GPU-accelerated End-to-end Differentiable Planning and Reasoning

Tim Rocktäschel

Whiteson Research Lab, University of Oxford

http://rockt.github.com Twitter: @_rockt tim.rocktaschel@cs.ox.ac.uk

Talk at GTC Europe, ID 23372

12th of October 2017

What vegetable is on the plate?
Neural Net: broccoli
Ground Truth: broccoli

What color are the shoes on the person's feet ?
Neural Net: brown
Ground Truth: brown

How many school busses are there?
Neural Net: 2
Ground Truth: 2

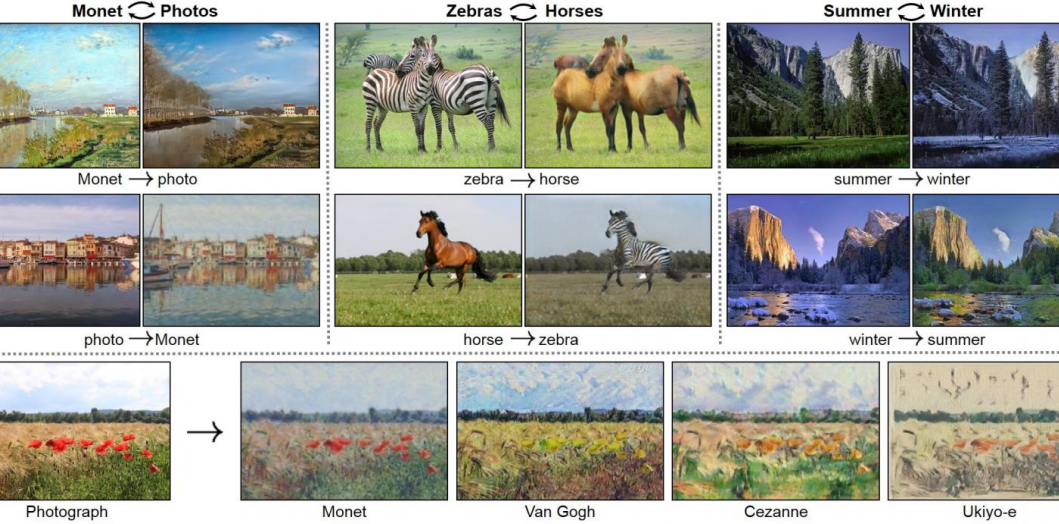
What sport is this?
Neural Net: baseball
Ground Truth: baseball

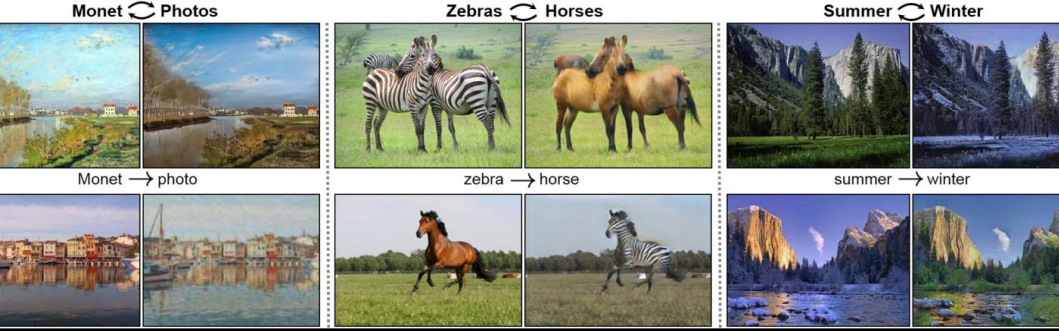
What is on top of the refrigerator?
Neural Net: magnets
Ground Truth: cereal

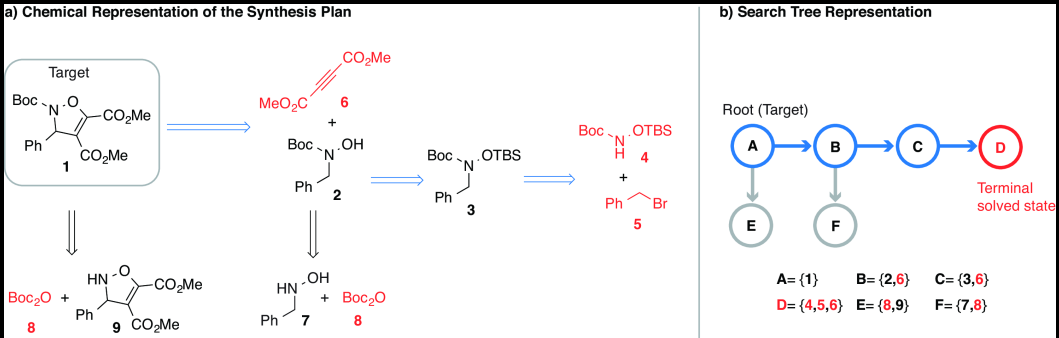
What uniform is she wearing?
Neural Net: shorts
Ground Truth: girl scout

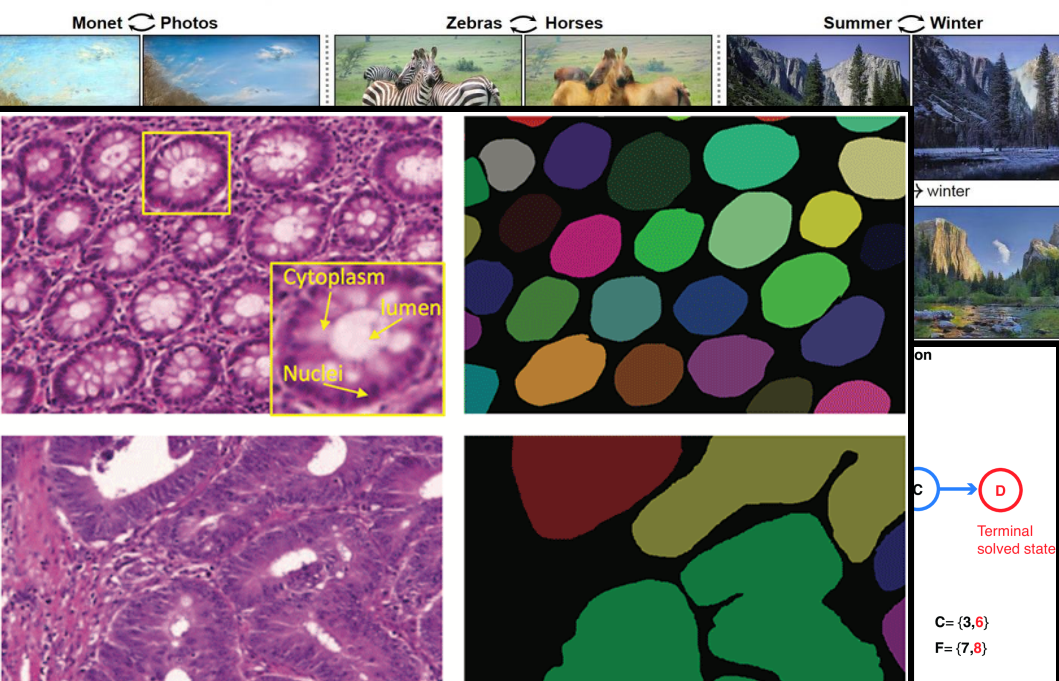
What is the table number?
Neural Net: 4
Ground Truth:40

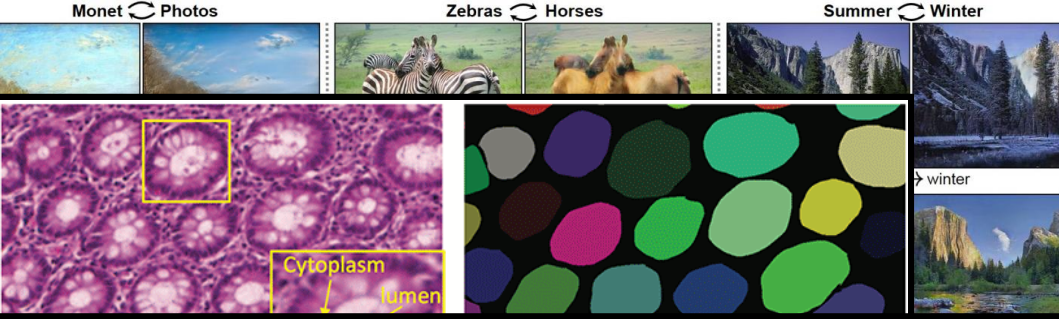
What are people sitting under in the back?
Neural Net: bench
Ground Truth: tent


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter


a) Chemical Representation of the Synthesis Plan
Target
1
2
3
4
5
6
7
8
9
b) Search Tree Representation
Root (Target)
A
B
C
D
E
F
Terminal solved state
A= {1}
B= {2,6}
C= {3,6}
D= {4,5,6}
E= {8,9}
F= {7,8}


Monet
Photos
Zebras
Horses
Summer
Winter
winter
Cytoplasm
lumen
Nuclei
on
C
D
Terminal
solved state
C= {3,6}
F= {7,8}



Translate from **GERMAN** (detected)

Die Polizei in den USA darf sich wieder schwere Ausrüstung und Waffen beim Militär besorgen. Das hat US-Präsident Donald Trump entschieden und so eine Anordnung seines Vorgängers Barack Obama aufgehoben, nach der es dem Verteidigungsministerium verboten war, die Polizei mit Granatwerfern, gepanzerten Fahrzeugen, Bajonetten, großkalibrigen Waffen und Munition auszurüsten.

