# (Exact) Lifted inference

Luc De Raedt, Kristian Kersting,Sriraam Natarajan, David Poole

Belgium, Germany, USA, Canada

February 2017

# Outline

1. Exact Inference
   - Lifted Inference
   - Recursive Conditioning
   - Lifted Recursive Conditioning

# Why Exact Inference?

Why do we care about exact inference?

- Gold standard

# Why Exact Inference?

Why do we care about exact inference?

- Gold standard
- Size of problems amenable to exact inference is growing

# Why Exact Inference?

Why do we care about exact inference?

- Gold standard
- Size of problems amenable to exact inference is growing
- Learning for inference

# Why Exact Inference?

Why do we care about exact inference?

- Gold standard
- Size of problems amenable to exact inference is growing
- Learning for inference
- Basis for efficient approximate inference:
    - Rao-Blackwellization
    - Variational Methods

# Outline

1. Exact Inference
   - Lifted Inference
   - Recursive Conditioning
   - Lifted Recursive Conditioning

# Lifted Inference

- Idea: treat those individuals about which you have the same information as a block; just count them.
- Use the ideas from lifted theorem proving - no need to ground.
- Potential to be exponentially faster in the number of non-differentialed individuals.
- Relies on knowing the number of individuals (the population size).

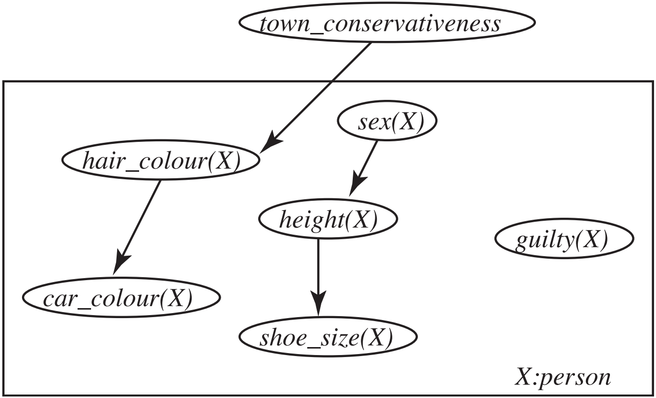
# Queries depend on population size

Suppose we observe:

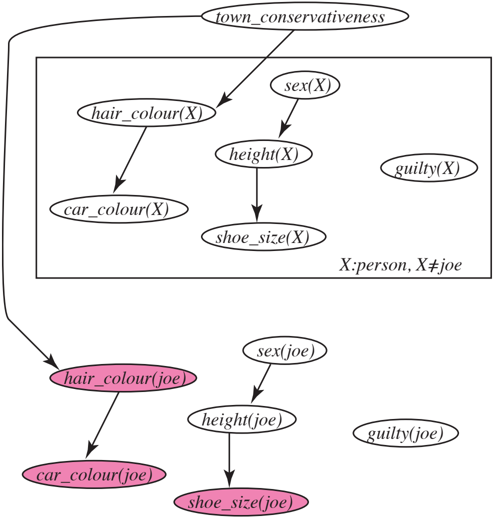
- Joe has purple hair, a purple car, and has big feet.
- A person with purple hair, a purple car, and who is very tall was seen committing a crime.

What is the probability that Joe is guilty?

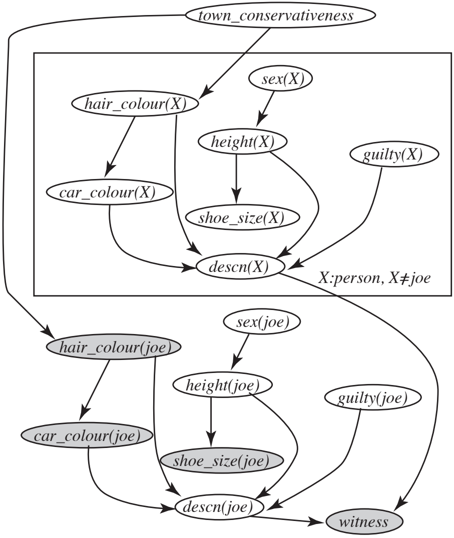
# Background parametrized belief network



