Lifted Inference: Exact Search Based Algorithms

Vibhav Gogate

The University of Texas at Dallas

Overview

- Background and Notation
  - Probabilistic Knowledge Bases
- Exact Inference in Propositional Models
- First-order Normal Forms
- Exact Lifted Inference using Normal Forms
- Exact Lifted Inference in General Models

Probabilistic Knowledge Bases (PKB): Propositional

- A set of propositional variables $\mathbf{X} = \{X_1, \ldots, X_n\}$
- A set of weighted formulas $\{(f_1, w_1), \ldots, (f_m, w_m)\}$
  Important: We will assume that each formula is a disjunction of literals, namely a clause. A literal is an atom or its negation (e.g., X, $\neg X$, etc.). Note that you can convert any conjunctive formula (namely a conjunction of literals) to a clause by negating the formula (which yields a clause) as well as its weight.
  In this talk, assume $-\infty < w_i < \infty$ for all i
- Distribution represented by a PKB:

$$\Pr(\overline{\mathbf{x}}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{m} w_i N(f_i, \overline{\mathbf{x}})\right) = \frac{1}{Z} \prod_{i=1}^{m} \exp\left(w_i N(f_i, \overline{\mathbf{x}})\right)$$

  where $\overline{\mathbf{x}}$ is a truth assignment to all variables in $\mathbf{X}$ and $N(f_i, \overline{\mathbf{x}}) = 1$ if $\overline{\mathbf{x}}$ satisfies f_i and 0 otherwise.

Graphical Models To PKBs

- **Why PKBs?**
  - We can encode any graphical model as a PKB
  - Most learning algorithms induce log-linear models instead of inducing a Markov network. Log-linear models can be easily translated to PKBs.
- **Encoding: Graphical model to PKBs**
  - Convert each potential/CPT entry to a weighted formula
    - Formula = Conjunction of propositions.
    - Weight = Log of the potential/CPT value
  - **Example:** $[(X_1 = 0, X_2 = 1), v]$ to $[\neg X_1 \wedge X_2, \ln(v)]$.
- We can take advantage of identical potential values to reduce the size of the representation.

- $[X_1 \wedge X_2, \ln(v)]$
- $[\neg X_1 \wedge X_2, \ln(v)]$
- $[X_1 \wedge \neg X_2, \ln(v)]$

is equivalent to
$========>$

- $X_1 \vee X_2, \ln(v)$.

PKB and its Distribution: Example

- **PKB**: Two weighted formulas
  1. $f_1 = A \vee B$, 5
  2. $f_2 = \neg B \vee C$, 10
- **Example assignment:** $\bar{\mathbf{x}} = (A = 0, B = 1, C = 0)$
  - Assignment evaluates the first formula to true. Therefore, $N(f_1, \bar{\mathbf{x}}) = 1$
  - Assignment evaluates the second formula to false. Therefore, $N(f_1, \bar{\mathbf{x}}) = 0$

$$\Pr(A = 0, B = 1, C = 0) = \frac{1}{Z}\exp(5 \times 1 + 10 \times 0) = \frac{1}{Z}\exp(5)$$

- Important Inference query:

$$Compute: \; Z = \sum_{\overline{x}} w(\overline{x})$$

Inference in PKBs: Computing the Partition Function

- **Most Basic Method:** Inference by Conditioning with "true and false clause deletion."
- Assume that the PKB contains a formula which is always true with weight 0. We do this for making the operations simpler (details later).
- Algorithm Z (Input: PKB K)
  - **Base case: if** K is defined over n variables and contains only the "always true" formula having weight w, **then** $Z(K) = \exp(w) \times 2^n$
  - **Key Step:** Condition on a proposition X_i (chosen heuristically).

