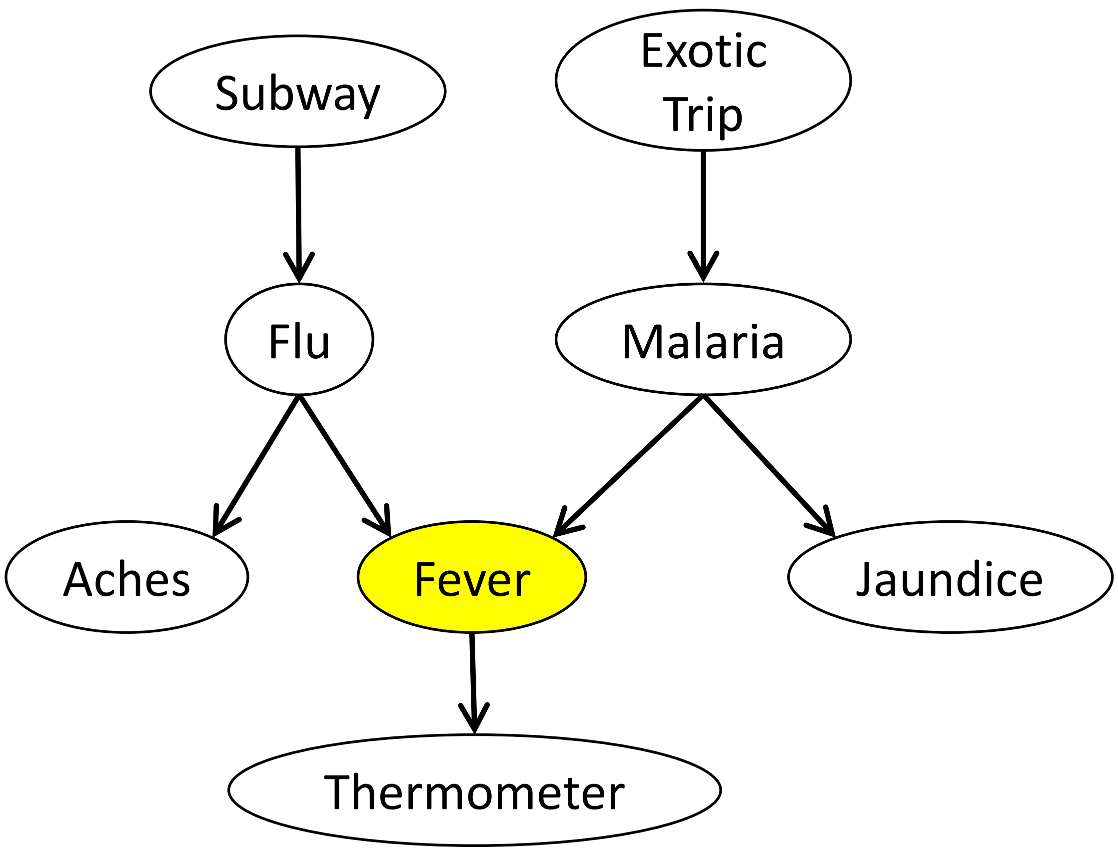


$\Pr(J|ET) = ?$

Simple Forward Inference

- Same idea applies with multiple parents

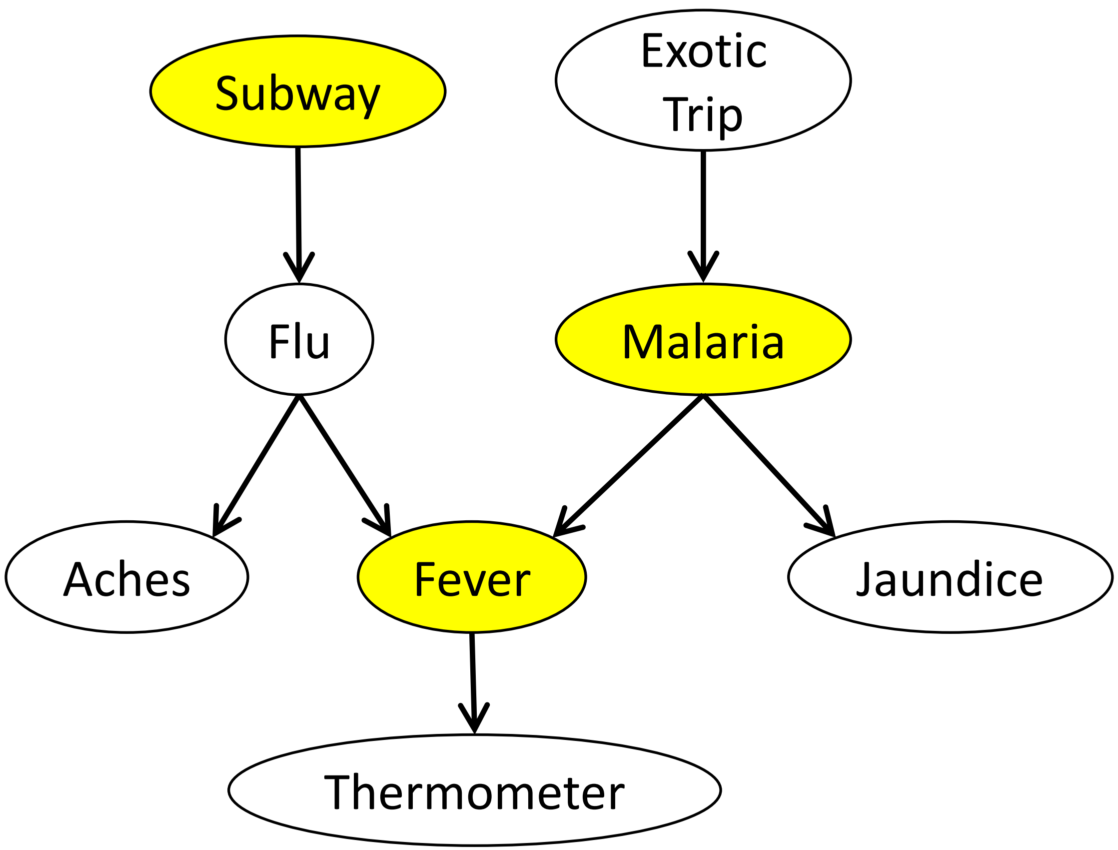