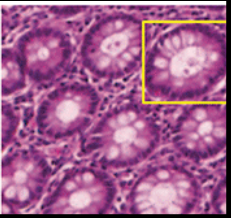
Mit der Maßnahme soll sichergestellt werden, dass die Polizei die lebensrettende Ausrüstung bekomme, die sie brauche, um ihren Job zu machen, sagte US-Justizminister Jeff Sessions.

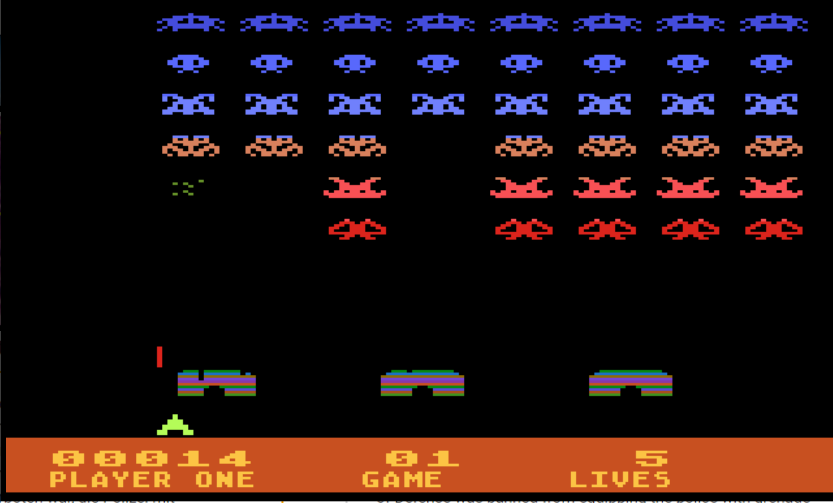
Translate into **ENGLISH**

The police in the USA are allowed to get heavy equipment and weapons from the military again. This was decided by US President Donald Trump, who overturned an order from his predecessor Barack Obama, according to which the Department of Defense was banned from equipping the police with grenade launchers, armoured vehicles, bayonets, large-calibre weapons and ammunition.

The measure is designed to ensure that the police get the life-saving equipment they need to do their job, US Attorney General Jeff Sessions said.

Monet Photos





Translat

Die Polizei in den USA darf si
Waffen beim Militär besorger
Trump entschieden und so ei
Barack Obama aufgehoben, r
Verteidigungsministerium ve
Granatwerfern, gepanzerten Fahrzeugen, Bajonetten, großkalibrigen Waffen und Munition auszurüsten.

Mit der Maßnahme soll sichergestellt werden, dass die Polizei die lebensrettende Ausrüstung bekomme, die sie brauche, um ihren Job zu machen, sagte US-Justizminister Jeff Sessions.

launchers, armoured vehicles, bayonets, large-calibre weapons and ammunition.

The measure is designed to ensure that the police get the life-saving equipment they need to do their job, US Attorney General Jeff Sessions said.

XKCD, 17th May 2017

Data &
Explanations
- Rules
- (Partial) Programs
- Natural Language

XKCD, 17th May 2017

Data &
Explanations
- Rules
- (Partial) Programs
- Natural Language

XKCD, 17th May 2017

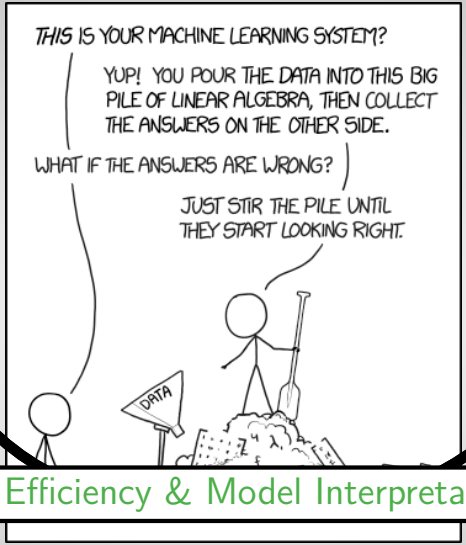
Answers &
Explanations
- Rules
- Programs
- Natural Language
- Plans

Data &
Explanations
- Rules
- (Partial) Programs
- Natural Language

XKCD, 17th May 2017

Answers &
Explanations
- Rules
- Programs
- Natural Language
- Plans

Data &
Explanations
- Rules
- (Partial) Programs
- Natural Language



Answers &
Explanations
- Rules
- Programs
- Natural Language
- Plans

Data Efficiency & Model Interpretability

XKCD, 17th May 2017

Joint work with

1. **End-to-end Differentiable Reasoning**, Application: Knowledge Base Inference, *NIPS 2017*

Sebastian Riedel, University College London

Joint work with

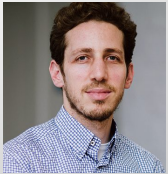
1. **End-to-end Differentiable Reasoning**, Application: Knowledge Base Inference, *NIPS 2017*

Sebastian Riedel, University College London

2. **End-to-end Differentiable Planning**, Application: Atari, *work-in-progress*

Gregory Farquhar

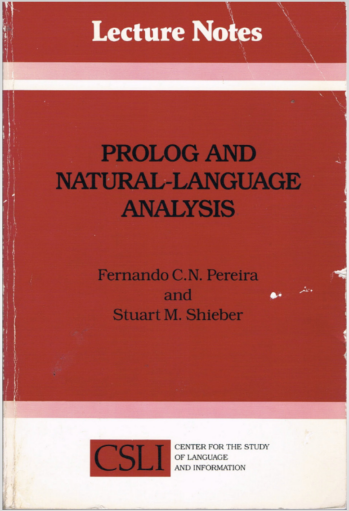
Maximilian Igl



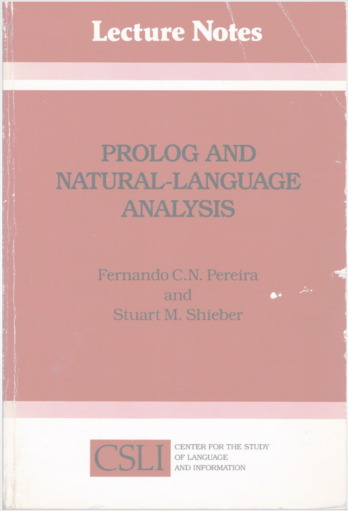
Shimon Whiteson

University of Oxford

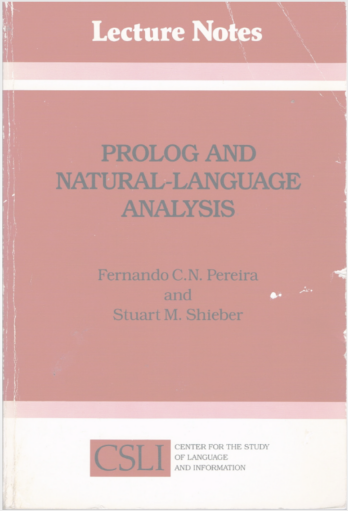
End-to-end Differentiable Reasoning


Lecture Notes
PROLOG AND NATURAL-LANGUAGE ANALYSIS
Fernando C.N. Pereira
and
Stuart M. Shieber
CSLI
CENTER FOR THE STUDY OF LANGUAGE AND INFORMATION

```
goal problem.

rule 1
  if not turn_over and
    battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
    gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
    gas_gauge is low
  then problem is out_of_gas cf 30.
```



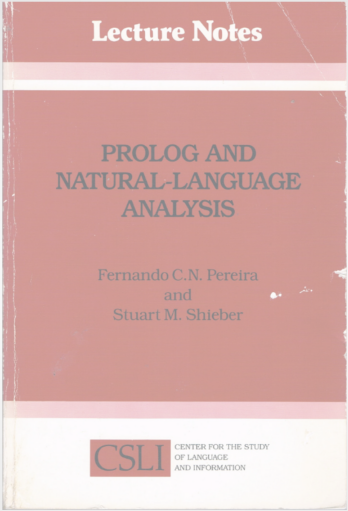
```
goal problem.

rule 1
  if not turn_over and
    battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
    gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
    gas_gauge is low
  then problem is out_of_gas cf 30.
```



Expert Systems

- No/little training data
- Interpretable

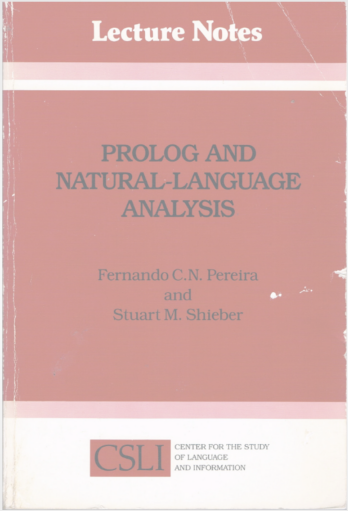
```
goal problem.

rule 1
  if not turn_over and
    battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
    gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
    gas_gauge is low
  then problem is out_of_gas cf 30.
```