$$Z(K) = Z(K_{|X_i=0}) + Z(K_{|X_i=1})$$

Inference in PKBs: Conditioning on a literal L_i

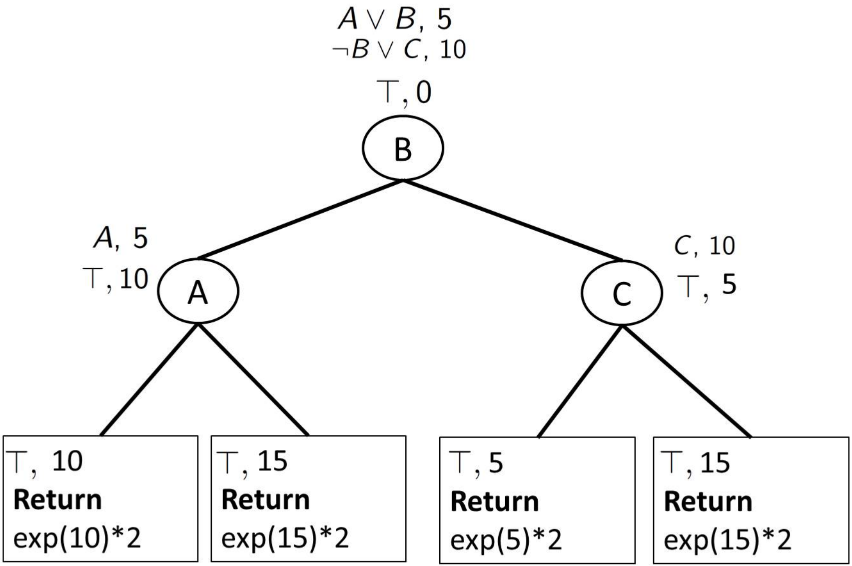
Given a literal L_i, the PKB obtained by conditioning on L_i can be computed as follows:

- Input: A PKB K and a literal L_i
- Output: A PKB $K_{|L_i}$ conditioned on L_i
  1. Let w be the weight attached to the all true formula
  2. **foreach** weighted formula (f_j, w_j) in K
     - 2.1 **if** L_i appears in f_j, **then** remove (f_j, w_j) from the K and update $w = w + w_j$
     - 2.2 **if** $\neg L_i$ appears in f_j, **then** remove $\neg L_i$ from f_j
     - 2.3 **if** f_j is empty, **then** remove (f_j, w_j) from K
  3. Remove the variable of L_i from K
  4. **return** K

Example: Conditioning on a literal L_i

- **PKB**: 3 variables $\{A, B, C\}$ and 3 weighted formulas
  1. $f_1 = A \vee B$, 5
  2. $f_2 = \neg B \vee C$, 10
  3. $\top, 0$ ("Always true" formula with weight 0)
- Condition on $\neg B$ (i.e., B is false)
  - The first weighted formula becomes $A, 5$ since we remove B from the formula.
  - The second formula is removed since it evaluates to true and its weight is added to $\top$.
  - **New PKB**: 2 variables $\{A, C\}$ and 2 weighted formulas
    1. $f_3 = A$, 5
    2. $\top, 10$

Full Example: Search Tree



First-Order Logic: Notation

- Predicate symbols: Relationship between objects.
  Example: $\texttt{Friends}(x, y)$
- Quantifiers: $\forall$, $\exists$ (we will ignore $\exists$)
- Constants: Objects in the domain (denoted by upper case letters such as X, Y and Z)
- Term : A term is a constant or a logical variable. Logical variables are denoted by lower case letters such as x, y and z.
- Formulas are constructed recursively as follows:
  - If $\texttt{R}$ is a predicate symbol having arity k and $t_1, \ldots, t_k$ are terms then $\texttt{R}(t_1, \ldots, t_k)$ is a formula.
  - A negation of a formula is a formula.
  - If f and g are formulas, then applying any binary logical operation, e.g., $\vee$, $\wedge$, etc. to them yields a formula
  - If x is a variable in a formula f then $\forall x\ f$ and $\exists x\ f$ are formulas. (assume that every variable is quantified).

We are assuming a strict subset of first-order logic called Herbrand Logic that does not have the equality symbol, infinite domains and function symbols.

Markov Logic Networks (MLNs)

- First-order logic formulas with weights.
- Template for generating Markov networks. Given a set of constants that model objects in the domain, a MLN yields the following Markov network (log-linear model)
  - Create one feature for each grounding of each formula in the Markov network. The weight of the feature is the weight attached to the corresponding first-order formula.
  - Create one random variable for each grounding of each predicate in the Markov network
- Distribution: Given a MLN $\{f_i, w_i\}_{i=1}^m$

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_{i=1}^{m} w_i N(f_i, \omega)\right)$$

where ω is a possible world, namely a truth assignment to all random variables in the Markov network and $N(f_i, \omega)$ is the number of groundings of f_i that evaluate to true given ω.

