Background: Knowledge Compilation

1) Transform logical theory into circuit language
2) Efficiently solve certain inference tasks

- Theory can be propositional or first-order
- For example, compile into a d-DNNF circuit for model counting:

\[ \forall x, y \in \text{People} : \text{smokes}(x) \land \text{friends}(x, y) \Rightarrow \text{smokes}(y) \]

![Model Counting](image)

- Application: Lifted Probabilistic Inference

Problem: Conditioning a Circuit

- Condition formula \( \Sigma \) on term \( \gamma \) by replacing atoms of \( \gamma \) in \( \Sigma \) by true or false.
- All propositional circuits can be conditioned in polytime.

\[ \Sigma = \text{sun} \land \text{rain} \Rightarrow \text{rainbow} \]
\[ \gamma = \neg \text{rainbow} \]
\[ \Sigma|_{\gamma} = \neg \text{sun} \lor \neg \text{rain} \]

![Conditioning](image)

Can we efficiently condition a first-order d-DNNF circuit?

Research Question

Application: Lifted Probabilistic Inference

- First-order knowledge compilation for lifted inference
- Logical conditioning to compute conditional probabilities

Positive Result: Algorithm when evidence consists of propositions and unary relations

Negative Result: In any first-order probabilistic model with a minimal expressivity, computing conditional probabilities exactly is \#P-hard.

Conclusions

- Showed when conditioning is possible in first-order knowledge compilation
- Algorithm for propositions and unary relations
- Proof that conditioning on higher-arity relations is \#P-hard
- Implications for lifted probabilistic inference

<table>
<thead>
<tr>
<th>Take-Away Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightly A</td>
</tr>
<tr>
<td>Polynomial if supported by compilation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Literal Arity</th>
<th>Complexity of Conditioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Polynomial</td>
</tr>
<tr>
<td>\geq 2</td>
<td>#P-hard</td>
</tr>
</tbody>
</table>