# Observing information about Joe



# Observing Joe and the crime

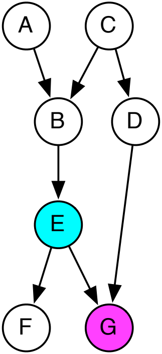


# Outline

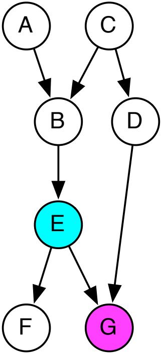
1. Exact Inference
   - Lifted Inference
   - Recursive Conditioning
   - Lifted Recursive Conditioning

# Inference via factorization in graphical models

$$P(E \mid g) \quad = \quad \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$



# Inference via factorization in graphical models



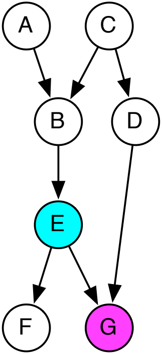

$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$=$$

# Inference via factorization in graphical models




$$P(E \mid g) \quad = \quad \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

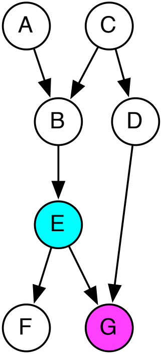
$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$=$$

$$\left( \sum_D P(D \mid C)P(g \mid ED) \right) \Bigg)$$

# Inference via factorization in graphical models



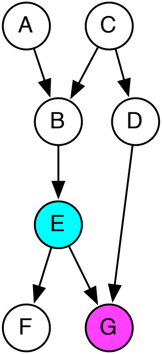

$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$= \left(\sum_A P(A)P(B \mid AC)\right) \left(\sum_D P(D \mid C)P(g \mid ED)\right)\Bigg)$$

# Inference via factorization in graphical models



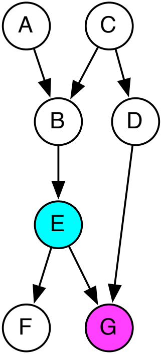

$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$= \sum_C \left( P(C) \left( \sum_A P(A)P(B \mid AC) \right) \left( \sum_D P(D \mid C)P(g \mid ED) \right) \right)$$

# Inference via factorization in graphical models



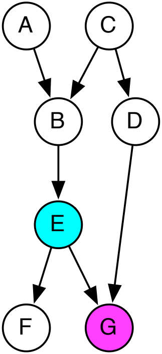

$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$= \sum_B P(E \mid B) \sum_C \left( P(C) \left( \sum_A P(A)P(B \mid AC) \right) \left( \sum_D P(D \mid C)P(g \mid ED) \right) \right)$$

# Inference via factorization in graphical models




$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$= \left( \sum_F P(F \mid E) \right) \sum_B P(E \mid B) \sum_C \left( P(C) \left( \sum_A P(A)P(B \mid AC) \right) \left( \sum_D P(D \mid C)P(g \mid ED) \right) \right)$$

# Recursive Conditioning

- Variable elimination is the dynamic programming variant of recursive conditioning.
- Recursive Conditioning is the search variant of variable elimination
- They do the same additions and multiplications.
- Complexity $O(nr^t)$, for $n$ variables, range size $r$, and treewidth $t$.

# Recursive Conditioning

procedure $rc(Con : \text{context}, \quad Fs : \text{set of factors})$:

if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$

return $v$

else if $vars(Con) \not\subseteq vars(Fs)$

return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$

else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$

return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$

else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$

return $rc(Con, Fs_1) \times rc(Con, Fs_2)$

else select variable $X \in vars(Fs)$

$sum \leftarrow 0$

for each $v \in domain(X)$

$sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$

$cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$

return $sum$

# Recursive Conditioning

procedure $rc(Con : \text{context}, \quad Fs : \text{set of factors})$:

if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$

return $v$

else if $vars(Con) \not\subseteq vars(Fs)$

return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$

else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$

return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$ Evaluate

else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$

return $rc(Con, Fs_1) \times rc(Con, Fs_2)$

else select variable $X \in vars(Fs)$ Branch

$sum \leftarrow 0$

for each $v \in domain(X)$

$sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$

$cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$

return $sum$

# Recursive Conditioning

procedure $rc(Con: \text{context}, \quad Fs: \text{set of factors})$:

  if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$ — Recall

    return $v$

  else if $vars(Con) \not\subseteq vars(Fs)$ — Forget

    return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$

  else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$

    return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$

  else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$

    return $rc(Con, Fs_1) \times rc(Con, Fs_2)$

  else select variable $X \in vars(Fs)$

    $sum \leftarrow 0$

    for each $v \in domain(X)$

      $sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$

    $cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$ — Remember

    return $sum$

# Recursive Conditioning

procedure $rc(Con : \text{context}, \quad Fs : \text{set of factors})$:
- if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$
  - return $v$
- else if $vars(Con) \not\subseteq vars(Fs)$
  - return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$
- else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$
  - return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$
- else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$
  - return $rc(Con, Fs_1) \times rc(Con, Fs_2)$ Disconnected
- else select variable $X \in vars(Fs)$
  - $sum \leftarrow 0$
  - for each $v \in domain(X)$
    - $sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$
  - $cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$
  - return $sum$

# Outline

1. Exact Inference
   - Lifted Inference
   - Recursive Conditioning
   - Lifted Recursive Conditioning

# Weighted Formula

A Weighted formula is a triple $\langle C, V, t\rangle$ where

- $C$ is a set of inequality constraints on parameters,
- $V$ a formula on parametrized random variables
- $t$ number

Example:

$\langle\{X \neq Y, Y \neq donald\}, likes(X, Y) \wedge rich(Y), 0.001\rangle$
$\langle\{X \neq donald\}, likes(X, X) \wedge rich(X), 0.1\rangle$

...

# Lifted Recursive Conditioning

$lrc(Con, Fs)$

- $Con$ is a set of assignments to random variables and counts to assignments of instances of relations. e.g.:

$$
\begin{aligned}
\{\neg a,\ & \#_X f(X) \wedge g(X) = 7,\\
& \#_X f(X) \wedge \neg g(X) = 5,\\
& \#_X \neg f(X) \wedge g(X) = 18,\\
& \#_X \neg f(X) \wedge \neg g(X) = 0\}
\end{aligned}
$$

- $Fs$ is a set of weighted formulae, e.g.,

$$
\begin{aligned}
\{ & \langle \{\}, \neg a \wedge \neg f(X) \wedge g(X), 0.1\rangle ,\\
& \langle \{\}, a \wedge \neg f(X) \wedge g(X), 0.2\rangle ,\\
& \langle \{\}, f(X) \wedge g(Y), 0.3\rangle ,\\
& \langle \{\}, f(X) \wedge h(X), 0.4\rangle \}
\end{aligned}
$$

# Evaluating Weighted Formulae

*Con*:

$$\begin{aligned}\{\neg a,\ & \#_X f(X) \wedge g(X) = 7, \\ & \#_X f(X) \wedge \neg g(X) = 5, \\ & \#_X \neg f(X) \wedge g(X) = 18, \\ & \#_X \neg f(X) \wedge \neg g(X) = 0\}\end{aligned}$$

*Fs*:

$$\begin{aligned}\{ & \langle \{\}, \neg a \wedge \neg f(X) \wedge g(X), 0.1 \rangle, \\ & \langle \{\}, a \wedge \neg f(X) \wedge g(X), 0.2 \rangle, \\ & \langle \{\}, f(X) \wedge g(Y), 0.3 \rangle, \\ & \langle \{\}, f(X) \wedge h(X), 0.4 \rangle \}\end{aligned}$$

$lrc(Con, Fs)$ returns:

# Evaluating Weighted Formulae

*Con*:

$$\begin{aligned}\{\neg a,\ & \#_X f(X) \wedge g(X) = 7, \\ & \#_X f(X) \wedge \neg g(X) = 5, \\ & \#_X \neg f(X) \wedge g(X) = 18, \\ & \#_X \neg f(X) \wedge \neg g(X) = 0\}\end{aligned}$$

*Fs*:

$$\begin{aligned}\{ & \langle \{\}, \neg a \wedge \neg f(X) \wedge g(X), 0.1 \rangle, \\ & \langle \{\}, a \wedge \neg f(X) \wedge g(X), 0.2 \rangle, \\ & \langle \{\}, f(X) \wedge g(Y), 0.3 \rangle, \\ & \langle \{\}, f(X) \wedge h(X), 0.4 \rangle \}\end{aligned}$$