Markov Logic: Example

- People having Asthma are generally not friends with smokers.

$$\forall\, x, y\ \mathtt{A}(x) \wedge \mathtt{F}(x, y) \Rightarrow \neg \mathtt{S}(y), w$$

- Given two individuals B and C, we get the following Markov network:
  - $\mathtt{A}(B) \wedge \mathtt{F}(B, B) \Rightarrow \neg \mathtt{S}(B), w$
  - $\mathtt{A}(B) \wedge \mathtt{F}(B, C) \Rightarrow \neg \mathtt{S}(C), w$
  - $\mathtt{A}(C) \wedge \mathtt{F}(C, B) \Rightarrow \neg \mathtt{S}(B), w$
  - $\mathtt{A}(C) \wedge \mathtt{F}(C, C) \Rightarrow \neg \mathtt{S}(C), w$
- Great news! We have propositional formulas with weights. We can use the same inference scheme as before.
- Bad news! Imagine a social network over a Billion individuals.

Normal PKBs

Our lifted conditioning algorithm which we will describe next operates on the so-called **normal PKBs** defined below.

- There are no constants in any formulas.
- Each formula is a clause and all its logical variables are universally quantified
- The domains of the logical variables are finite
- There are no self joins (namely the same predicate symbol does not appear twice in a clause)
- The domains of the logical variables belonging to each equivalence class defined below are the same.
  - Binding Equivalence classes: Let $\mathbf{x}$ be the set of all logical variables in the PKB. Let $\sim$ be a binary relation, which we call the binding relation. We say that $x \sim y$ if (a) they appear as the same argument of a predicate R in a formula in the PKB; or (b) there exists a logical variable z such that $x \sim z$ and $y \sim z$. The binding equivalence classes are the equivalence classes of $\mathbf{x}$ due to the binding relation.

Examples of Normal PKBs and Binding Classes

We are assuming that all clauses are standardized apart. Namely, the logical variables are unique to each clause.

Not a normal PKB:
$\forall x\ \mathtt{R}(x) \vee \mathtt{S}(y)$
$\forall a, b\ \mathtt{S}(a) \vee \mathtt{S}(b)$
Variable y is free
Self-join on $\mathtt{S}$

Not a normal PKB:
$\forall x\ \mathtt{R}(x) \vee \mathtt{S}(C)$
$\forall x, y\ \mathtt{S}(y) \vee \mathtt{T}(z)$
constant in the domain of $\mathtt{S}$

Binding classes example:
$\forall x\ \mathtt{R}(x) \vee \mathtt{S}(x)$
$\forall y, z\ \mathtt{R}(y) \vee \mathtt{T}(z)$
$\forall i\ \mathtt{S}(i) \vee \mathtt{T}(i)$
$\forall a, b\ \mathtt{F}(a, b) \vee \mathtt{T}(b)$
Two Equivalence classes:
$\{x, y, i, z, b\}$ and $\{a\}$.

Normal PKB:
$\forall x\ \mathtt{R}(x) \vee \mathtt{S}(x)$
$\forall y, z\ \mathtt{R}(y) \vee \mathtt{S}(z)$

Efficient Inference using Lifting

- **Basic Idea:** Exploit symmetries in the first-order representation. Instead of conditioning on a proposition, condition on a first-order atom.
- But this is still exponential because conditioning on a first-order atom means that we are conditioning on all possible truth assignments to all groundings of the first-order atom.
- **Example:** If $\mathtt{R}(x)$ is an atom and $A_1, A_2, \ldots, A_n$ are constants then conditioning on $\mathtt{R}(x)$ means that we are conditioning on the 2^n truth assignments to the propositions $\mathtt{R}(A_1)$, ..., $\mathtt{R}(A_n)$.
- **Lifted Conditioning:** Condition on the number of true groundings of a first-order atom.
  Assumption: The sub-problems induced are the same.

Lifted Conditioning: Example

- PKB
  - $\forall\, x, y\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, y) \vee \neg \mathtt{S}(y), w$
  - $\top, 0$
- Assuming that there are n constants in the domain of y, we will condition on the following $n+1$ assignments:
  - i groundings of $\mathtt{S}(y)$ are true and the remaining are false.
- Why this works for $\mathtt{S}(y)$? Fix $i = 3$ and assume that we have 5 constants $\{B_1, \ldots, B_5\}$.

