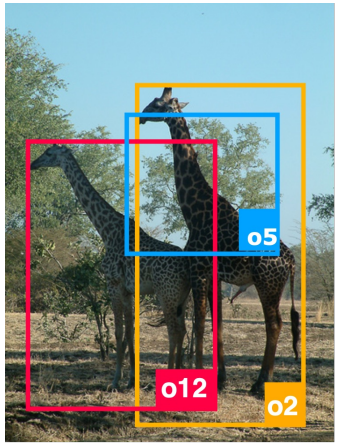
Part 1: Reasoning over Implicit Graphs

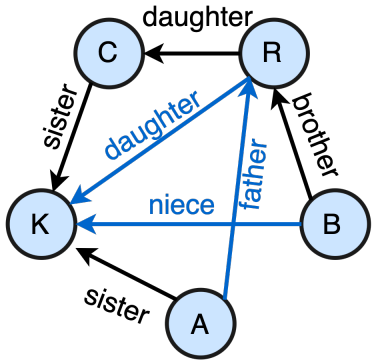
Symbolic Reasoning over Implicit Graphs

Unstructured Input
(Natural language Texts, Images, Audio)

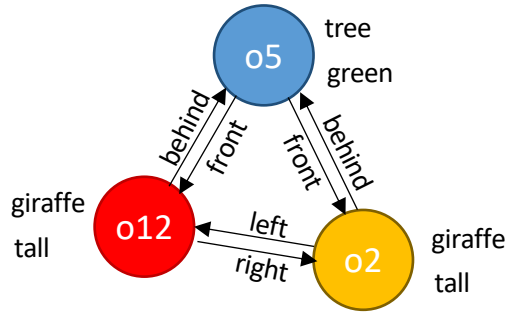
Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.



Graph Representation



Kinship Graph from Natural Language



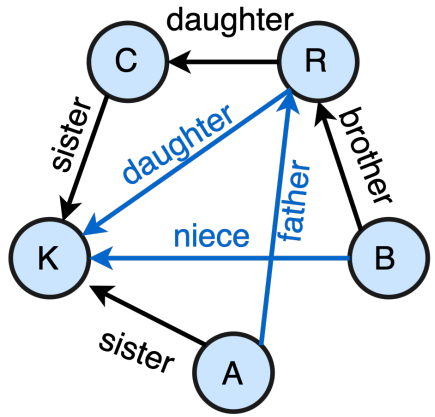
Scene Graph from Images

Symbolic Reasoning over Implicit Graphs

- Unstructured input contains implicit graphs
- Neural components transform unstructured input to structured input (graphs), learning representations
- And then, discrete symbolic & logical reasoning engine (e.g. Datalog, Prolog) can be applied to structured graphs to derive results

Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network



Symbolic Reasoner

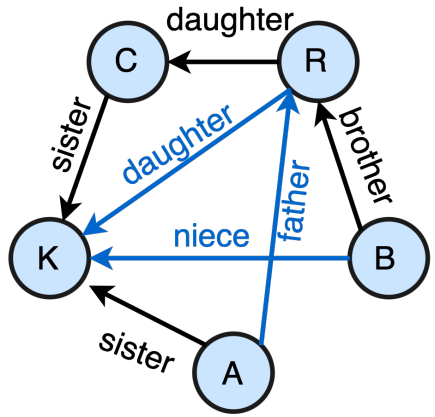
Rich is Anne’s Father

Symbolic Reasoning over Implicit Graphs

- Advantages:
  - Neural components and symbolic reasoning are disentangled
  - Neural networks are responsible for processing noisy inputs, handling ambiguity
  - Symbolic reasoners are sound and exact, generalizes better, supporting multi-hop reasoning, constraint solving, and can process external knowledge graphs

Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network



Symbolic Reasoner

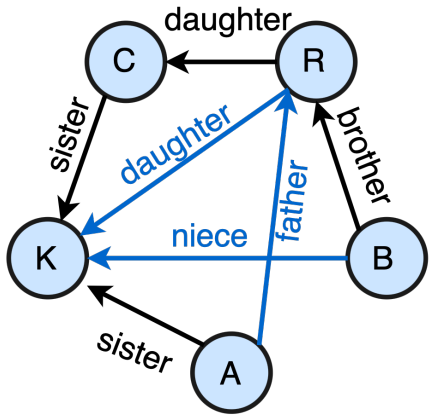
Rich is Anne’s Father

Symbolic Reasoning over Implicit Graphs

- Challenges:
  - If without supervision on the intermediate graph, how do we train the underlying neural networks to predict the graph?
  - How do we implement the symbolic reasoner? What if we do not want to specify the logical reasoning rules? Can the learning systems automatically infer the rules?

Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network



Symbolic Reasoner

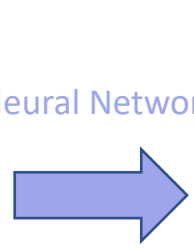
Rich is Anne’s Father

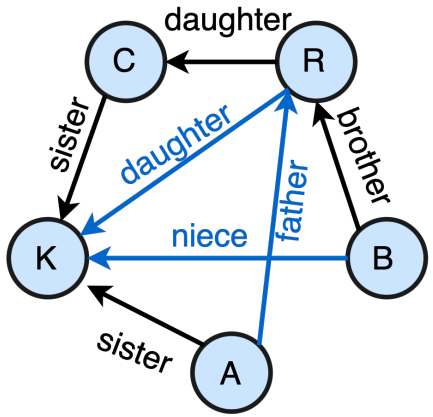
Symbolic Reasoning over Implicit Graphs

- Challenges:
  - If without supervision on the intermediate graph, how do we train the underlying neural networks to predict the graph?
    - [Differentiable Symbolic Reasoning]
  - How do we implement the symbolic reasoner? What if we do not want to specify the logical reasoning rules? Can the learning systems automatically infer the rules?
    - [First-order Logic, Inductive Logic Programming]

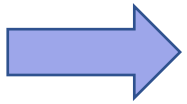
Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network





Symbolic Reasoner



Rich is Anne's Father

Agenda

- Differentiable Symbolic Reasoning Engines
  - TensorLog, DeepProbLog, Scallop (ours)
- Rule Learning
- Integrity Constraints and Semantic Loss
- Live demo with Scallop

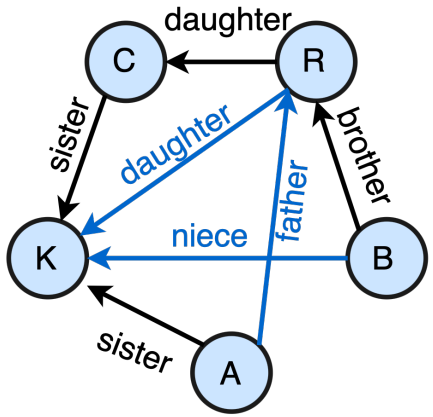
Differentiable Symbolic Reasoning

- Discrete graph is represented as Deductive Database (DDB)
- A fact under a relation (e.g. brother) connects one or more entities (e.g. B, R)

Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network

```
brother(B, R)
daughter(R, C)
sister(C, K)
…
```



Symbolic Reasoner

Rich is Anne’s Father

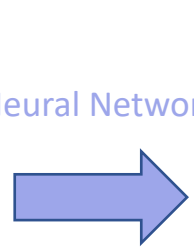
Differentiable Symbolic Reasoning

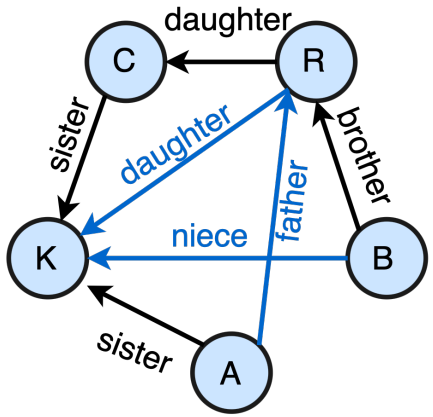
- For differentiability, we attach weights to each fact, where the weights are predicted by the underlying neural networks

```
0.01::father(B, R);    0.00::father(C, K);
0.98::brother(B, R);   0.02::brother(C, K);
0.00::sister(B, R);    0.95::sister(C, K);
…                      …
```

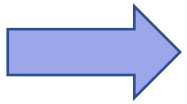
Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network





Symbolic Reasoner



Rich is Anne's Father

Differentiable Symbolic Reasoning

- Rules for logic reasoning are expressed as horn clauses
- A set of rules form a program to deduce new facts; rules are applied iteratively until no more fact is derived

```
daughter(a, c) :- daughter(a, b) and sister(b, c)
```

Daughter's sister is daughter

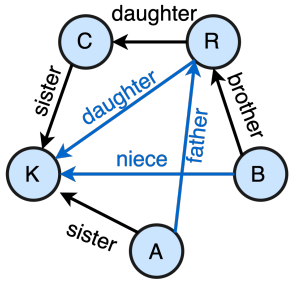
Differentiable Symbolic Reasoning

- Rules for logic reasoning are expressed as horn clauses
- A set of rules form a program to deduce new facts; rules are applied iteratively until no more fact is derived

```
daughter(a, c) :- daughter(a, b) and sister(b, c)
```