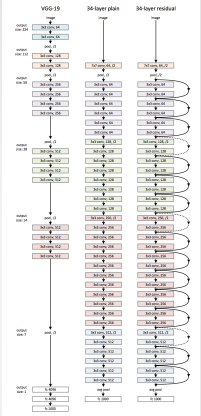


Expert Systems

- No/little training data
- Interpretable
- Behavior manually defined
- No generalization

```
goal problem.

rule 1
  if not turn_over and
    battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
    gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
    gas_gauge is low
  then problem is out_of_gas cf 30.
```

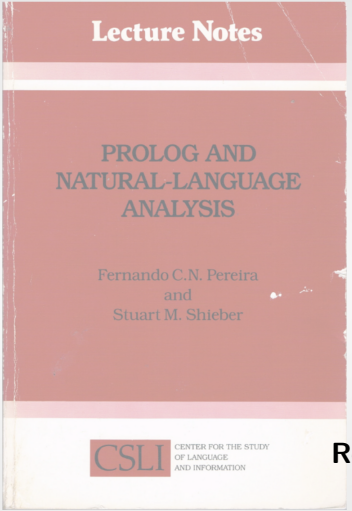


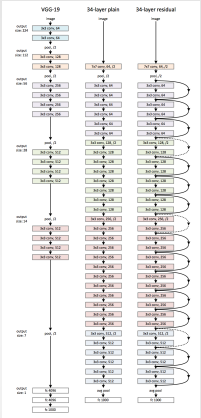


Expert Systems

- No/little training data
- Interpretable
- Behavior manually defined
- No generalization

```
goal problem.

rule 1
  if not turn_over and
     battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
    gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
    gas_gauge is low
  then problem is out_of_gas cf 30.
```





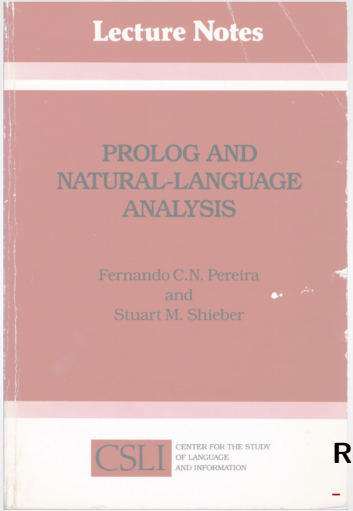
Expert Systems

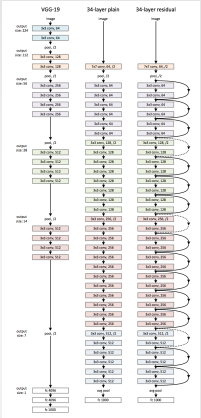
- No/little training data
- Interpretable
- Behavior manually defined
- No generalization

Representation Learning

- Behavior learned
- Strong generalization

```
goal problem.

rule 1
  if not turn_over and
     battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
    gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
    gas_gauge is low
  then problem is out_of_gas cf 30.
```





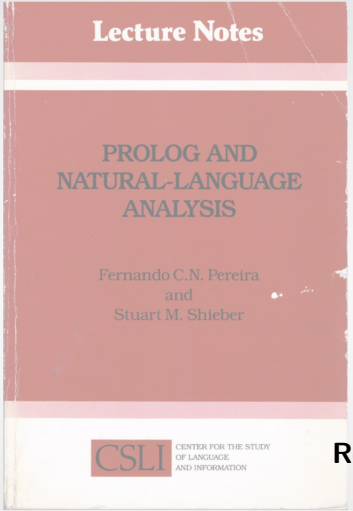
Expert Systems

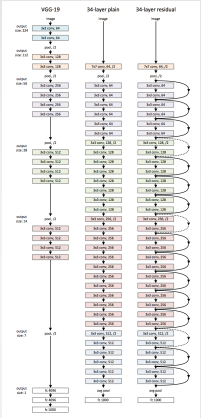
- No/little training data
- Interpretable
- Behavior manually defined
- No generalization

Representation Learning

- Lot of training data needed
- Not interpretable
- Behavior learned
- Strong generalization

```
goal problem.

rule 1
  if not turn_over and
     battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
 if turn_over and
    smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
     gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
     gas_gauge is low
  then problem is out_of_gas cf 30.
```





Expert Systems

- No/little training data
- Interpretable

Representation Learning

- Behavior learned
- Strong generalization

Nando de Freitas @NandoDF · 5 Aug 2016

Neuralise (verb,#neuralize): to implement a known thing with deep nets. Usage: Let's neuralize warping, neuralize this! And train it!

6 28 64

Yann LeCun
@ylecun

Following

Replying to @NandoDF

sort of like "kernelize" used to be.

10:11 AM - 5 Aug 2016

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases
- Incorporate background knowledge in form of rules
  $\Rightarrow$ **Data Efficiency**

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases
- Incorporate background knowledge in form of rules
  $\Rightarrow$ **Data Efficiency**
- Calculate gradient of proof success w.r.t. subsymbolic representations

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases
- Incorporate background knowledge in form of rules
  $\Rightarrow$ **Data Efficiency**
- Calculate gradient of proof success w.r.t. subsymbolic representations
- Rule application is explicit, but symbol comparison is neural

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases
- Incorporate background knowledge in form of rules
  $\Rightarrow$ **Data Efficiency**
- Calculate gradient of proof success w.r.t. subsymbolic representations
- Rule application is explicit, but symbol comparison is neural
- Use similarity between vector representations of symbols in proofs

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases
- Incorporate background knowledge in form of rules
  $\Rightarrow$ **Data Efficiency**
- Calculate gradient of proof success w.r.t. subsymbolic representations
- Rule application is explicit, but symbol comparison is neural
- Use similarity between vector representations of symbols in proofs
- Learn vector representations of symbols from data using gradient descent

Aims

- Modular construction of neural networks for end-to-end differentiable reasoning in knowledge bases
- Incorporate background knowledge in form of rules
  $\Rightarrow$ **Data Efficiency**
- Calculate gradient of proof success w.r.t. subsymbolic representations
- Rule application is explicit, but symbol comparison is neural
- Use similarity between vector representations of symbols in proofs
- Learn vector representations of symbols from data using gradient descent
- Induce interpretable logical rules from data by gradient descent
  $\Rightarrow$ **Model Interpretability**

Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!

Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly

Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts

Task: Link Prediction

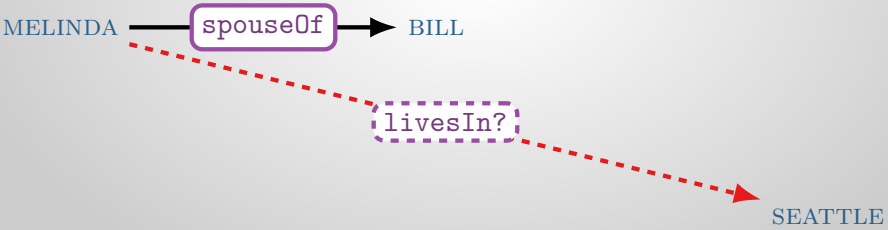
Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts

Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

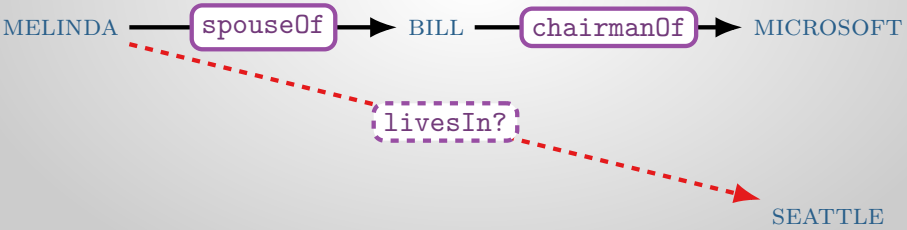
- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts



Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

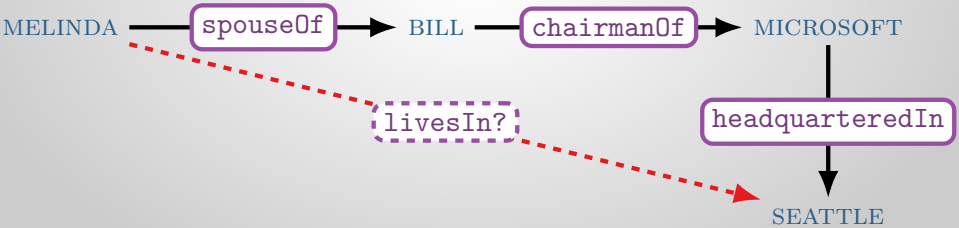
- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts



Task: Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts



Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)

Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)

Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable

Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
- **Predicate**: `fatherOf`, `parentOf` etc.
  function from terms to a Boolean

Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
- **Predicate**: `fatherOf`, `parentOf` etc.
  function from terms to a Boolean
- **Atom**: predicate and terms, e.g., `parentOf`(X, BART)

Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
- **Predicate**: `fatherOf`, `parentOf` etc.
  function from terms to a Boolean
- **Atom**: predicate and terms, e.g., `parentOf`(X, BART)
- **Rule**: `head :- body`.
  `head`: atom
  `body`: (possibly empty) list of atoms representing conjunction
  `grandfatherOf`(X, Y) :- `fatherOf`(X, Z), `parentOf`(Z, Y).

Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
- **Predicate**: `fatherOf`, `parentOf` etc.
  function from terms to a Boolean
- **Atom**: predicate and terms, e.g., `parentOf`(X, BART)
- **Rule**: `head :- body.`
  `head`: atom
  `body`: (possibly empty) list of atoms representing conjunction
  `grandfatherOf`(X, Y) :- `fatherOf`(X, Z), `parentOf`(Z, Y).
- **Fact**: ground rule (no free variables) with empty body, e.g.,
  `parentOf`(HOMER, BART).

Example Knowledge Base

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, LISA).
3. parentOf(HOMER, BART).
4. grandpaOf(ABE, LISA).
5. grandfatherOf(ABE, MAGGIE).

Example Knowledge Base

1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER}).$
2. $\texttt{parentOf}(\text{HOMER}, \text{LISA}).$
3. $\texttt{parentOf}(\text{HOMER}, \text{BART}).$
4. $\texttt{grandpaOf}(\text{ABE}, \text{LISA}).$
5. $\texttt{grandfatherOf}(\text{ABE}, \text{MAGGIE}).$
6. $\texttt{grandfatherOf}(\text{X}_1, \text{Y}_1) :-$
   $\texttt{fatherOf}(\text{X}_1, \text{Z}_1),$
   $\texttt{parentOf}(\text{Z}_1, \text{Y}_1).$
7. $\texttt{grandparentOf}(\text{X}_2, \text{Y}_2) :-$
   $\texttt{grandfatherOf}(\text{X}_2, \text{Y}_2).$

Prolog Backward Chaining Example

Example Knowledge Base:

1. `fatherOf(ABE, HOMER).`
2. `parentOf(HOMER, BART).`
3. `grandfatherOf(X, Y) :-`
   `fatherOf(X, Z),`
   `parentOf(Z, Y).`

Prolog Backward Chaining Example

Example Knowledge Base:

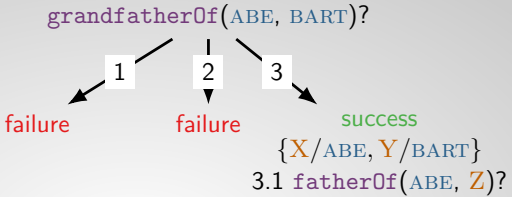
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

grandfatherOf(ABE, BART)?

Prolog Backward Chaining Example

Example Knowledge Base:

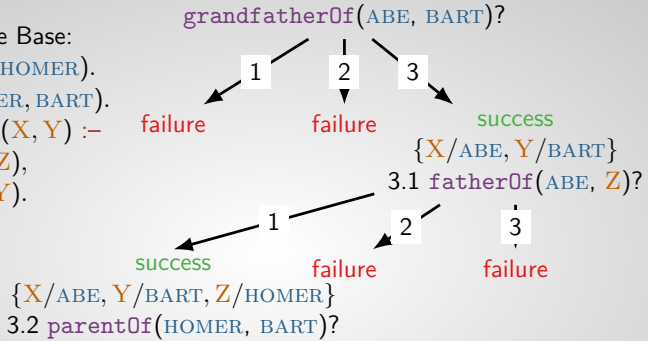
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :–
   fatherOf(X, Z),
   parentOf(Z, Y).



Prolog Backward Chaining Example

Example Knowledge Base:

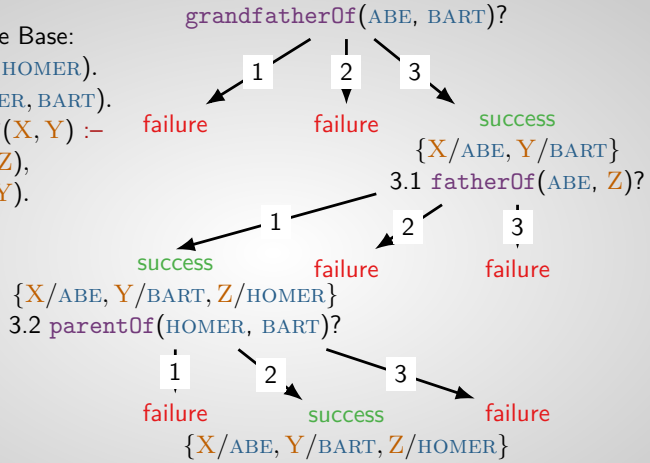
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :–
   fatherOf(X, Z),
   parentOf(Z, Y).



Prolog Backward Chaining Example

Example Knowledge Base:

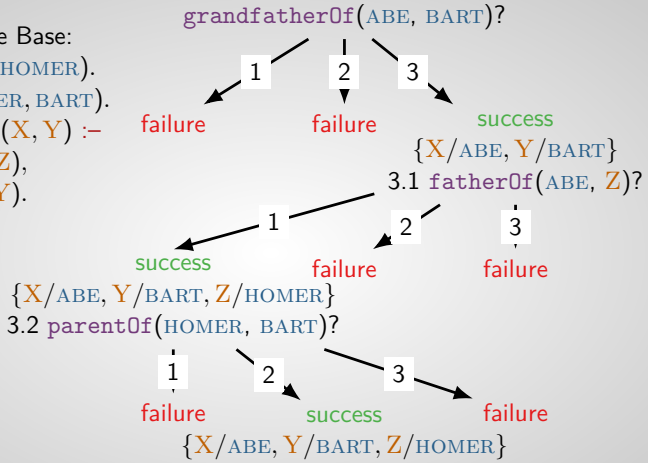
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :–
   fatherOf(X, Z),
   parentOf(Z, Y).



Prolog Backward Chaining Example

Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :–
   fatherOf(X, Z),
   parentOf(Z, Y).



What about grandpaOf(ABE, BART)?

Symbolic Representations

- Symbols (constants and predicates) do not share any information: `grandpaOf` $\neq$ `grandfatherOf`

Symbolic Representations

- Symbols (constants and predicates) do not share any information: $\texttt{grandpaOf} \neq \texttt{grandfatherOf}$
- No notion of similarity: $\text{APPLE} \sim \text{ORANGE}$, $\texttt{professorAt} \sim \texttt{lecturerAt}$

Symbolic Representations

- Symbols (constants and predicates) do not share any information: $\texttt{grandpaOf} \neq \texttt{grandfatherOf}$
- No notion of similarity: $\text{APPLE} \sim \text{ORANGE}$, $\texttt{professorAt} \sim \texttt{lecturerAt}$
- No generalization beyond what can be symbolically inferred: $\texttt{isFruit}(\text{APPLE})$, $\text{APPLE} \sim \text{ORGANGE}$, $\texttt{isFruit}(\text{ORANGE})$?

Symbolic Representations

- Symbols (constants and predicates) do not share any information: `grandpaOf` $\neq$ `grandfatherOf`
- No notion of similarity: APPLE $\sim$ ORANGE, `professorAt` $\sim$ `lecturerAt`
- No generalization beyond what can be symbolically inferred: `isFruit`(APPLE), APPLE $\sim$ ORGANGE, `isFruit`(ORANGE)?
- Hard to work with language, vision and other modalities ``` ``is a film based on the novel of the same name by''(X, Y)

Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  $\texttt{grandpaOf} \neq \texttt{grandfatherOf}$
- No notion of similarity: $\textsc{apple} \sim \textsc{orange}$, $\texttt{professorAt} \sim \texttt{lecturerAt}$
- No generalization beyond what can be symbolically inferred:
  $\texttt{isFruit}(\textsc{apple})$, $\textsc{apple} \sim \textsc{organge}$, $\texttt{isFruit}(\textsc{orange})$?
- Hard to work with language, vision and other modalities
  $\texttt{``is a film based on the novel of the same name by''}(\mathrm{X}, \mathrm{Y})$
- But... leads to powerful inference mechanisms and proofs for predictions:
  $\texttt{fatherOf}(\textsc{abe}, \textsc{homer})$. $\texttt{parentOf}(\textsc{homer}, \textsc{lisa})$. $\texttt{parentOf}(\textsc{homer}, \textsc{bart})$.
  $\texttt{grandfatherOf}(\mathrm{X}, \mathrm{Y}) :\!- \texttt{fatherOf}(\mathrm{X}, \mathrm{Z}), \texttt{parentOf}(\mathrm{Z}, \mathrm{Y})$.
  $\texttt{grandfatherOf}(\textsc{abe}, \mathrm{Q})$? $\{\mathrm{Q}/\textsc{lisa}\}, \{\mathrm{Q}/\textsc{bart}\}$

Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  $\texttt{grandpaOf} \neq \texttt{grandfatherOf}$
- No notion of similarity: $\textsc{apple} \sim \textsc{orange}$, $\texttt{professorAt} \sim \texttt{lecturerAt}$
- No generalization beyond what can be symbolically inferred:
  $\texttt{isFruit}(\textsc{apple})$, $\textsc{apple} \sim \textsc{organge}$, $\texttt{isFruit}(\textsc{orange})$?
- Hard to work with language, vision and other modalities
  $\texttt{``is a film based on the novel of the same name by''}(\mathrm{X}, \mathrm{Y})$
- But... leads to powerful inference mechanisms and proofs for predictions:
  $\texttt{fatherOf}(\textsc{abe}, \textsc{homer})$. $\texttt{parentOf}(\textsc{homer}, \textsc{lisa})$. $\texttt{parentOf}(\textsc{homer}, \textsc{bart})$.
  $\texttt{grandfatherOf}(\mathrm{X}, \mathrm{Y}) \texttt{:-} \texttt{fatherOf}(\mathrm{X}, \mathrm{Z}), \texttt{parentOf}(\mathrm{Z}, \mathrm{Y})$.
  $\texttt{grandfatherOf}(\textsc{abe}, \mathrm{Q})$? $\{\mathrm{Q}/\textsc{lisa}\}, \{\mathrm{Q}/\textsc{bart}\}$
- Fairly easy to debug and trivial to incorporate domain knowledge:
  Show to domain expert and let her change/add rules and facts

Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  $\boldsymbol{v}_{\text{APPLE}}, \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\texttt{fatherOf}}, \ldots \in \mathbb{R}^k$

Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  $\boldsymbol{v}_{\text{APPLE}}, \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\texttt{fatherOf}}, \ldots \in \mathbb{R}^k$
- Can capture similarity and even semantic hierarchy of symbols:
  $\boldsymbol{v}_{\texttt{grandpaOf}} = \boldsymbol{v}_{\texttt{grandfatherOf}}$,
  $\boldsymbol{v}_{\text{APPLE}} \sim \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\text{APPLE}} < \boldsymbol{v}_{\text{FRUIT}}$

Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  $\boldsymbol{v}_{\text{APPLE}}, \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\texttt{fatherOf}}, \ldots \in \mathbb{R}^k$
- Can capture similarity and even semantic hierarchy of symbols:
  $\boldsymbol{v}_{\texttt{grandpaOf}} = \boldsymbol{v}_{\texttt{grandfatherOf}}$,
  $\boldsymbol{v}_{\text{APPLE}} \sim \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\text{APPLE}} < \boldsymbol{v}_{\text{FRUIT}}$
- Can be trained from raw task data (e.g. facts in a knowledge base)

Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  $\boldsymbol{v}_{\text{APPLE}}, \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\texttt{fatherOf}}, \ldots \in \mathbb{R}^k$
- Can capture similarity and even semantic hierarchy of symbols:
  $\boldsymbol{v}_{\texttt{grandpaOf}} = \boldsymbol{v}_{\texttt{grandfatherOf}}$,
  $\boldsymbol{v}_{\text{APPLE}} \sim \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\text{APPLE}} < \boldsymbol{v}_{\text{FRUIT}}$
- Can be trained from raw task data (e.g. facts in a knowledge base)
- Can be compositional
  $\boldsymbol{v}_{\texttt{``is the father of''}} = \text{RNN}_\theta(\boldsymbol{v}_{\texttt{is}}, \boldsymbol{v}_{\texttt{the}}, \boldsymbol{v}_{\texttt{father}}, \boldsymbol{v}_{\texttt{of}})$

Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  $\boldsymbol{v}_{\text{APPLE}}, \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\texttt{fatherOf}}, \ldots \in \mathbb{R}^k$
- Can capture similarity and even semantic hierarchy of symbols:
  $\boldsymbol{v}_{\texttt{grandpaOf}} = \boldsymbol{v}_{\texttt{grandfatherOf}}$,
  $\boldsymbol{v}_{\text{APPLE}} \sim \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\text{APPLE}} < \boldsymbol{v}_{\text{FRUIT}}$
- Can be trained from raw task data (e.g. facts in a knowledge base)
- Can be compositional
  $\boldsymbol{v}_{\texttt{``is the father of''}} = \text{RNN}_\theta(\boldsymbol{v}_{\texttt{is}}, \boldsymbol{v}_{\texttt{the}}, \boldsymbol{v}_{\texttt{father}}, \boldsymbol{v}_{\texttt{of}})$
- But... need large amount of training data

Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  $\boldsymbol{v}_{\text{APPLE}}, \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\texttt{fatherOf}}, \ldots \in \mathbb{R}^k$
- Can capture similarity and even semantic hierarchy of symbols:
  $\boldsymbol{v}_{\texttt{grandpaOf}} = \boldsymbol{v}_{\texttt{grandfatherOf}}$,
  $\boldsymbol{v}_{\text{APPLE}} \sim \boldsymbol{v}_{\text{ORANGE}}, \boldsymbol{v}_{\text{APPLE}} < \boldsymbol{v}_{\text{FRUIT}}$
- Can be trained from raw task data (e.g. facts in a knowledge base)
- Can be compositional
  $\boldsymbol{v}_{\texttt{``is the father of''}} = \text{RNN}_\theta(\boldsymbol{v}_{\texttt{is}}, \boldsymbol{v}_{\texttt{the}}, \boldsymbol{v}_{\texttt{father}}, \boldsymbol{v}_{\texttt{of}})$
- But... need large amount of training data
- No direct way of incorporating prior knowledge
  $\boldsymbol{v}_{\texttt{grandfatherOf}}(\text{X}, \text{Y}) \coloneqq \boldsymbol{v}_{\texttt{fatherOf}}(\text{X}, \text{Z}), \boldsymbol{v}_{\texttt{parentOf}}(\text{Z}, \text{Y}).$

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Plotkin (1970), Shapiro (1991), Muggleton (1991), De Raedt (1999) ...

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Plotkin (1970), Shapiro (1991), Muggleton (1991), De Raedt (1999) ...
  - **Statistical Predicate Invention** (Kok and Domingos, 2007)

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Plotkin (1970), Shapiro (1991), Muggleton (1991), De Raedt (1999) ...
  - **Statistical Predicate Invention** (Kok and Domingos, 2007)
- Neural-symbolic Connectionism

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Plotkin (1970), Shapiro (1991), Muggleton (1991), De Raedt (1999) ...
  - **Statistical Predicate Invention** (Kok and Domingos, 2007)
- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (Garcez and Zaverucha, 1999)

Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), **Markov Logic Networks** (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Plotkin (1970), Shapiro (1991), Muggleton (1991), De Raedt (1999) ...
  - **Statistical Predicate Invention** (Kok and Domingos, 2007)
- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (Garcez and Zaverucha, 1999)
  - First-order inference (no training of symbol representations): **Unification Neural Networks** (Holldöbler, 1990; Komendantskaya 2011), SHRUTI (Shastri, 1992), **Neural Prolog** (Ding, 1995), CLIP++ (Franca et al. 2014), Lifted Relational Networks (Sourek et al. 2015)

State-of-the-art Neural Link Prediction

$$\texttt{livesIn}(\text{MELINDA}, \text{SEATTLE})? = f(\boldsymbol{v}_{\texttt{livesIn}}, \boldsymbol{v}_{\text{MELINDA}}, \boldsymbol{v}_{\text{SEATTLE}})$$

State-of-the-art Neural Link Prediction

$$\texttt{livesIn}(\text{MELINDA}, \text{SEATTLE})? = f(\boldsymbol{v}_{\texttt{livesIn}}, \boldsymbol{v}_{\text{MELINDA}}, \boldsymbol{v}_{\text{SEATTLE}})$$

DistMult (Yang et al., 2015)
$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^k$

State-of-the-art Neural Link Prediction

$$\texttt{livesIn}(\text{MELINDA}, \text{SEATTLE})? = f(\boldsymbol{v}_{\texttt{livesIn}}, \boldsymbol{v}_{\text{MELINDA}}, \boldsymbol{v}_{\text{SEATTLE}})$$

DistMult (Yang et al., 2015)

$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^k$

$$\begin{aligned} f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j) &= \boldsymbol{v}_s^\top (\boldsymbol{v}_i \odot \boldsymbol{v}_j) \\ &= \sum_k \boldsymbol{v}_{sk} \boldsymbol{v}_{ik} \boldsymbol{v}_{jk} \end{aligned}$$

State-of-the-art Neural Link Prediction

$$\texttt{livesIn}(\text{MELINDA}, \text{SEATTLE})? = f(\boldsymbol{v}_{\texttt{livesIn}}, \boldsymbol{v}_{\text{MELINDA}}, \boldsymbol{v}_{\text{SEATTLE}})$$

DistMult (Yang et al., 2015)
$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^k$

$$\begin{aligned} f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j) &= \boldsymbol{v}_s^\top (\boldsymbol{v}_i \odot \boldsymbol{v}_j) \\ &= \sum_k \boldsymbol{v}_{sk} \boldsymbol{v}_{ik} \boldsymbol{v}_{jk} \end{aligned}$$

