End-to-End Differentiable Proving

Tim Rocktäschel[1] and Sebastian Riedel[2,3]

[1]University of Oxford, [2]University College London, [3]Bloomsbury AI

Motivation

Logic-based Expert Systems

- No training data
- Interpretable
- No generalization beyond what is manually defined in rules

Representation Learning

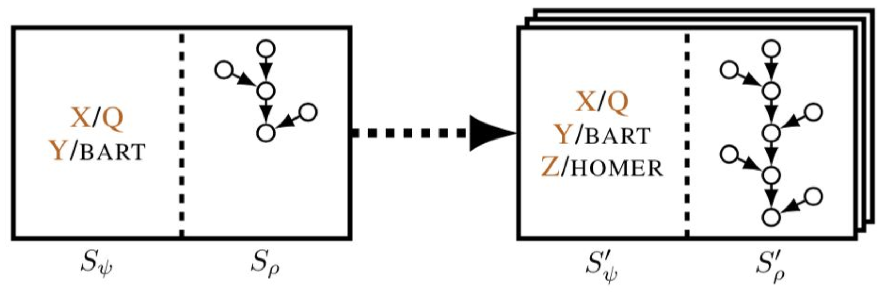
- Behavior is learned from input-output examples
- Achieves strong generalization
- Needs a lot of training data
- Generally not interpretable

Can we get the best of both worlds?

Differentiable Backward Chaining

- Neural network for proving queries to a knowledge base
- Proof success is differentiable with respect to vector representations of symbols
- **Learn vector representations of symbols** using SGD
- **Make use of provided rules** in soft proofs
- **Induce interpretable first-order logic rules** using SGD

Proof States and Modules



- Proof state S = (φ,ρ) is a tuple consisting of
  - S_φ: Substitution set (variable bindings)
  - S_ρ: Neural network calculating real-valued proof success
- Modules map upstream proof state to a list of new proof states
  - Extending the substitution set (adding variable bindings)
  - Extending the neural network (adding nodes to comp. graph)

