Symmetric Weighted First-Order Model Counting and Factorial-Like Functions

Paulius Dilkas

WFOMC: State of the Art

I'm excluding techniques that are restricted to two variables and purely theoretical results.

1. S. M. Kazemi et al. "New Liftable Classes for First-Order Probabilistic Inference". In: *NIPS*. 2016
   - generic domain recursion (implementation unavailable)
2. Forclift: G. Van den Broeck et al. "Lifted Probabilistic Inference by First-Order Knowledge Compilation". In: *IJCAI*. 2011
   - somewhat restrictive but well-developed
3. L2C: S. M. Kazemi and David Poole. "Knowledge Compilation for Lifted Probabilistic Inference: Compiling to a Low-Level Language". In: *KR*. 2016
   - very basic
4. Alchemy: P. M. Domingos et al. "Unifying Logical and Statistical AI". In: *AAAI*. 2006
   - old, mostly focused on approximations

Counting (Unweighted) Functions: Currently Unliftable (1)

Injections

Theory:

$$\forall x \in M. \forall y, z \in N. P(x,y) \wedge P(x,z) \Rightarrow y = z$$
$$\forall x \in M. \exists y \in N. P(x,y)$$
$$\forall w, x \in M. \forall y \in N. P(w,y) \wedge P(x,y) \Rightarrow w = x$$

Answer: $n^{\underline{m}} = n \cdot (n-1) \cdots (n-m+1)$ if $m \leq n$ and 0 otherwise (for positive m and n).

Counting (Unweighted) Functions: Currently Unliftable (1)

Injections

Theory:

$$\forall x \in M . \forall y, z \in N . P(x, y) \wedge P(x, z) \Rightarrow y = z$$
$$\forall x \in M . \exists y \in N . P(x, y)$$
$$\forall w, x \in M . \forall y \in N . P(w, y) \wedge P(x, y) \Rightarrow w = x$$

Answer: $n^{\underline{m}} = n \cdot (n-1) \cdots (n-m+1)$ if $m \leq n$ and 0 otherwise (for positive m and n).

Answer found by Forclift:

$$f(m, n) = \sum_{k=0}^{m} \binom{m}{k} (-1)^{m-k} g(k, n),$$

where

$$g(k, n) = \begin{cases} 1 & \text{if } k = 0 \\ \sum_{l=0}^{n} [l < 2] g(k-1, n-l) & \text{otherwise.} \end{cases}$$

(exponential...)

Counting (Unweighted) Functions: Currently Unliftable (2)

Surjections

Theory:

$$\forall x \in M. \forall y, z \in N. P(x, y) \wedge P(x, z) \Rightarrow y = z$$
$$\forall x \in M. \exists y \in N. P(x, y)$$
$$\forall y \in N. \exists x \in M. P(x, y)$$

Answer: $n! \left\{ {m \atop n} \right\} = \sum_{i=0}^{n} (-1)^i \binom{n}{i} (n-i)^m$.

Bijections

Theory:

$$\forall x \in M. \forall y, z \in N. P(x, y) \wedge P(x, z) \Rightarrow y = z$$
$$\forall x \in M. \exists y \in N. P(x, y)$$
$$\forall w, x \in M. \forall y \in N. P(w, y) \wedge P(x, y) \Rightarrow w = x$$
$$\forall y \in N. \exists x \in M. P(x, y)$$

Answer: $n!$ if $m = n$.

Counting (Unweighted) Functions: Currently Unliftable (3)

Partial functions

Theory: $\forall x \in M. \forall y, z \in N. P(x, y) \wedge P(x, z) \Rightarrow y = z$
Answer: $(n+1)^m$.

Partial injections

Theory:

$$\forall x \in M. \forall y, z \in N. P(x, y) \wedge P(x, z) \Rightarrow y = z$$
$$\forall w, x \in M. \forall y \in N. P(w, y) \wedge P(x, y) \Rightarrow w = x$$

My answer: $\sum_{k=0}^{\min\{m,n\}} k! \binom{m}{k} \binom{n}{k}$.

Counting (Unweighted) Functions: Currently Unliftable (3)

Partial functions

Theory: $\forall x \in M . \forall y, z \in N . P(x, y) \wedge P(x, z) \Rightarrow y = z$
Answer: $(n+1)^m$.

Partial injections

Theory:

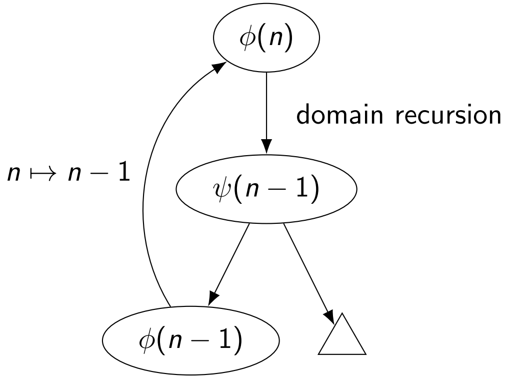
$$\forall x \in M . \forall y, z \in N . P(x, y) \wedge P(x, z) \Rightarrow y = z$$
$$\forall w, x \in M . \forall y \in N . P(w, y) \wedge P(x, y) \Rightarrow w = x$$

My answer: $\sum_{k=0}^{\min\{m,n\}} k! \binom{m}{k} \binom{n}{k}$.
Answer found by Forclift:

$$f(m, n) = \begin{cases} 1 & \text{if } m = 0 \\ \sum_{k=0}^{n} \binom{n}{k} [k < 2] f(m-1, k) & \text{otherwise.} \end{cases}$$

(exponential...)

Conceptual Plan of Action



Remarks on the Implementation

- WMC computation now loops over the graph.
- We propagate information (e.g., for smoothing) in reverse until convergence.
- Need a good way to recognise equivalent/isomorphic theories.
- Need to be careful about the order of operations:
  - Create a (half-empty) vertex v.
  - Add it to the cache.
  - Recurse on its direct successors S.
  - Add the edges from v to S.
  - After the graph is constructed, propagate information through the graph that would otherwise cause infinite loops.