ComplEx (Trouillon et al., 2016)
$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{C}^k$

State-of-the-art Neural Link Prediction

$$\texttt{livesIn}(\text{MELINDA}, \text{SEATTLE})? = f(\boldsymbol{v}_{\texttt{livesIn}}, \boldsymbol{v}_{\text{MELINDA}}, \boldsymbol{v}_{\text{SEATTLE}})$$

DistMult (Yang et al., 2015)

$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^k$

$$\begin{aligned} f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j) &= \boldsymbol{v}_s^\top (\boldsymbol{v}_i \odot \boldsymbol{v}_j) \\ &= \sum_k \boldsymbol{v}_{sk} \boldsymbol{v}_{ik} \boldsymbol{v}_{jk} \end{aligned}$$

ComplEx (Trouillon et al., 2016)

$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{C}^k$

$$\begin{aligned} f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j) = \\ & \text{real}(\boldsymbol{v}_s)^\top (\text{real}(\boldsymbol{v}_i) \odot \text{real}(\boldsymbol{v}_j)) \\ & + \text{real}(\boldsymbol{v}_s)^\top (\text{imag}(\boldsymbol{v}_i) \odot \text{imag}(\boldsymbol{v}_j)) \\ & + \text{imag}(\boldsymbol{v}_s)^\top (\text{real}(\boldsymbol{v}_i) \odot \text{imag}(\boldsymbol{v}_j)) \\ & - \text{imag}(\boldsymbol{v}_s)^\top (\text{imag}(\boldsymbol{v}_i) \odot \text{real}(\boldsymbol{v}_j)) \end{aligned}$$

State-of-the-art Neural Link Prediction

$$\texttt{livesIn}(\text{MELINDA}, \text{SEATTLE})? = f(\boldsymbol{v}_{\texttt{livesIn}}, \boldsymbol{v}_{\text{MELINDA}}, \boldsymbol{v}_{\text{SEATTLE}})$$

DistMult (Yang et al., 2015)

$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^k$

$$\begin{aligned} f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j) &= \boldsymbol{v}_s^\top (\boldsymbol{v}_i \odot \boldsymbol{v}_j) \\ &= \sum_k \boldsymbol{v}_{sk} \boldsymbol{v}_{ik} \boldsymbol{v}_{jk} \end{aligned}$$

ComplEx (Trouillon et al., 2016)

$\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{C}^k$

$$\begin{aligned} f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j) = \\ & \text{real}(\boldsymbol{v}_s)^\top (\text{real}(\boldsymbol{v}_i) \odot \text{real}(\boldsymbol{v}_j)) \\ & + \text{real}(\boldsymbol{v}_s)^\top (\text{imag}(\boldsymbol{v}_i) \odot \text{imag}(\boldsymbol{v}_j)) \\ & + \text{imag}(\boldsymbol{v}_s)^\top (\text{real}(\boldsymbol{v}_i) \odot \text{imag}(\boldsymbol{v}_j)) \\ & - \text{imag}(\boldsymbol{v}_s)^\top (\text{imag}(\boldsymbol{v}_i) \odot \text{real}(\boldsymbol{v}_j)) \end{aligned}$$

Training Loss

$$\mathfrak{L} = \sum_{r_s(e_i, e_j), y \in \mathcal{T}} -y \log(\sigma(f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j))) - (1-y) \log(1 - \sigma(f(\boldsymbol{v}_s, \boldsymbol{v}_i, \boldsymbol{v}_j)))$$

Differentiable Proving in a Nutshell

grandfatherOf ABE BART

Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Differentiable Proving in a Nutshell

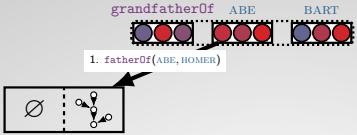
Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Differentiable Proving in a Nutshell

Example Knowledge Base:

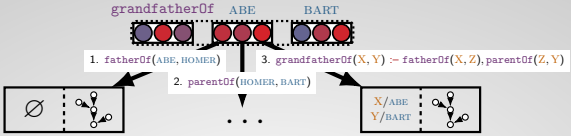
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

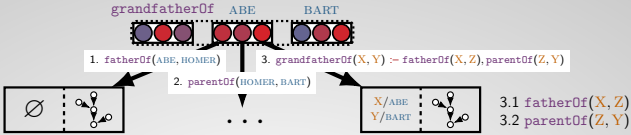
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

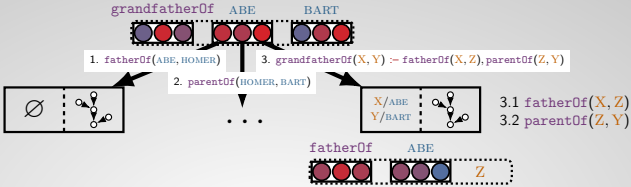
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

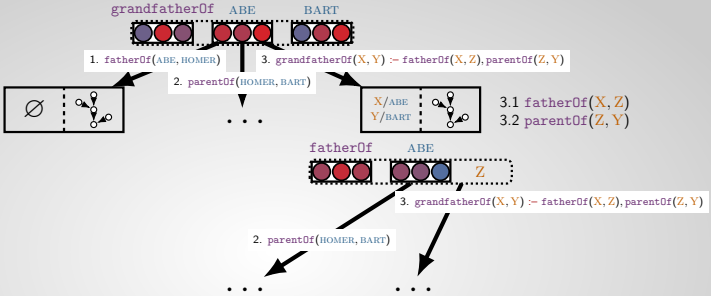
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :- fatherOf(X, Z), parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

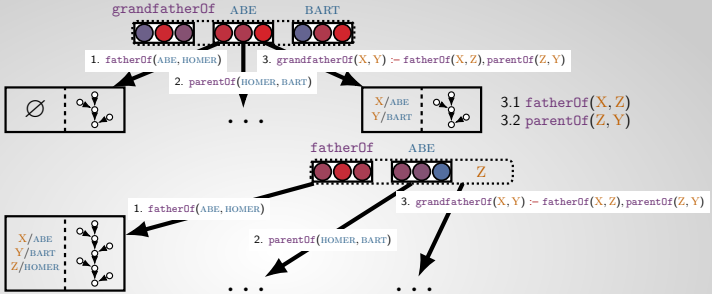
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

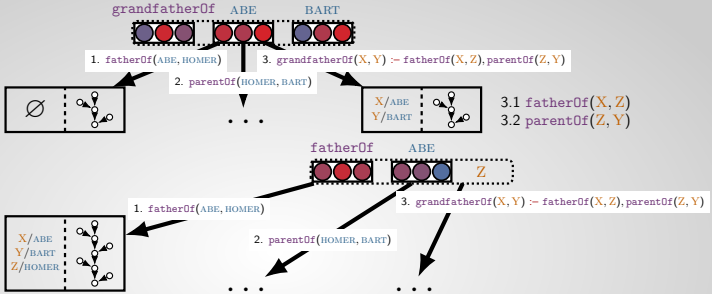
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

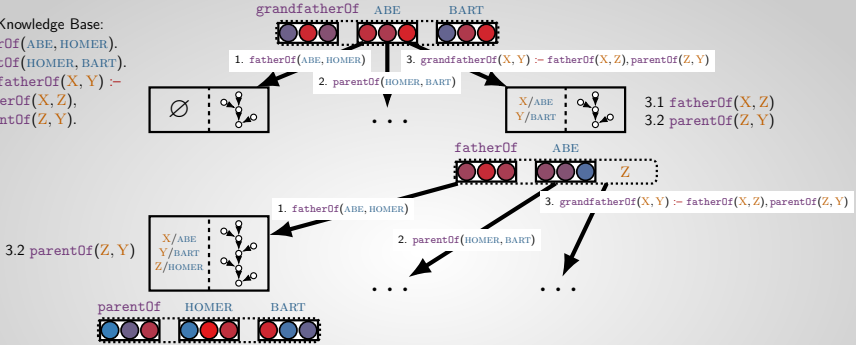
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

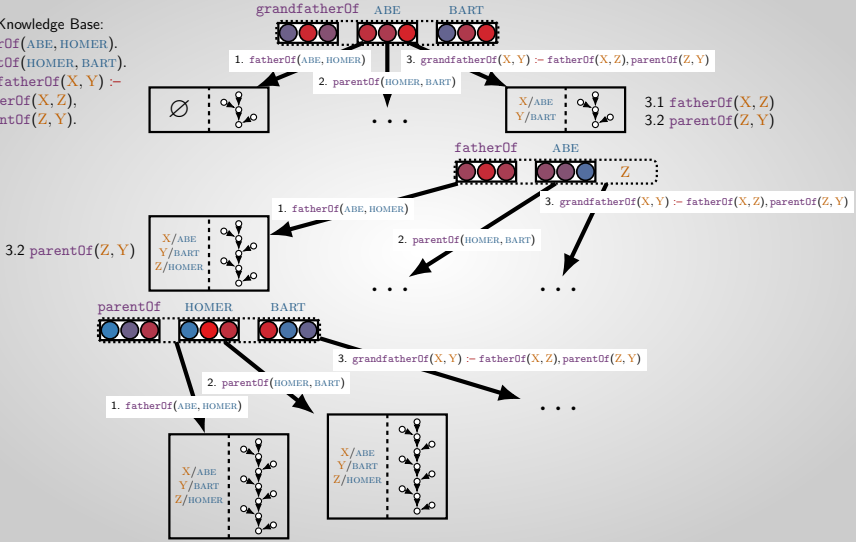
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Differentiable Proving in a Nutshell

Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).