Daughter's sister is daughter

$$\frac{\texttt{daughter(R, C)} \qquad \texttt{sister(C, K)}}{\texttt{daughter(R, K)}}$$



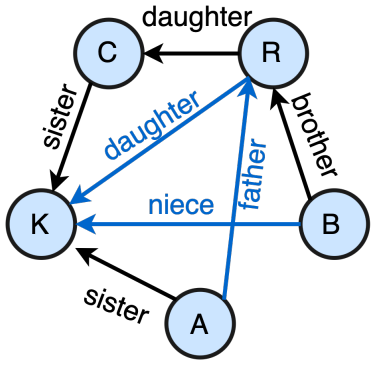
Differentiable Symbolic Reasoning

- How to incorporate weights and probabilities with logical reasoning?

$$\frac{\texttt{0.95::daughter(R, C)} \qquad \texttt{0.92::sister(C, K)}}{\texttt{??::daughter(R, K)}}$$

TensorLog (Cohen, 2017)

- Assume that relations are of arity-1 or arity-2, and that there are n entities:
  - Each entity is represented as a one-hot vector in $\mathbb{R}^n$
  - Arity-1 relations are 1 dimensional vectors in $\mathbb{R}^n$
  - Arity-2 relations are 2 dimensional matrices in $\mathbb{R}^{n \times n}$



$$R_{\text{sister}} = \begin{array}{c} \begin{matrix} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{K} & \texttt{R} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.95 & 0 & 0.96 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} \texttt{A} \\ \texttt{B} \\ \texttt{C} \\ \texttt{K} \\ \texttt{R} \end{matrix}$$

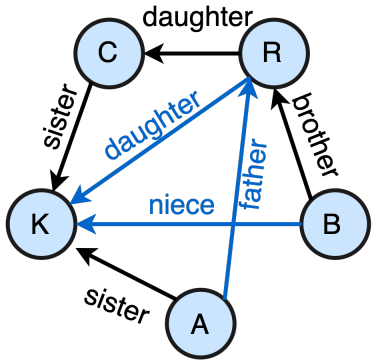
$$\mathbf{e}_{\texttt{B}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \texttt{A} \\ \texttt{B} \\ \texttt{C} \\ \texttt{K} \\ \texttt{R} \end{matrix}$$

TensorLog (Cohen, 2017)

- Rules and queries are compiled into matrix and vector multiplications
- Equivalent to performing fuzzy logic using AND (*) and OR (+)
- Recursion is handled by manually specifying iteration count

```
answer(x) :- sister(C, x)
```

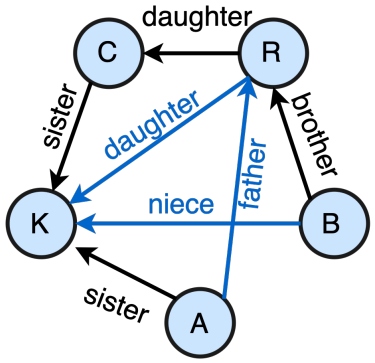
Who is (C)hristine's sister?



$$r = R_{\texttt{sister}}\mathbf{e}_{\texttt{B}} = \begin{array}{c} \begin{matrix} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{K} & \texttt{R} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.95 & 0 & 0.96 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} \texttt{A} \\ \texttt{B} \\ \texttt{C} \\ \texttt{K} \\ \texttt{R} \end{matrix} \\ \cdot \\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} \texttt{A} \\ \texttt{B} \\ \texttt{C} \\ \texttt{K} \\ \texttt{R} \end{matrix} \\ = \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.96 \\ 0 \end{bmatrix} \end{array} \begin{array}{c} \texttt{A} \\ \texttt{B} \\ \texttt{C} \\ \texttt{K} \\ \texttt{R} \end{array}$$

DeepProbLog

- DeepProbLog is another language for differentiable logic programming
- It is backed by differentiable probabilistic inference over possible worlds semantics – each outcome might be derived by many possible worlds

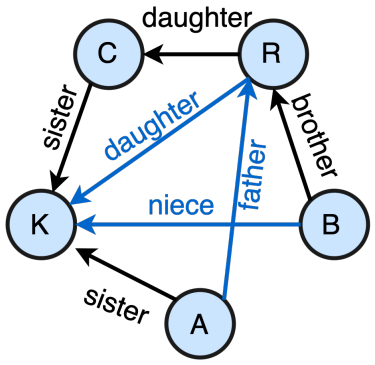



```
0.9::daughter(R, C)      sister(C, K)
-------------------------------------  [World 1]
          0.9::daughter(R, K)

 0.1::son(R, C)          sister(C, K)
-------------------------------------  [World 2]
          0.1::daughter(R, K)
```