$lrc(Con, Fs)$ returns:

$$0.1^{18} * 0.3^{12*25} * lrc(Con, \{\langle \{\}, f(X) \wedge h(X), 0.4 \rangle\})$$

# Branching

*Con*:

$$\{\neg a,\ \#_X f(X) \wedge g(X) = 7,$$
$$\#_X f(X) \wedge \neg g(X) = 5,$$
$$\#_X \neg f(X) \wedge g(X) = 18,$$
$$\#_X \neg f(X) \wedge \neg g(X) = 0\}$$

*Fs*:

$$\{\langle \{\}, f(X) \wedge h(X), 0.4 \rangle, \ldots\}$$

Branching on $H$ for the 7 "$X$" individuals s.th. $f(X) \wedge g(X)$:
$lrc(Con, Fs) =$

# Branching

*Con*:

$$\{\neg a,\ \#_X f(X) \wedge g(X) = 7, \\ \#_X f(X) \wedge \neg g(X) = 5, \\ \#_X \neg f(X) \wedge g(X) = 18, \\ \#_X \neg f(X) \wedge \neg g(X) = 0\}$$

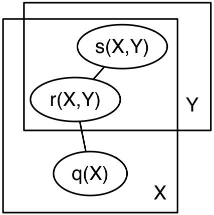
*Fs*:

$$\{\langle \{\}, f(X) \wedge h(X), 0.4\rangle, \ldots\}$$

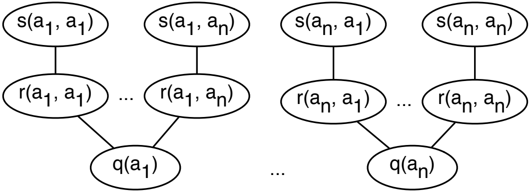
Branching on $H$ for the 7 "$X$" individuals s.th. $f(X) \wedge g(X)$:
$lrc(Con, Fs) =$

$$\sum_{i=0}^{7} \binom{7}{i} lrc(\{\neg a,\ \#_X f(X) \wedge g(X) \wedge h(X) = i, \\ \#_X f(X) \wedge g(X) \wedge \neg h(X) = 7 - i, \\ \#_X f(X) \wedge \neg g(X) = 5, \ldots\}, Fs)$$

# Recognizing Disconnectedness




Relational Model



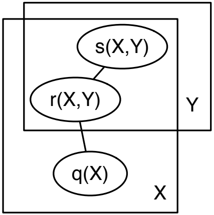

Grounding

Weighted formulae $Fs$:

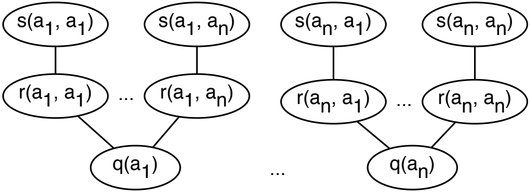
$$\{\langle\{\},\{s(X,Y)\wedge r(X,Y)\},t_1\rangle$$
$$\langle\{\},\{q(X)\wedge r(X,Y)\},t_2\rangle\}$$

$lrc(Con, Fs) =$

# Recognizing Disconnectedness




Relational Model




Grounding

Weighted formulae $Fs$:

$$\{\,\langle\{\},\{s(X,Y)\wedge r(X,Y)\},t_1\rangle$$
$$\langle\{\},\{q(X)\wedge r(X,Y)\},t_2\rangle\}$$

$lrc(Con, Fs) = lrc(Con, Fs\{X/c\})^n$
...now we only have unary predicates

# Simplifying Formulae

- So far, weighted formulae are not modified, only evaluated.
- Idea: branching creates new types (with disjoint populations)

  Example:

  Consider a context where 7 "X" individuals have $f(X) \wedge g(X)$,
  For each $i$ in $[0, \ldots, 7]$ create variables:
  - $X_0$ with population $7 - i$, all where $f(X) \wedge g(X) \wedge \neg h(X)$
  - $X_1$ with population $i$, all where $f(X) \wedge g(X) \wedge h(X)$
    these populations are disjoint

