# Knowledge Compilation for Lifted Probabilistic Inference

## Compiling to a Low-Level Program

Seyed Mehran Kazemi & David Poole

## Markov Logic Networks

- Weighted Formula (WF): $< \{x, m\},\ \neg likes(x, m) \vee rates(x, m),\ 1.4 >$

$x \in \{X_1, X_2, X_3, X_4, X_5\}$
$m \in \{M_1, M_2, M_3, M_4, M_5\}$

Logical variables | A first-order clause | Weight

- A **Markov logic network (MLN)** consists of a set of weighted formulae:

$$WF_1: < \{x, m\}, \neg likes(x, m) \vee rates(x, m), 0.8 >$$
$$WF_2: < \{x, m\}, \neg comedy(m) \vee likes(x, m), 0.6 >$$

- For a world $\omega$ in which $likes(X1, M1), rates(X1, M2), comedy(M1), comedy(M2)$ and the other atoms are false:

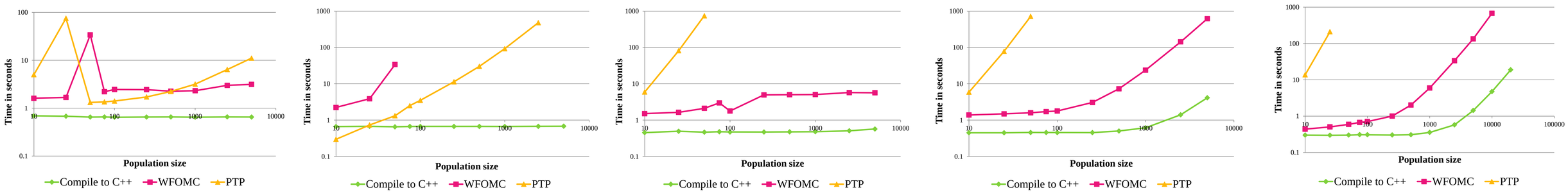
$$P(\omega) = \frac{1}{Z} \exp(24 * 0.8) * \exp(16 * 0.6)$$

$\eta(\omega, WF_1) = 24$
$\eta(\omega, WF_2) = 16$

$$Z = \sum_{\omega'} \exp(\eta(\omega', WF_1) * 0.8) * \exp(\eta(\omega', WF_2) * 0.6)$$

## Results on Lifted Inference

- We comparing "compile to C++" with weighted first-order model counting (WFOMC) and probabilistic theorem proving (PTP), the state of the art lifted inference softwares, on different benchmarks:
  - PTP does lifted inference using lifted search.
  - WFOMC does lifted inference by compiling to a data structure.
  - We do lifted inference by compiling to a low-level language (e.g. C/C++) instead of a data structure.
- The following 5 diagrams correspond (from left to right) to the benchmarks below:
  1) $goodProf(x) \wedge goodStudent(y) \wedge advises(x, y) \Rightarrow futureProf(y)$, $coauthor(x, y) \Rightarrow advises(x, y)$, varying |y|
  2) Same as (1) but varying |x| and |y| at the same time
  3) $a(x) \wedge b(y), a(x) \wedge c(x), b(y) \wedge d(y), c(x) \wedge d(y), e \wedge d(y)$, varying |y|
  4) Same as (3) but varying |x| and |y| at the same time
  5) $a(x) \wedge b(x) \wedge c(x, m) \wedge d(m) \wedge e(m) \wedge f$, varying |x| and |m| at the same time



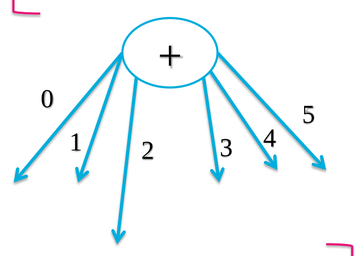

## Calculating Z by Lifted Search

$$Z \left[ \begin{array}{l} < \{x, m\}, \neg l(x, m) \vee r(x, m), 0.8 > \\ < \{x, m\}, \neg c(m) \vee l(x, m), 0.6 > \end{array} \right]$$

Lifted decomposition on $m$

$$pow\left(Z \left[ \begin{array}{l} < \{x\}, \neg l(x, M_1) \vee r(x, M_1), 0.8 > \\ < \{x\}, \neg c(M_1) \vee l(x, M_1), 0.6 > \end{array} \right], 5\right)$$

Lifted case analysis on $l(x, M_1)$




The $i$-th branch represents $l(x, M_1)$ is true for exactly $i/5$ individuals and must be multiplied by C(5,i): the number of ways one can choose $i$ from 5 individuals to be true.