DeepProbLog

- At the end, for exact probabilistic inference, one needs to systematically aggregate over all the possible worlds to obtain the result probability. This derivation process is also differentiable, and thus the entire pipeline can be trained fully
- Using a process called *Weighted Model Counting* (WMC), which is very computationally expensive (#P-complete)




```
0.9::daughter(R, C)      sister(C, K)
-------------------------------------  [World 1]
        0.9::daughter(R, K)

 0.1::son(R, C)          sister(C, K)
-------------------------------------  [World 2]
        0.1::daughter(R, K)
```

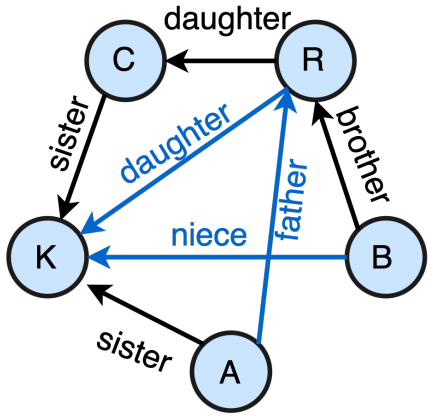
Scallop (Huang, 2021)

- We present Scallop, a neurosymbolic language and framework generalizing differentiable logical reasoning paradigms through *provenance semiring. It* supports all existing fuzzy logic operators (t-norms) as well as the DeepProbLog style possible worlds semantics
- Scallop is based on Datalog, and thus naturally operates on graphs and deductive databases
- It offers an easy-to-use PyTorch interface in order to be used along with existing models such as BERT and ResNet
- One can simultaneously integrate Graph Extraction, Inductive Logic Programming, and Integrity Constraint Reasoning

Graph Extraction / Representation Learning

Context:
Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network



Query:
Rich is Anne's ??

Symbolic Reasoner

Rich is Anne's Father

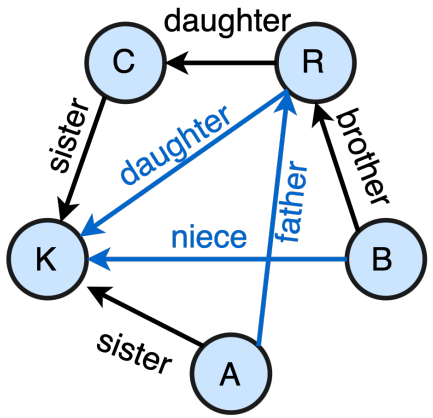
CLUTRR Dataset (TODO: Cite)

Graph Extraction / Representation Learning

Query:
Rich is Anne's ??

Context:
Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network



Symbolic Reasoner

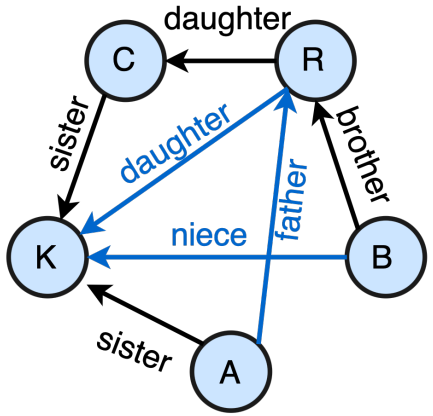
Rich is Anne's Father

We apply and fine-tune RoBERTa for Processing the natural language context into the graph

Graph Extraction / Representation Learning

Context:
Rich's daughter Christine made dinner for her sister Kim. Beth went to her brother Rich's birthday party. Anne went shopping with her sister Kim.

Neural Network



Query:
Rich is Anne's ??

Symbolic Reasoner

Rich is Anne's Father

We apply and fine-tune RoBERTa for Processing the natural language context into the graph

We manually craft a Scallop program as the Symbolic reasoner for this task

Higher Order Predicate / Rule Template

```
daughter(a, c) :- daughter(a, b) and sister(b, c)
```

```
composition(DAUGHTER, SISTER, DAUGHTER)
```

```
type Relation <: usize
type query(sub: String, obj: String)
type kinship(rela: Relation, sub: String, obj: String)
const MOTHER = 0, FATHER = 1, DAUGHTER = 2, SON = 3, ...

rel composition = {(DAUGHTER, SISTER, DAUGHTER), /* … 92 composition facts */}
rel kinship(r3, x, z) = composition(r1, r2, r3), kinship(r1, x, y), kinship(r2, y, z), x != z
rel answer(r) = question(s, o), kinship(r, s, o)
```