# Simplifying Formulae

- So far, weighted formulae are not modified, only evaluated.
- Idea: branching creates new types (with disjoint populations)
  Example:
  Consider a context where 7 "X" individuals have $f(X) \wedge g(X)$,
  For each $i$ in $[0, \ldots, 7]$ create variables:
  - $X_0$ with population $7 - i$, all where $f(X) \wedge g(X) \wedge \neg h(X)$
  - $X_1$ with population $i$, all where $f(X) \wedge g(X) \wedge h(X)$
    these populations are disjoint
- Can evaluate, and simplify weighted clauses for each population:
  - $\langle \{\}, f(X_0) \wedge h(X_0) \wedge m(X_0), 0.2 \rangle \longrightarrow$ removed
  - $\langle \{\}, f(X_0) \wedge \neg h(X_0), 0.2 \rangle \longrightarrow$ evaluates to $0.2^{7-i}$.
  - $\langle \{\}, f(X_1) \wedge h(X_1) \wedge m(X_1), 0.2 \rangle \longrightarrow \langle \{\}, m(X_1), 0.2 \rangle$

# Observations and Queries

- Observations become the initial context.
  Observations can be ground or lifted.
- $P(q|obs) = rc(q \wedge obs, Fs)/(rc(q \wedge obs, Fs) + rc(\neg q \wedge obs, Fs))$
  calls can share the cache
- "How many?" queries are also allowed

# Complexity

As the population size $n$ of undifferentiated individuals increases:

- If grounding is polynomial — instances must be disconnected — lifted inference is constant in $n$ (taking $r^n$ for real $r$)
- Otherwise, for unary relations, grounding is exponential and lifted inference is polynomial.
- If non-unary relations become unary, above holds.
- Otherwise, ground an argument.
  Always exponentially better than grounding everything.

# What we can and cannot lift

We can lift a model that consists just of

$$\langle \{\}, \{f(X) \wedge g(Z)\}, \alpha_4 \rangle$$

# What we can and cannot lift

We can lift a model that consists just of

$$\langle \{\}, \{f(X) \wedge g(Z)\}, \alpha_4 \rangle$$

or just of

$$\langle \{\}, \{f(X, Z) \wedge g(Y, Z)\}, \alpha_2 \rangle$$

# What we can and cannot lift

We can lift a model that consists just of

$$\langle \{\}, \{f(X) \wedge g(Z)\}, \alpha_4 \rangle$$

or just of

$$\langle \{\}, \{f(X, Z) \wedge g(Y, Z)\}, \alpha_2 \rangle$$

or just of

$$\langle \{\}, \{f(X, Z) \wedge g(Y, Z) \wedge h(Y)\}, \alpha_3 \rangle$$

# What we can and cannot lift

We can lift a model that consists just of

$$\langle \{\}, \{f(X) \wedge g(Z)\}, \alpha_4 \rangle$$

or just of

$$\langle \{\}, \{f(X, Z) \wedge g(Y, Z)\}, \alpha_2 \rangle$$

or just of

$$\langle \{\}, \{f(X, Z) \wedge g(Y, Z) \wedge h(Y)\}, \alpha_3 \rangle$$

We cannot lift (still exponential) a model that consists just of:

$$\langle \{\}, \{f(X, Z) \wedge g(Y, Z) \wedge h(Y, W)\}, \alpha_3 \rangle$$

# What we can and cannot lift

We can lift a model that consists just of

$$\langle\{\}, \{f(X) \wedge g(Z)\}, \alpha_4\rangle$$

or just of

$$\langle\{\}, \{f(X, Z) \wedge g(Y, Z)\}, \alpha_2\rangle$$

or just of

$$\langle\{\}, \{f(X, Z) \wedge g(Y, Z) \wedge h(Y)\}, \alpha_3\rangle$$

We cannot lift (still exponential) a model that consists just of:

$$\langle\{\}, \{f(X, Z) \wedge g(Y, Z) \wedge h(Y, W)\}, \alpha_3\rangle$$

or

$$\langle\{\}, \{f(X, Z) \wedge g(Y, Z) \wedge h(Y, X)\}, \alpha_3\rangle$$

# Compilation

- The computation reduces to products and sums
- The structure can be determined at compile time
- Orders of magnitude faster than lifted recursive conditioning