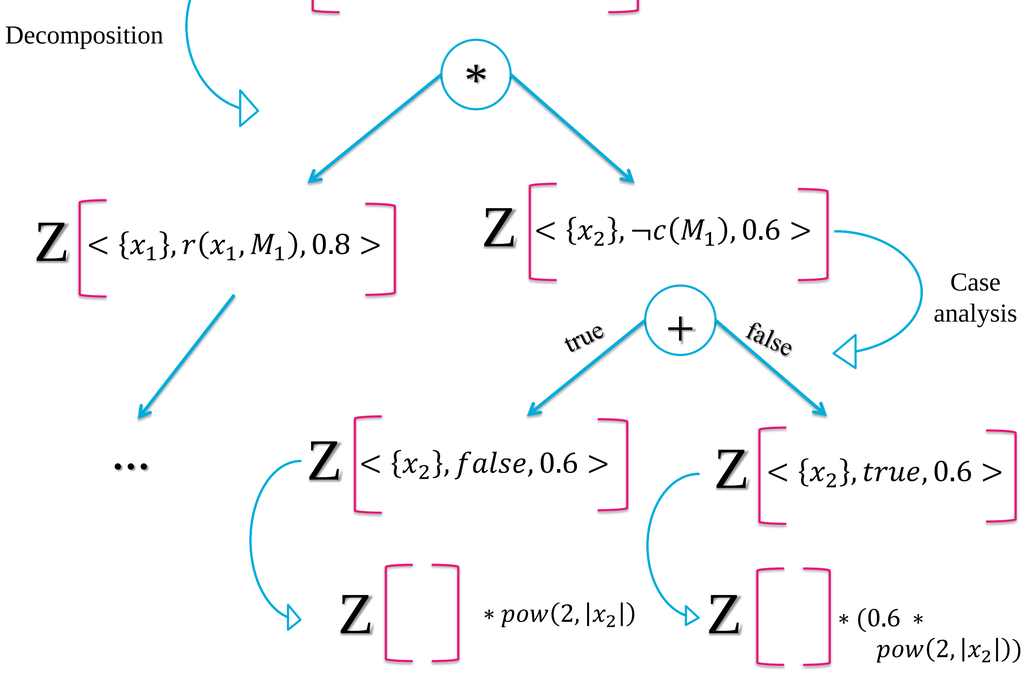
$$Z \left[ \begin{array}{l} < \{x_1\}, false \vee r(x_1, M_1), 0.8 > \\ < \{x_2\}, true \vee r(x_2, M_1), 0.8 > \\ < \{x_1\}, \neg c(M_1) \vee true, 0.6 > \\ < \{x_2\}, \neg c(M_1) \vee false, 0.6 > \end{array} \right]$$

$x_1$ represents the 2 individuals for which $l(x, M_1)$ is true and $x_2$ represents the other 3 individuals