Proof States

$$S = (\Psi, \rho)$$

- Substitution set Ψ constructed in the proof so far

Proof States

$$S = (\Psi, \rho)$$

- Substitution set Ψ constructed in the proof so far
- Neural network ρ that outputs a real-valued proof success score

Proof States

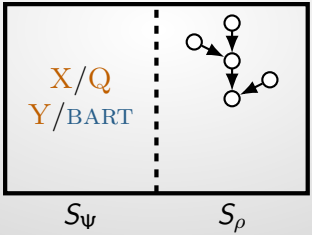
$$S = (\Psi, \rho)$$

- Substitution set Ψ constructed in the proof so far
- Neural network ρ that outputs a real-valued proof success score

Proof States

$$S = (\Psi, \rho)$$

- Substitution set Ψ constructed in the proof so far
- Neural network ρ that outputs a real-valued proof success score



Proof Modules

$$\texttt{unify}_{\boldsymbol{\theta}}, \texttt{or}_{\boldsymbol{\theta}}^{\hat{\mathfrak{K}}}, \texttt{and}_{\boldsymbol{\theta}}^{\hat{\mathfrak{K}}}$$

- Modular construction of differentiable prover

Proof Modules

$$\texttt{unify}_{\boldsymbol{\theta}}, \texttt{or}_{\boldsymbol{\theta}}^{\hat{\kappa}}, \texttt{and}_{\boldsymbol{\theta}}^{\hat{\kappa}}$$

- Modular construction of differentiable prover
- Discrete objects (rules, atoms etc.) are used to instantiate proof modules

Proof Modules

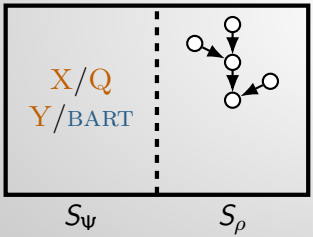
$$\texttt{unify}_{\boldsymbol{\theta}}, \texttt{or}_{\boldsymbol{\theta}}^{\hat{\mathcal{R}}}, \texttt{and}_{\boldsymbol{\theta}}^{\hat{\mathcal{R}}}$$

- Modular construction of differentiable prover
- Discrete objects (rules, atoms etc.) are used to instantiate proof modules
- **Modules** transform proof states into new proof states

Proof Modules

$$\mathtt{unify}_{\boldsymbol{\theta}}, \mathtt{or}_{\boldsymbol{\theta}}^{\hat{\mathfrak{K}}}, \mathtt{and}_{\boldsymbol{\theta}}^{\hat{\mathfrak{K}}}$$

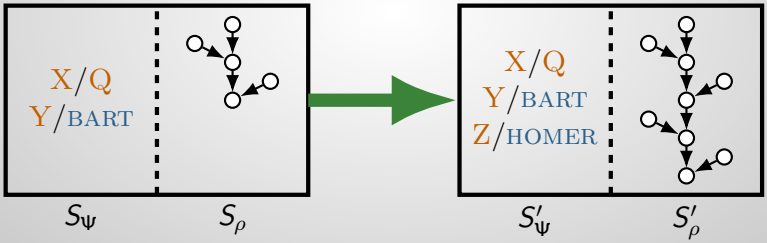
- Modular construction of differentiable prover
- Discrete objects (rules, atoms etc.) are used to instantiate proof modules
- **Modules** transform proof states into new proof states



Proof Modules

$$\texttt{unify}_{\boldsymbol{\theta}}, \texttt{or}_{\boldsymbol{\theta}}^{\hat{\kappa}}, \texttt{and}_{\boldsymbol{\theta}}^{\hat{\kappa}}$$

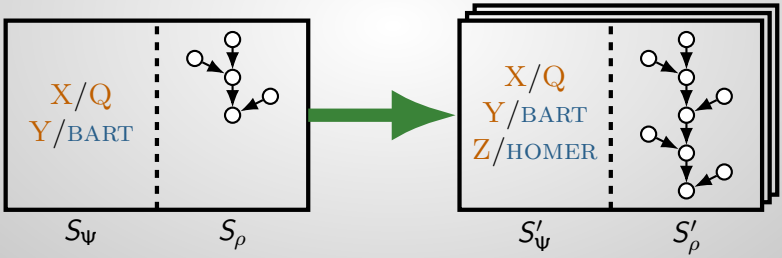
- Modular construction of differentiable prover
- Discrete objects (rules, atoms etc.) are used to instantiate proof modules
- **Modules** transform proof states into new proof states



Proof Modules

$$\texttt{unify}_{\boldsymbol{\theta}}, \texttt{or}_{\boldsymbol{\theta}}^{\hat{\mathfrak{K}}}, \texttt{and}_{\boldsymbol{\theta}}^{\hat{\mathfrak{K}}}$$

- Modular construction of differentiable prover
- Discrete objects (rules, atoms etc.) are used to instantiate proof modules
- **Modules** transform proof states into new proof states



Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

1. $\texttt{unify}_\theta([\,], [\,], S) = S$

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

1. $\texttt{unify}_\theta([\,], [\,], S) = S$
2. $\texttt{unify}_\theta([\,], \boldsymbol{G}, S) = \texttt{FAIL}$

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

1. $\texttt{unify}_\theta([\,], [\,], S) = S$
2. $\texttt{unify}_\theta([\,], \boldsymbol{G}, S) = \texttt{FAIL}$
3. $\texttt{unify}_\theta(\boldsymbol{H}, [\,], S) = \texttt{FAIL}$

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

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

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

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

$$S'_\Psi = S_\Psi \cup \left\{ \begin{array}{ll} \{h/g\} & \text{if } h \in \mathcal{V} \\ \{g/h\} & \text{if } g \in \mathcal{V}, h \notin \mathcal{V} \\ \varnothing & \text{otherwise} \end{array} \right\}$$

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

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

$$S'_\Psi = S_\Psi \cup \left\{ \begin{array}{ll} \{h/g\} & \text{if } h \in \mathcal{V} \\ \{g/h\} & \text{if } g \in \mathcal{V}, h \notin \mathcal{V} \\ \varnothing & \text{otherwise} \end{array} \right\}$$

$$S'_\rho = \min\left(S_\rho, \left\{ \begin{array}{ll} \exp\left(-\|\boldsymbol{\theta}_{h:} - \boldsymbol{\theta}_{g:}\|_2\right) & \text{if } h \notin \mathcal{V}, g \notin \mathcal{V} \\ 1 & \text{otherwise} \end{array} \right\}\right)$$

Unification Module

`unify` takes two atoms represented as lists of terms and an upstream proof state, and maps these to a new proof state (substitution set and proof success)

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

$$S'_{\Psi} = S_{\Psi} \cup \left\{ \begin{array}{ll} \{h/g\} & \text{if } h \in \mathcal{V} \\ \{g/h\} & \text{if } g \in \mathcal{V}, h \notin \mathcal{V} \\ \varnothing & \text{otherwise} \end{array} \right\}$$

$$S'_{\rho} = \min\left(S_{\rho}, \left\{ \begin{array}{ll} \exp\left(-\|\boldsymbol{\theta}_{h:} - \boldsymbol{\theta}_{g:}\|_2\right) & \text{if } h \notin \mathcal{V}, g \notin \mathcal{V} \\ 1 & \text{otherwise} \end{array} \right\}\right)$$

Example:

$$\texttt{unify}_{\boldsymbol{\theta}}([\texttt{grandpaOf}, \text{ABE}, \text{BART}], [s, \text{Q}, i], (\varnothing, 0.7)) =$$
$$\left(\{\text{Q}/\text{ABE}\}, \min(0.7, \exp(-\|\boldsymbol{\theta}_{\texttt{grandpaOf}:} - \boldsymbol{\theta}_{s:}\|_2), \exp(-\|\boldsymbol{\theta}_{\text{BART}:} - \boldsymbol{\theta}_{i:}\|_2))\right)$$

OR Module

1. $\mathtt{or}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d, S) = [S' \mid S' \in \mathtt{and}_{\theta}^{\mathfrak{K}}(\boldsymbol{B}, d, \mathtt{unify}_{\theta}(\boldsymbol{H}, \boldsymbol{G}, S)), \boldsymbol{H} :\!- \boldsymbol{B} \in \mathfrak{K}]$
   - $\boldsymbol{G}$ is a goal atom, d is the maximum proof depth, and $\boldsymbol{H} :\!- \boldsymbol{B}$ is a rule

OR Module

1. $\mathtt{or}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d, S) = [S' \mid S' \in \mathtt{and}_{\theta}^{\mathfrak{K}}(\boldsymbol{B}, d, \mathtt{unify}_{\theta}(\boldsymbol{H}, \boldsymbol{G}, S)), \boldsymbol{H} :\!- \boldsymbol{B} \in \mathfrak{K}]$
   - $\boldsymbol{G}$ is a goal atom, d is the maximum proof depth, and $\boldsymbol{H} :\!- \boldsymbol{B}$ is a rule
   - `or` iterates through all rules (including rules with an empty body, *i.e.*, facts) and unifies the goal with the respective rule head

OR Module

1. $\mathtt{or}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d, S) = [S' \mid S' \in \mathtt{and}_{\theta}^{\mathfrak{K}}(\boldsymbol{B}, d, \mathtt{unify}_{\theta}(\boldsymbol{H}, \boldsymbol{G}, S)), \boldsymbol{H} :\!- \boldsymbol{B} \in \mathfrak{K}]$
   - $\boldsymbol{G}$ is a goal atom, d is the maximum proof depth, and $\boldsymbol{H} :\!- \boldsymbol{B}$ is a rule
   - `or` iterates through all rules (including rules with an empty body, *i.e.*, facts) and unifies the goal with the respective rule head
   - If unification succeeds, it instantiates an `and` module to prove all atoms in the body of the rule.