Pr(Fever) = ?



Simple Forward Inference

- Same idea applies with evidence

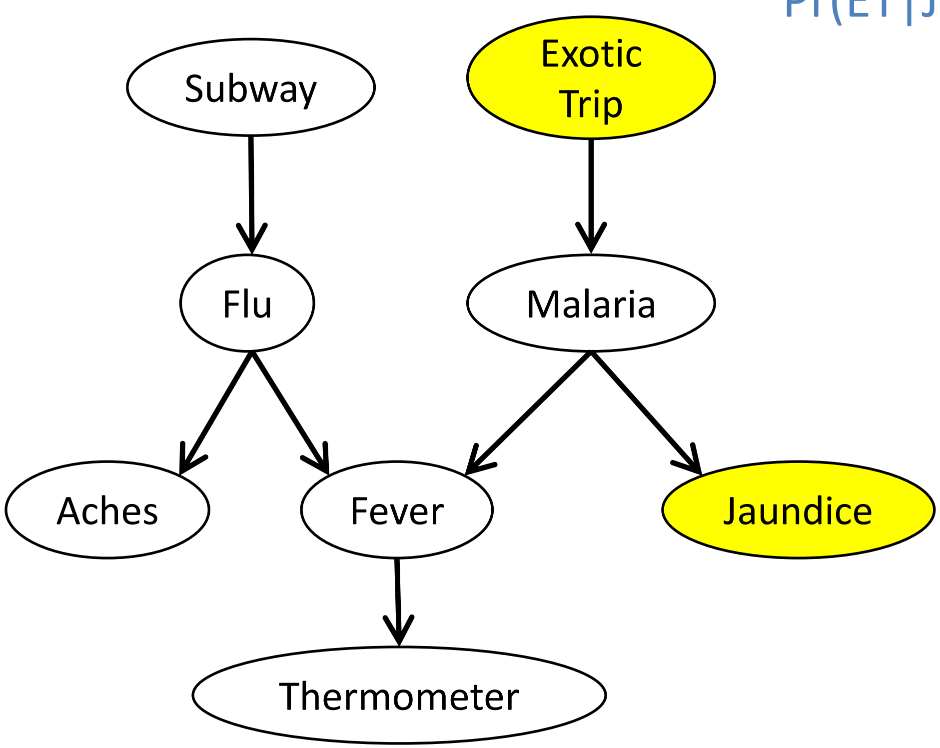
Pr(Fev|s,~m) = ?



