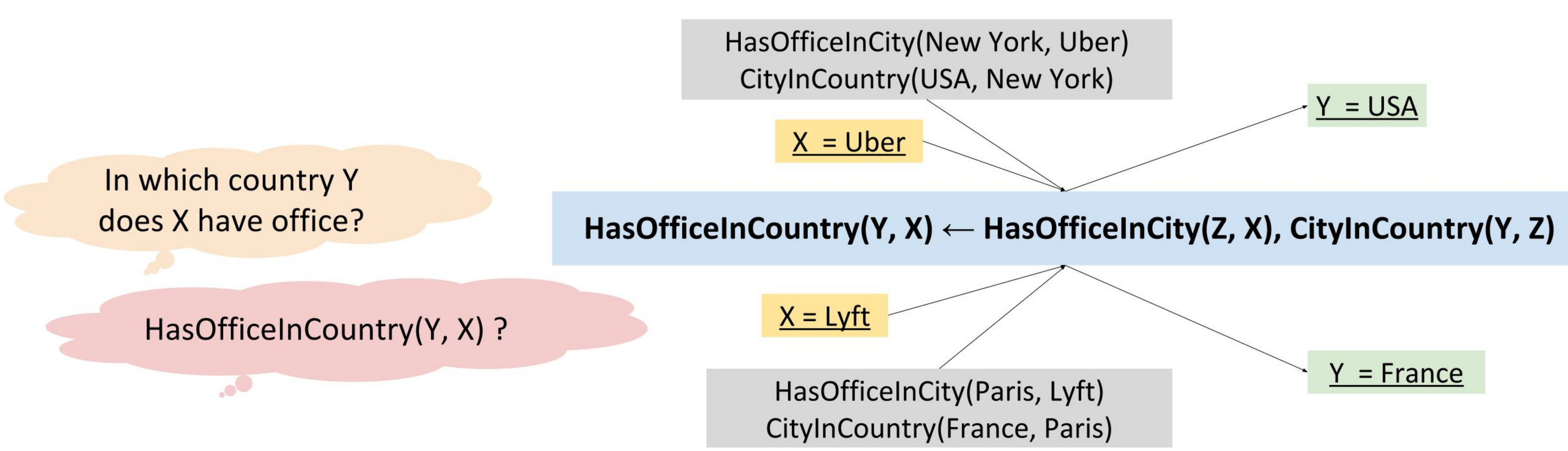
Differentiable Learning of Logical Rules for Knowledge Base Reasoning

Fan Yang, Zhilin Yang, William W. Cohen
Carnegie Mellon University
{fanyang1,zhiliny,wcohen}@cs.cmu.edu

Motivation

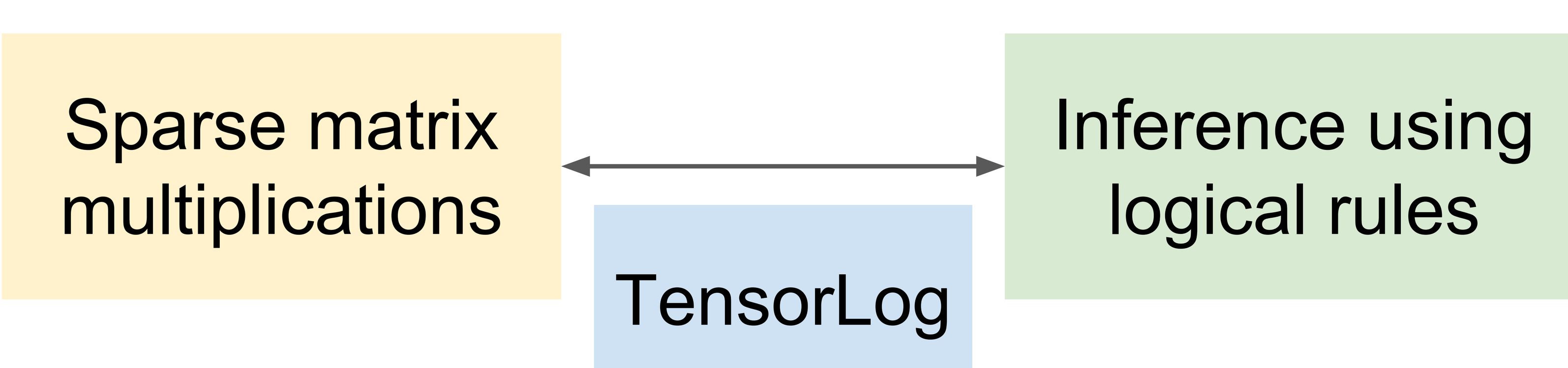
First-order logical rules are useful for knowledge base reasoning.
- **Interpretable**
- **Transferrable** to unseen entities.