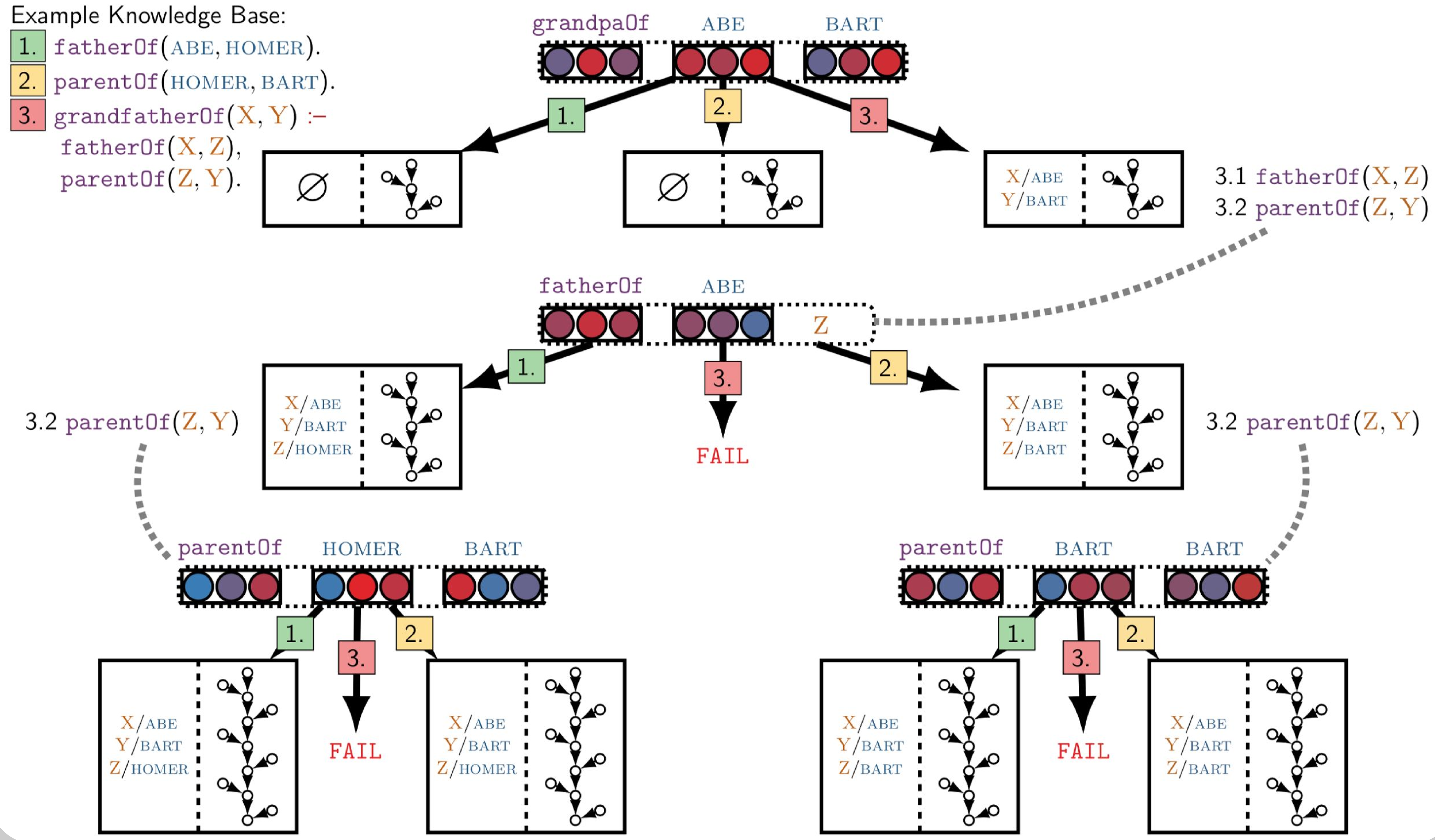
Unification

- Update substitution set S_φ by creating new variable bindings
- **Compare vector representations** of non-variable symbols using a Radial Basis Function kernel (extending neural net S_ρ)

1. $\mathtt{unify}_{\boldsymbol\theta}([\,],[\,],S) = S$
2. $\mathtt{unify}_{\boldsymbol\theta}([\,],\_,\_) = \mathtt{FAIL}$
3. $\mathtt{unify}_{\boldsymbol\theta}(\_,[\,],\_) = \mathtt{FAIL}$
4. $\mathtt{unify}_{\boldsymbol\theta}(h:\mathrm{H}, g:\mathrm{G}, S) = \mathtt{unify}_{\boldsymbol\theta}(\mathrm{H},\mathrm{G},S') = (S'_\psi, S'_\rho)$ where

$$S'_\psi = \left\{ \begin{array}{ll} S_\psi \cup \{h/g\} & \text{if } h \in \mathcal{V} \\ S_\psi \cup \{g/h\} & \text{if } g \in \mathcal{V}, h \notin \mathcal{V} \\ S_\psi & \text{otherwise} \end{array} \right\}, \quad S'_\rho = \min\left(S_\rho, \left\{ \begin{array}{ll} \exp\left(\frac{-\|\boldsymbol\theta_{h:} - \boldsymbol\theta_{g:}\|_2}{2\mu^2}\right) & \text{if } h, g \notin \mathcal{V} \\ 1 & \text{otherwise} \end{array} \right\}\right)$$