Simple Backward Inference

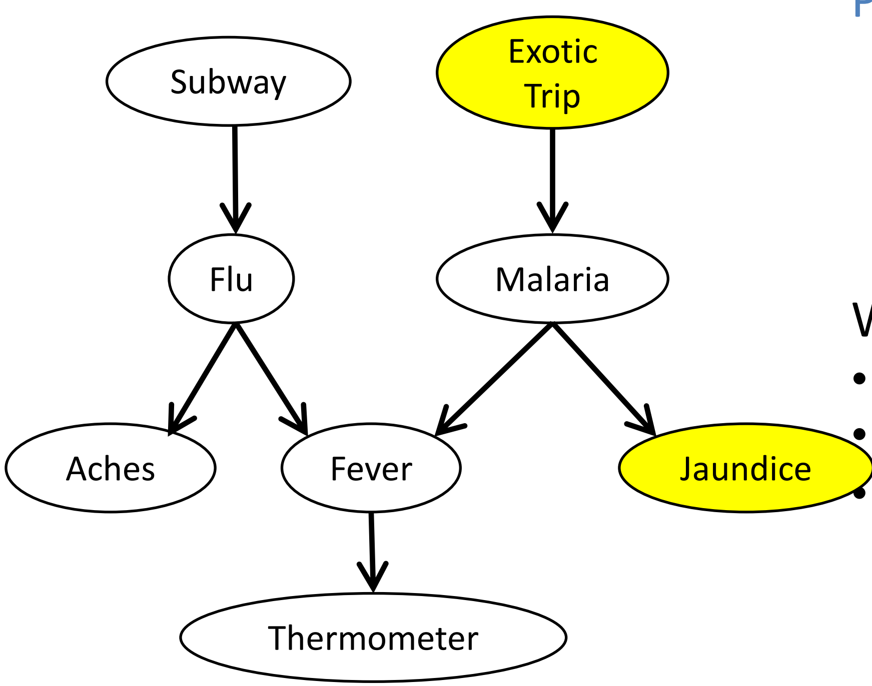
- When evidence is downstream of the query variable, we must reason "backwards"
  - Requires the use of Bayes rule

Pr(ET|J) = ?



Simple Backward Inference

- When evidence is downstream of the query variable, we must reason "backwards"
  - Requires the use of Bayes rule



$$\Pr(ET|j) = \alpha \Pr(j|ET)\Pr(ET)$$
$$= \alpha \textstyle\sum_M \Pr(j, M|ET) \Pr(ET)$$
$$= \alpha \textstyle\sum_M \Pr(j|M, ET)\Pr(M|ET) \Pr(ET)$$
$$= \alpha \textstyle\sum_M \Pr(j|M)\Pr(M|ET) \Pr(ET)$$