Learning probabilistic logical rules is difficult -- it requires learning
- the **discrete structure**, i.e. the particular set of rules to include, and
- the **continuous parameters**, i.e. confidence associated with each rule.

Our Approach

An end-to-end **differentiable** framework -- *Neural Logic Programming* (Neural LP).



TensorLog operators

E = set of entities. R = set of binary relations.
- For each entity i, define v_i in $\{0, 1\}^{|E|}$.
- For each relation R, define M_R in $\{0, 1\}^{|E| \times |E|}$ where the (i, j) entry is 1 if and only if R(i, j),

Key idea

A neural controller that **learns to compose** TensorLog operators.

Learning -- Objective function

Learn **weighted chain-like logical rules** to reason over the knowledge base.

$$\alpha \ \texttt{query}(Y, X) \leftarrow R_n(Y, Z_n) \wedge \cdots \wedge R_1(Z_1, X)$$

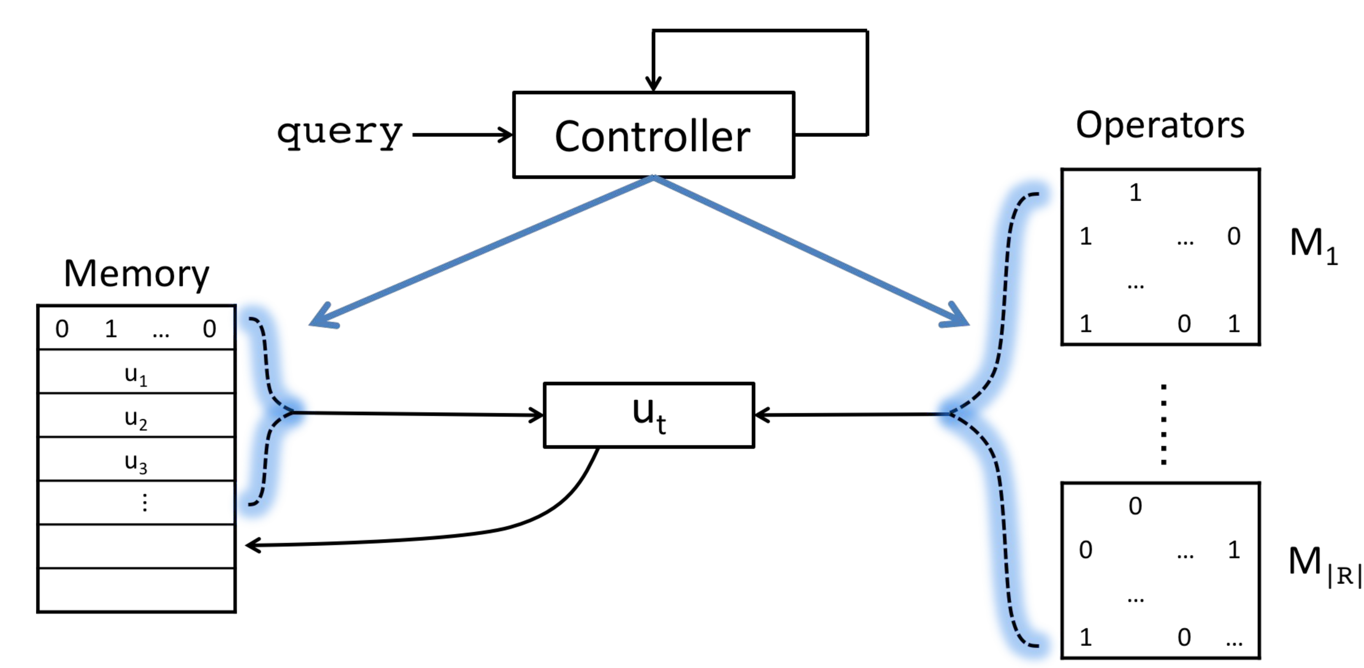
Using TensorLog operators, the **objective** is:

$$\max_{\{\alpha_l, \beta_l\}} \sum_{\{x,y\}} \text{score}(y \mid x) = \max_{\{\alpha_l, \beta_l\}} \sum_{\{x,y\}} \mathbf{v}_y^T \left(\sum_l \left(\alpha_l \left(\Pi_{k \in \beta_l} \mathbf{M}_{R_k} \mathbf{v}_x \right) \right) \right)$$

- l indexes over all possible rules
- α_l is the confidence of the rule
- β_l is an ordered list of all relations in the rule

Learning -- Recurrent formulation

An **equivalent but recurrent** formulation to allow **end-to-end differentiable** optimization.



At each step,
- predict **attentions over** TensorLog operators,
- use hidden states to **read from memory**,
- apply operators and write to memory.

$$\mathbf{u}_0 = \mathbf{v}_x$$

$$\mathbf{u}_t = \sum_k^{|R|} a_t^k \mathbf{M}_{R_k} \left(\sum_{\tau=0}^{t-1} b_t^\tau \mathbf{u}_\tau \right)$$

$$\mathbf{u}_{T+1} = \sum_{\tau=0}^{T} b_{T+1}^\tau \mathbf{u}_\tau$$

$$\mathbf{h}_t = \text{update}(\mathbf{h}_{t-1}, \text{input})$$

$$\mathbf{a}_t = \text{softmax}(W\mathbf{h}_t + b)$$

$$\mathbf{b}_t = \text{softmax}\left([\mathbf{h}_0, \ldots, \mathbf{h}_{t-1}]^T \mathbf{h}_t \right)$$

Experiments

Statistical relational learning

	ISG		Neural LP	
	T=2	T=3	T=2	T=3
UMLS	43.5	43.3	92.0	**93.2**
Kinship	59.2	59.0	**90.2**	90.1

WikiMovies with natural language queries.

Knowledge base	directed_by(Blade Runner,Ridley Scott) written_by(Blade Runner,Philip K. Dick) starred_actors(Blade Runner,Harrison Ford) starred_actors(Blade Runner,Sean Young)
Questions	What year was the movie Blade Runner released? Who is the writer of the film Blade Runner?

Model	Accuracy
Key-Value Memory Network	93.9
Neural LP	**94.6**

Knowledge base completion

- Example of learned rules

1.00	partially_contains(C,A) ← contains(B,A) ∧ contains(B,C)
0.45	partially_contains(C,A) ← contains(A,B) ∧ contains(B,C)
0.35	partially_contains(C,A) ← contains(C,B) ∧ contains(B,A)
1.00	marriage_location(C,A) ← nationality(C,B) ∧ contains(B,A)
0.35	marriage_location(B,A) ← nationality(B,A)
0.24	marriage_location(C,A) ← place_lived(C,B) ∧ contains(B,A)
1.00	film_edited_by(B,A) ← nominated_for(A,B)
0.20	film_edited_by(C,A) ← award_nominee(B,A) ∧ nominated_for(B,C)

- **Inductive** and **transductive** settings

	WN18	FB15K	FB15KSelected
TransE	0.01	0.48	0.53
Neural LP	**94.49**	**73.28**	**27.97**
Node+LinkFeat	94.3	**87.0**	34.7
DistMult	94.2	57.7	**40.8**
Neural LP	**94.5**	83.7	36.2