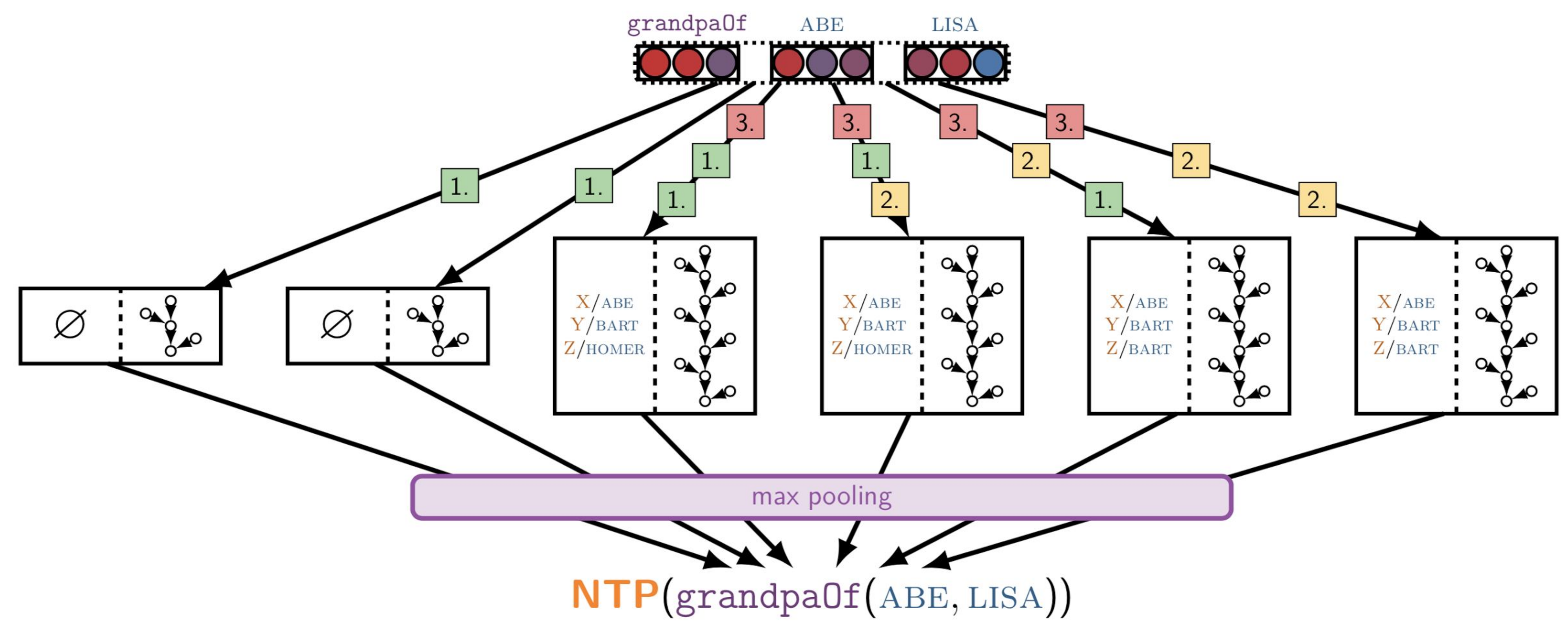
Example



Recursion

- Iterate through all rules in the knowledge base and unify goal with rule heads
  1. $\mathtt{or}^{\mathfrak{K}}_{\boldsymbol\theta}(\mathrm{G}, d, S) = [S' \mid S' \in \mathtt{and}^{\mathfrak{K}}_{\boldsymbol\theta}(\mathbb{B}, d, \mathtt{unify}_{\boldsymbol\theta}(\mathrm{H}, \mathrm{G}, S)) \text{ for } \mathrm{H} :\!-\, \mathbb{B} \in \mathfrak{K}]$
- Recursively prove subgoals in rule body
  1. $\mathtt{and}^{\mathfrak{K}}_{\boldsymbol\theta}(\_,\_,\mathtt{FAIL}) = \mathtt{FAIL}$
  2. $\mathtt{and}^{\mathfrak{K}}_{\boldsymbol\theta}(\_,0,\_) = \mathtt{FAIL}$
  3. $\mathtt{and}^{\mathfrak{K}}_{\boldsymbol\theta}([\,],\_,S) = S$
  4. $\mathtt{and}^{\mathfrak{K}}_{\boldsymbol\theta}(\mathrm{G}:\mathbb{G}, d, S) = [S'' \mid S'' \in \mathtt{and}^{\mathfrak{K}}_{\boldsymbol\theta}(\mathbb{G}, d, S') \text{ for } S' \in \mathtt{or}^{\mathfrak{K}}_{\boldsymbol\theta}(\text{substitute}(\mathrm{G}, S_\psi), d-1, S)]$