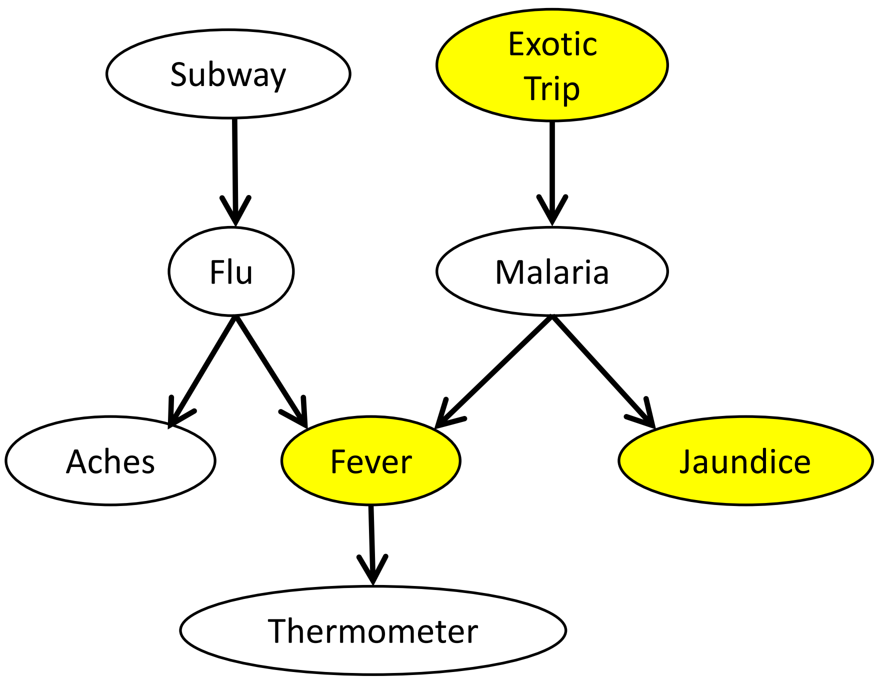
Where $\alpha = 1/\Pr(j)$

- Don't need to compute α
- Can compute Pr(ET|j) for each value of ET
- Add up terms Pr(j|ET)Pr(ET) for all values of ET (they sum up to Pr(j))

Simple Backward Inference

- Same idea applies when several pieces of evidence appear downstream

Pr(ET|j,fever) = ?



More Efficient Algorithms

- Variable elimination
  - Push sums in as far as possible
  - Amount of work is bounded by size of largest term

More Efficient Algorithms

- Variable elimination
  - Push sums in as far as possible
  - Amount of work is bounded by size of largest term

- Dynamic programming
  - Avoid redundant computation by computing several marginals at the same time
  - If BN is a tree, can use a local message passing algorithm (called forwards-backwards algorithm in HMMs)
  - Double counts if BN has undirected cycles

More Efficient Algorithms

- Variable elimination
  - Push sums in as far as possible
  - Amount of work is bounded by size of largest term
- Dynamic programming
  - Avoid redundant computation by computing several marginals at the same time
  - If BN is a tree, can use a local message passing algorithm (called forwards-backwards algorithm in HMMs)
  - Double counts if BN has undirected cycles
- Clique/Junction tree inference (one of the project options)
  - Converts BN into a junction tree (of clusters of nodes)
  - Then run local message passing algorithm on junction tree

More Efficient Algorithms

- Variable elimination
  - Push sums in as far as possible
  - Amount of work is bounded by size of largest term
- Dynamic programming
  - Avoid redundant computation by computing several marginals at the same time
  - If BN is a tree, can use a local message passing algorithm (called forwards-backwards algorithm in HMMs)
  - Double counts if BN has undirected cycles
- Clique/Junction tree inference (one of the project options)
  - Converts BN into a junction tree (of clusters of nodes)
  - Then run local message passing algorithm on junction tree
- Approximation algorithms (sampling, loopy belief propagation, etc.)

Key Ideas

- Main concept
  - Simple forward and backward inference involves basic probability manipulations and calculations
- Main tasks of interest:
  - Updating belief of $\mathbf{X}$: $\Pr(\mathbf{X}|\mathbf{e})$
  - Most probable explanation: $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{e})$
- Algorithm:
  - Most general exact inference algorithm is called clique inference or junction inference