Evaluating and simplifying WFs

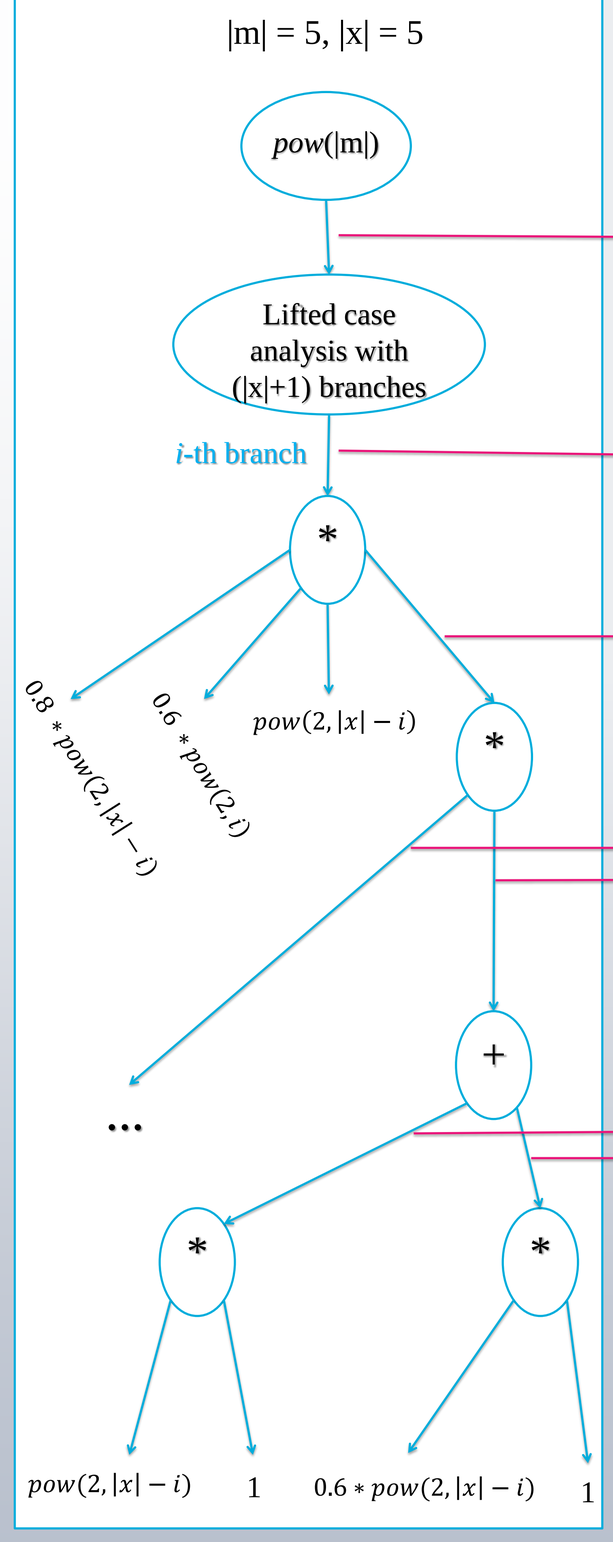
$$Z \left[ \begin{array}{l} < \{x_1\}, r(x_1, M_1), 0.8 > \\ < \{x_2\}, \neg c(M_1), 0.6 > \end{array} \right] * (0.8 * pow(2, |x_2|)) * (0.6 * pow(2, |x_1|)) * pow(2, |x_2|)$$

Decomposition




## Calculating Z by Knowledge Compilation

### Compiling to a Data Structure




### Compiling to C++

```
pop_m = 5;
pop_x = 5;
```

C++ code which stores Z of the sub-graph in *v2*

```
v1 = pow(v2, pop_m);
```

```
v2 = 0;
for(int i = 0; i <= pop_x; i++){
```

C++ code which stores Z of the *i*-th sub-graph in *v3*

```
  v2 += choose(pop_x, i) * v3;
}
```

C++ code which stores Z of the rightmost sub-graph in *v4*

```
v3 = (0.8 * pow(2, pop_x - i)) *
     (0.6 * pow(2, i)) * pow(2, pop_x - i) * v4;
```

C++ code which stores Z of the leftmost sub-graph in *v5*
C++ code which stores Z of the rightmost sub-graph in *v6*

```
v4 = v5 * v6;
```

C++ code which stores Z of the leftmost sub-graph in *v7*
C++ code which stores Z of the rightmost sub-graph in *v8*

```
v6 = v7 + v8;
```

```
v7 = pow(2, pop_x - i);
v8 = 0.6 * pow(2, pop_x - i);
```

After generating and running the above C++ program, the Z of the MLN will be stored in *v1*.

## References


1) Matthew Richardson and Pedro Domingos. 2006. *Markov logic networks*. Machine Learning 62:107–136.
2) Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. *Lifted probabilistic inference by first-order knowledge compilation*. In Proceedings of International Joint Conference on AI (IJCAI) , pages 2178–2185, 2011.
3) Vibhav Gogate and Pedro Domingos. *Probabilistic theorem proving*. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, pages 256–265, 2011.