Training Objective



- Loss: $L(r_s(e_i,e_j), y) = -y \cdot \log(\text{NTP}(r_s(e_i,e_j))) - (1-y)\log(1-\text{NTP}(r_s(e_i,e_j)))$
- **NTPλ** variant: SotA neural link prediction model (**ComplEx**) as auxiliary task

Neural Inductive Logic Programming

- Architecture allows us to induce rules of predefined structure
- We can, for instance, incorporate the inductive bias of a transitivity relationship in the knowledge base

$$\boldsymbol\theta_1(X,Y) :\!-\, \boldsymbol\theta_2(X,Z),\ \boldsymbol\theta_3(Z,Y).$$

- $\boldsymbol\theta_i$ are vector representations for unknown predicates
- They can be learned like all other vector representations
- They can be decoded at test time by finding the closest known relation using the RBF kernel
- Rule confidence is minimum RBF similarity over all decodings
- Confidence is an upper bound on the proof success that can be achieved when applying the induced rule

Results

Knowledge Base		Metric	Model		
			ComplEx	NTP	NTPλ
Countries	S1	AUC-PR	99.37 ± 0.4	90.83 ± 15.4	**100.00** ± 0.0
	S2	AUC-PR	87.95 ± 2.8	87.40 ± 11.7	**93.04** ± 0.4
	S3	AUC-PR	48.44 ± 6.3	56.68 ± 17.6	**77.26** ± 17.0
Kinship		MRR	**0.81**	0.60	0.80
		HITS@1	0.70	0.48	**0.76**
Nations		MRR	*0.75*	*0.75*	0.74
		HITS@1	*0.62*	*0.62*	0.59
UMLS		MRR	0.89	0.88	**0.93**
		HITS@1	0.82	0.82	**0.87**

Induced Rules

Knowledge Base		Examples of induced rules and their confidence
Countries	S1	0.90 locatedIn(X,Y) :– locatedIn(X,Z), locatedIn(Z,Y).
	S2	0.63 locatedIn(X,Y) :– neighborOf(X,Z), locatedIn(Z,Y).
	S3	0.32 locatedIn(X,Y) :– neighborOf(X,Z), neighborOf(Z,W), locatedIn(W,Y).
Nations		0.68 blockpositionindex(X,Y) :– blockpositionindex(Y,X).
		0.46 expeldiplomats(X,Y) :– negativebehavior(X,Y).
		0.38 negativecomm(X,Y) :– commonbloc0(X,Y).
		0.38 intergovorgs3(X,Y) :– intergovorgs(Y,X).
UMLS		0.88 interacts_with(X,Y) :– interacts_with(X,Z), interacts_with(Z,Y).
		0.77 isa(X,Y) :– isa(X,Z), isa(Z,Y).
		0.71 derivative_of(X,Y) :– derivative_of(X,Z), derivative_of(Z,Y).

Limitations and Future Work

- Scale to larger knowledge bases (beyond 10k facts)
  - Hierarchical attention for unification with facts
  - Reinforcement learning for pruning proof tree
- Train jointly with RNNs that encode natural language statements which can then be used in proofs
- Learn to prove mathematical theorems
- Incorporate commonsense knowledge for Visual Q&A

Email: tim.rocktaschel@cs.ox.ac.uk s.riedel@cs.ucl.ac.uk Twitter: @_rockt @riedelcastro