Template Based Rule Learning

- For our motivating task of kinship relation reasoning, one single template “composition” is enough for all deduction
- Given that there is a finite number of relations (20) and that “composition” relation is arity-3, there are in total $20^3 = 8000$ possible “composition” facts
- We can attach learnable weights to each “composition fact”, and allow optimizer to update these weights
- At the end, composition facts with high weights are treated as “learnt rules”

```
0.99 :: composition(DAUGHTER, SISTER, DAUGHTER);
0.01 :: composition(DAUGHTER, SISTER, SON);
0.00 :: composition(DAUGHTER, SISTER, FATHER);
…
```

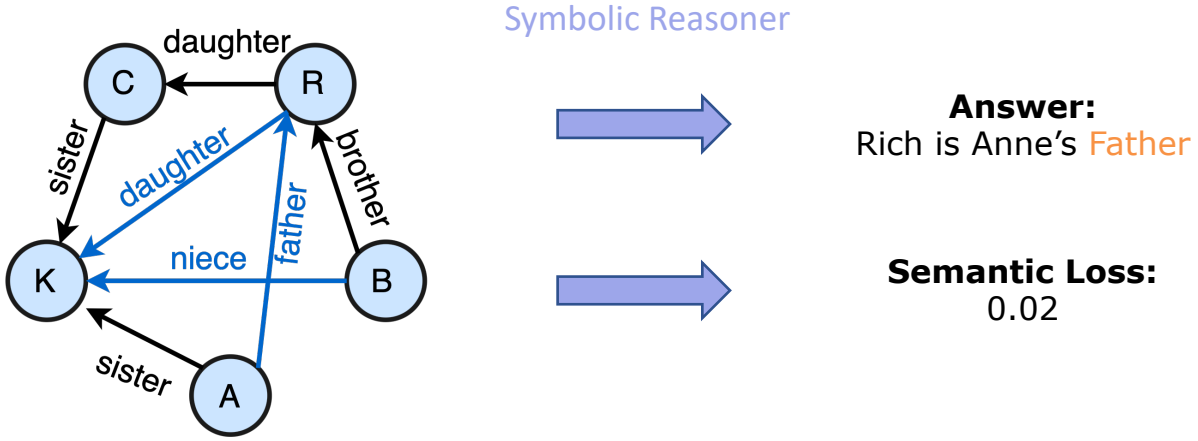
Integrity Constraint and Semantic Loss

- The kinship graph extracted by neural networks might be ill-formed
- i.e. Some integrity constraints maybe violated
- One example integrity constraint:
  - "If A is B's grandmother, then B should be A's grandson or granddaughter"
  - $\forall A, B, grandmother(A, B) \Rightarrow grandson(B, A)\ or\ granddaughter(B, A)$
  - Written in Scallop:

```
forall(a, b: kinship(GRANDMOTHER, a, b) implies
  kinship(GRANDSON, b, a) or kinship(GRANDDAUGHTER, b, a))
```

Integrity Constraint and Semantic Loss

- Violation of any integrity constraint is also associated with differentiable probability
- We can treat this “probability of violation” as *Semantic Loss*
- In addition to getting the correct result, we want to minimize the Semantic Loss during training as well, to penalize the model for generating ill-formed graphs



Putting it all together…

- Our model for natural language kinship reasoning has…
  - Graph Representation Extractor
    - (Fine-tuned Language Model for Named-Entity Recognition and Relation Extraction)
  - Composition rules and their learnable weights
    - (Directly optimized using gradient descent)
  - Integrity constraints
    - (Used for generate semantic loss; there are 6 integrity constraints we manually crafted)
  - A Scallop program
    - Takes in the learnt graph, composition rules, and integrity constraints
    - Outputs the predicted relation and semantic loss

Dataset and Results

- CLUTRR dataset for kinship reasoning
  - The number of reasoning steps required: k = 2, 3, …, 10
  - To test systematic generalizability (i.e. generalizability to more difficult queries)
    - Training dataset: k = 2, 3
    - Testing dataset: k = 2, 3, …, 10
  - Train / Test: 10K / 3K

Live Demo

Part 1 Conclusion

- We presented a few differentiable symbolic reasoning engines, and in particular, Scallop, our general and scalable framework
- We discussed how one can use such engine to train an underlying neural model to extract graph information from unstructured data
- We talked about rule learning which is a helpful tool to lift the rule crafting burden from human programmers and gain interpretability from neural models
- We demonstrated a few integrity constraints and showed that it helps to improve the robustness of model to extract well-formed graph