$\mathtt{S}(B_1)$, $\mathtt{S}(B_2)$ and $\mathtt{S}(B_3)$ are true	$\mathtt{S}(B_3)$, $\mathtt{S}(B_4)$ and $\mathtt{S}(B_5)$ are true
$\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_1) \vee \neg \mathtt{S}(B_1)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_2) \vee \neg \mathtt{S}(B_2)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_3) \vee \neg \mathtt{S}(B_3)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_4) \vee \neg \mathtt{S}(B_4)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_5) \vee \neg \mathtt{S}(B_5)$	$\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_1) \vee \neg \mathtt{S}(B_1)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_2) \vee \neg \mathtt{S}(B_2)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_3) \vee \neg \mathtt{S}(B_3)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_4) \vee \neg \mathtt{S}(B_4)$ $\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, B_5) \vee \neg \mathtt{S}(B_5)$
$[\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, y'), w]$ $[\top, 10w]$ Domain of y' is $\{B_1, B_2, B_3\}$	$[\forall x\ \neg \mathtt{A}(x) \vee \neg \mathtt{F}(x, y''), w]$, $[\top, 10w]$ Domain of y'' is $\{B_3, B_4, B_5\}$

(Red formulas are removed because they are true, red literals are removed from the formulas because they are false).

The two PKBs are equivalent subject to renaming of constants.

Lifted Conditioning: Example

- Does lifted conditioning work for $\mathtt{F}(x, y)$?
- Let us fix $i = 2$ (which means that 2 groundings of $\mathtt{F}(x, y)$ are true) and assume 3 constants: $\{B_1, B_2, B_3\}$.

$\mathtt{F}(B_1, B_1)$, $\mathtt{F}(B_1, B_2)$ are true	$\mathtt{F}(B_2, B_3)$, $\mathtt{F}(B_3, B_2)$ are true
$\neg A(B_1) \vee \neg F(B_1, B_1) \vee \neg S(B_1)$	$\neg A(B_1) \vee \neg F(B_1, B_1) \vee \neg S(B_1)$
$\neg A(B_1) \vee \neg F(B_1, B_2) \vee \neg S(B_2)$	$\neg A(B_1) \vee \neg F(B_1, B_2) \vee \neg S(B_2)$
$\neg A(B_1) \vee \neg F(B_1, B_3) \vee \neg S(B_3)$	$\neg A(B_1) \vee \neg F(B_1, B_3) \vee \neg S(B_3)$
$\neg A(B_2) \vee \neg F(B_2, B_1) \vee \neg S(B_1)$	$\neg A(B_2) \vee \neg F(B_2, B_1) \vee \neg S(B_1)$
$\neg A(B_2) \vee \neg F(B_2, B_2) \vee \neg S(B_2)$	$\neg A(B_2) \vee \neg F(B_2, B_2) \vee \neg S(B_2)$
$\neg A(B_2) \vee \neg F(B_2, B_3) \vee \neg S(B_3)$	$\neg A(B_2) \vee \neg F(B_2, B_3) \vee \neg S(B_3)$
$\neg A(B_3) \vee \neg F(B_3, B_1) \vee \neg S(B_1)$	$\neg A(B_3) \vee \neg F(B_3, B_1) \vee \neg S(B_1)$
$\neg A(B_3) \vee \neg F(B_3, B_2) \vee \neg S(B_2)$	$\neg A(B_3) \vee \neg F(B_3, B_2) \vee \neg S(B_2)$
$\neg A(B_3) \vee \neg F(B_3, B_3) \vee \neg S(B_3)$	$\neg A(B_3) \vee \neg F(B_3, B_3) \vee \neg S(B_3)$
$[\forall y\ \neg \mathtt{A}(B_1) \vee \neg \mathtt{S}(y'), w]$ $[\top, 7w]$ Domain of y' is $\{B_1, B_2\}$	$[\neg \mathtt{A}(B_2) \vee \neg \mathtt{S}(B_3), w]$, $[\top, 7w]$ $[\neg \mathtt{A}(B_3) \vee \neg \mathtt{S}(B_2), w]$

The two PKBs are not equivalent.

Lifted Conditioning

- Turns out that lifted conditioning always works if you apply it to singleton atoms (atoms which have only one quantified variable)
- **General rule:** Given a PKB K, a singleton atom $\mathtt{R}$ and n constants, the partition function of the PKB can be written as:

$$Z(K) = \sum_{i=0}^{n} \binom{n}{i} Z(K_{|\mathtt{R}=i})$$

- $\binom{n}{i}$ gives the number of assignments in which i groundings are true and the remaining $n - i$ groundings are false.
- How to compute $K_{|\mathtt{R}=i}$?

Lifted Conditioning Operations: Example

$$\forall z\ \mathtt{R}(z) \vee \mathtt{S}(z), w_1$$
$$\forall x, y\ \mathtt{R}(x) \vee \mathtt{S}(y), w_2$$
$$\top, 0$$

n constants in the domain.

- Split x, y, z and all predicates that mention them into two, one having domain size of i and another having domain size of $n - i$. Then remove the formulas that evaluate to true, adding their weight to $\top$ and remove the false atoms from all formulas.

T

After Splitting:

$$\forall z_1\ \mathtt{R}_1(z_1) \vee \mathtt{S}_1(z_1), w_1$$
$$\forall z_2\ \mathtt{R}_2(z_2) \vee \mathtt{S}_2(z_2), w_1$$
$$\forall x_1, y_1\ \mathtt{R}_1(x_1) \vee \mathtt{S}_1(y_1), w_2$$
$$\forall x_1, y_2\ \mathtt{R}_1(x_1) \vee \mathtt{S}_2(y_2), w_2$$
$$\forall x_2, y_1\ \mathtt{R}_2(x_2) \vee \mathtt{S}_1(y_1), w_2$$
$$\forall x_2, y_2\ \mathtt{R}_2(x_2) \vee \mathtt{S}_2(y_2), w_2$$
$$\top, 0$$

After removing true formulas and false atoms:

$$\forall z_2\ S_2(z_2), w_1$$
$$\forall y_1\ S_1(y_1), iw_2$$
$$\forall y_2\ S_2(y_2), (n-i)w_2$$
$$\top, iw_1 + i^2 w_2 + i(n-i)w_2$$

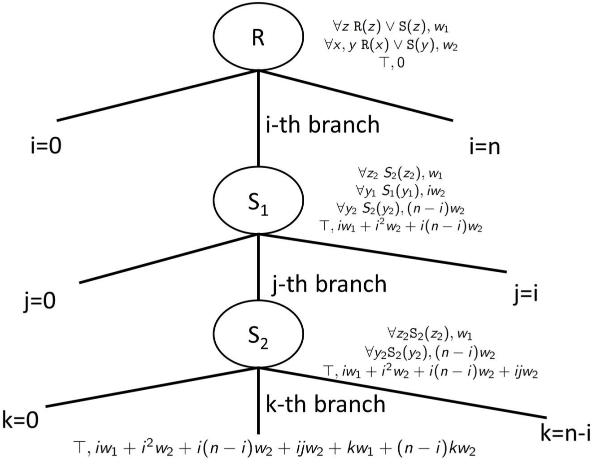
i constants in the domain of x_1, y_1 and $n - i$ constants in the domain of x_2, y_2.

Lifted Conditioning: Formal algorithm

Given a normal PKB K, a singleton atom $\mathtt{R}$ and n constants, $K_{|\mathtt{R}=i}$ can be computed as follows:

- Split all logical variables and corresponding predicates into two. Rewrite the formulas using the new predicates and logical variables.
- Delete all satisfied formulas, updating the weight of $\top$ accordingly
- Remove all literals that evaluate to false from each formula. Update the weight of the formula appropriately. If the formula becomes empty, delete the formula from K.

Lifted Search Tree: Example



Lifted AND/OR Search

- The Conditioning method explores the OR search tree. We can reduce the complexity of the search procedure by exploiting problem decomposition (aka AND nodes)
- Decomposition in Propositional Search
  - If the formulas in the PKB K can be partitioned into two or more sets $\{K_1, \ldots, K_p\}$ such that no two formulas that appear in different partitions share an atom, then the partition function of the PKB can be computed as follows:

$$Z(K) = \prod_{i=1}^{p} Z(K_i)$$

- Same applies to first-order PKBs. However, we can do better.

Lifted Decomposition

- Example PKB: $[\forall x \ \mathtt{R}(x) \vee \mathtt{S}(x), w]$ $[\top, 0]$
- Assume that the constants are $\{B_1, \ldots, B_5\}$.

5-way partition of the ground PKB

1	$\mathtt{R}(B_1) \vee \mathtt{S}(B_1),\ w$	$Z_1 = 3\exp(w) + 1$
2	$\mathtt{R}(B_2) \vee \mathtt{S}(B_2),\ w$	$Z_2 = 3\exp(w) + 1$
3	$\mathtt{R}(B_3) \vee \mathtt{S}(B_3),\ w$	$Z_3 = 3\exp(w) + 1$
4	$\mathtt{R}(B_4) \vee \mathtt{S}(B_4),\ w$	$Z_4 = 3\exp(w) + 1$
5	$\mathtt{R}(B_5) \vee \mathtt{S}(B_5),\ w$	$Z_5 = 3\exp(w) + 1$

$$Z = \prod_{i=1}^{5} Z_i = (3\exp(w) + 1) \times (3\exp(w) + 1) \ldots 5 times = (3\exp(w) + 1)^5$$

All partitions are identical subject to renaming of variables.

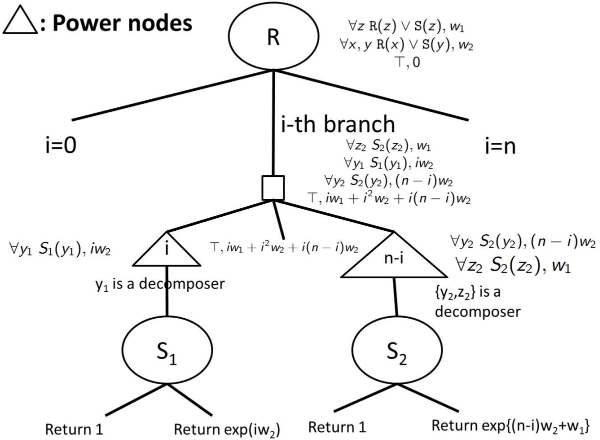
- **Alternate method:** Compute the partition function of one of the partitions and raise it to the power of the number of partitions.

Lifted Decomposition

- Possible when all variables in a binding equivalence class obey the following property.
  - Each variable in the class appears in all atoms of a formula.
- We call the binding class obeying the above property as a decomposer.
- **Theorem** Let **x** be a decomposer in a PKB, *n* be the number of constants and let PKB' be obtained from PKB by setting the domain size of all logical variables in the decomposer to one, then

$$Z(PKB) = [Z(PKB')]^n$$

Lifted AND/OR Search Tree Example



Formal Algorithm: LSD aka Lifted Search and Decomposition

Input: A PKB K
Output: $Z(K)$

- Simplify(K)
- **Base case: if** K contains only the all true clause with weight w **then return** $\exp(w) \times 2^{|Vars(K)|}$
- **AND Decomposition: if** there exists a partition $K_1, K_2, \ldots, K_p$ of K such that no two K_i's share any atoms **then return** $\prod_{i=1}^{p} LSD(K_i)$
- **Lifted Decomposition: if** there is a *decomposer* **x** having domain size n, **then** set the domain size of all members of **x** to 1 and **return** $[LSD(K|Dom(\mathbf{x}) = 1)]^n$.
- **Lifted Conditioning: if** there is a singleton atom `R` (heuristically selected) in K **then return** $\sum_{i=0}^{n} \binom{n}{i} LSD(K_{|R=i})$
- **Partially grounding:** Ground all variables in a binding class yielding a new PKB K'. **return** $LSD(K')$

Complexity of LSD

- Run the algorithm schematically. Namely, at each conditioning step, instead of exploring all the *n* branches, just branch on the i-th branch.
- Complexity: Exponential in the minimum depth of all the schematic trees constructed this way (which is same as the pseudo tree)
  - We can include the branching factor and come up with a precise expression for the complexity given an ordering heuristic.
  - If we use caching then the complexity is the minimum **lifted context size**, which gives us a notion of **lifted treewidth**.

Generalizing the Algorithm: Extensions

- Until now: No evidence and restrictions on the PKB!
- Unlike traditional propositional inference in which evidence reduces the complexity of exact inference, evidence increases the complexity of lifted inference because it breaks symmetries.
  - **Method 1:** Add evidence and rewrite the PKB in normal form.
  - **Method 2:** Add constraints. Think of a weighted formula as a triplet (formula,weight,constraints on logical variables)
    - Example: $\forall x, y\ \mathtt{R}(x) \vee \mathtt{S}(x, y) \vee \mathtt{T}(y), w$. Evidence: odd $\mathtt{R}$'s and $\mathtt{T}$'s are true and the remaining are unknown.
    - This is the same as the following two constrained formulas: $\forall x, y\ \mathtt{R}(x) \vee \mathtt{S}(x, y) \vee \mathtt{T}(y), w$ such that $x \neq odd \wedge y \neq odd$ $\top, \#(x = odd \vee y = odd)w$.
- **Extensions** Handling Self-joins using constraints or grounding.

Tomorrow

- Sampling-Based Inference: Gibbs and Importance
- MAP Inference