OR Module

1. $\mathtt{or}_\theta^{\mathfrak{K}}(\boldsymbol{G}, d, S) = [S' \mid S' \in \mathtt{and}_\theta^{\mathfrak{K}}(\boldsymbol{B}, d, \mathtt{unify}_\theta(\boldsymbol{H}, \boldsymbol{G}, S)), \boldsymbol{H} \text{ :– } \boldsymbol{B} \in \mathfrak{K}]$
   - $\boldsymbol{G}$ is a goal atom, d is the maximum proof depth, and $\boldsymbol{H}$:– $\boldsymbol{B}$ is a rule
   - or iterates through all rules (including rules with an empty body, *i.e.*, facts) and unifies the goal with the respective rule head
   - If unification succeeds, it instantiates an and module to prove all atoms in the body of the rule.
   - In other words, it is translating goals into subgoals using rules, *e.g.*, $\mathtt{grandfatherOf}(\mathrm{Q}, \mathrm{BART})$ is translated into subgoals $\mathtt{fatherOf}(\mathrm{Q}, \mathrm{Z})$ and $\mathtt{parentOf}(\mathrm{Z}, \mathrm{BART})$ using the rule $\mathtt{grandfatherOf}(\mathrm{X}, \mathrm{Y}) \text{ :– } \mathtt{fatherOf}(\mathrm{X}, \mathrm{Z}), \mathtt{parentOf}(\mathrm{Z}, \mathrm{Y})$

OR Module

1. $\mathtt{or}_\theta^{\mathfrak{K}}(\boldsymbol{G}, d, S) = [S' \mid S' \in \mathtt{and}_\theta^{\mathfrak{K}}(\boldsymbol{B}, d, \mathtt{unify}_\theta(\boldsymbol{H}, \boldsymbol{G}, S)), \boldsymbol{H} :\!- \boldsymbol{B} \in \mathfrak{K}]$
   - $\boldsymbol{G}$ is a goal atom, d is the maximum proof depth, and $\boldsymbol{H} :\!- \boldsymbol{B}$ is a rule
   - or iterates through all rules (including rules with an empty body, *i.e.*, facts) and unifies the goal with the respective rule head
   - If unification succeeds, it instantiates an and module to prove all atoms in the body of the rule.
   - In other words, it is translating goals into subgoals using rules, *e.g.*, $\mathtt{grandfatherOf}(\mathrm{Q}, \mathrm{BART})$ is translated into subgoals $\mathtt{fatherOf}(\mathrm{Q}, \mathrm{Z})$ and $\mathtt{parentOf}(\mathrm{Z}, \mathrm{BART})$ using the rule $\mathtt{grandfatherOf}(\mathrm{X}, \mathrm{Y}) :\!- \mathtt{fatherOf}(\mathrm{X}, \mathrm{Z}), \mathtt{parentOf}(\mathrm{Z}, \mathrm{Y})$

Example:

$$\mathtt{or}_\theta^{\mathfrak{K}}([\mathtt{grandfatherOf}, \mathrm{Q}, \mathrm{BART}], d, S) =$$

$$[S' | S' \in \mathtt{and}_\theta^{\mathfrak{K}}([[\mathtt{fatherOf}, \mathrm{X}, \mathrm{Z}], [\mathtt{parentOf}, \mathrm{Z}, \mathrm{Y}]], d, \underbrace{(\{\mathrm{X}/\mathrm{Q}, \mathrm{Y}/\mathrm{BART}\}, \hat{S}_\rho)}_{\text{result of } \mathtt{unify}_\theta}), \ldots]$$

AND Module

1. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, d, \mathtt{FAIL}) = \mathtt{FAIL}$

AND Module

1. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, d, \mathtt{FAIL}) = \mathtt{FAIL}$
2. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, 0, S) = \mathtt{FAIL}$

AND Module

1. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, d, \mathtt{FAIL}) = \mathtt{FAIL}$
2. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, 0, S) = \mathtt{FAIL}$
3. $\mathtt{and}_{\theta}^{\mathfrak{K}}([\,], d, S) = S$

AND Module

1. $\mathtt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathcal{G}, d, \mathtt{FAIL}) = \mathtt{FAIL}$
2. $\mathtt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathcal{G}, 0, S) = \mathtt{FAIL}$
3. $\mathtt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}([\,], d, S) = S$
4. $\mathtt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\boldsymbol{G} :: \mathcal{G}, d, S) = [S'' \mid S'' \in \mathtt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathcal{G}, d, S'), S' \in \mathtt{or}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\text{substitute}(\boldsymbol{G}, S), d-1, S)]$

AND Module

1. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, d, \mathtt{FAIL}) = \mathtt{FAIL}$
2. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, 0, S) = \mathtt{FAIL}$
3. $\mathtt{and}_{\theta}^{\mathfrak{K}}([\,], d, S) = S$
4. $\mathtt{and}_{\theta}^{\mathfrak{K}}(\boldsymbol{G} :: \mathcal{G}, d, S) = [S'' \mid S'' \in \mathtt{and}_{\theta}^{\mathfrak{K}}(\mathcal{G}, d, S'), S' \in \mathtt{or}_{\theta}^{\mathfrak{K}}(\text{substitute}(\boldsymbol{G}, S), d-1, S)]$

Example:

$$\mathtt{and}_{\theta}^{\mathfrak{K}}([[\mathtt{fatherOf}, \mathrm{X}, \mathrm{Z}], [\mathtt{parentOf}, \mathrm{Z}, \mathrm{Y}]], d, \underbrace{(\{\mathrm{X}/\mathrm{Q}, \mathrm{Y}/\mathrm{BART}\}, \hat{S}_{\rho})}_{\text{result of } \mathtt{unify}_{\theta} \text{ in } \mathtt{or}_{\theta}^{\mathfrak{K}}}) =$$

$$[S'' | S'' \in \mathtt{and}_{\theta}^{\mathfrak{K}}([[\mathtt{parentOf}, \mathrm{Z}, \mathrm{Y}]], d, S'), S' \in \mathtt{or}_{\theta}^{\mathfrak{K}}(\underbrace{[\mathtt{fatherOf}, \mathrm{Q}, \mathrm{Z}]}_{\text{result of substitute}}, d-1, S)]$$

Proof Aggregation

- Goal $\boldsymbol{G} = [s, i, j]$ where s is the index of a predicate symbol and i, j are indices of constant symbols
- d maximum proof depth and proof start state $(\varnothing, 1)$

$$\mathtt{ntp}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d) = \max_{\substack{S \in \mathtt{or}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d, (\varnothing, 1)) \\ S \neq \mathtt{FAIL}}} S_{\rho}$$

Proof Aggregation

- Goal $\boldsymbol{G} = [s, i, j]$ where s is the index of a predicate symbol and i, j are indices of constant symbols
- d maximum proof depth and proof start state $(\varnothing, 1)$

$$\mathtt{ntp}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d) = \max_{\substack{S \in \mathtt{or}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d, (\varnothing, 1)) \\ S \neq \mathtt{FAIL}}} S_{\rho}$$

Training Loss

$$\mathcal{L}_{\mathtt{ntp}_{\theta}^{\mathfrak{K}}} = \sum_{(\boldsymbol{G}, y) \in \mathcal{T}} -y \log(\mathtt{ntp}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d)) - (1 - y) \log(1 - \mathtt{ntp}_{\theta}^{\mathfrak{K}}(\boldsymbol{G}, d))$$

Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

$\mathrm{or}_\theta^{\mathbb{R}}([s, i, j], 2, (\varnothing, 1))$

Example Knowledge Base:

1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER})$.
2. $\texttt{parentOf}(\text{HOMER}, \text{BART})$.
3. $\texttt{grandfatherOf}(\text{X}, \text{Y}) :-$
   $\texttt{fatherOf}(\text{X}, \text{Z}),$
   $\texttt{parentOf}(\text{Z}, \text{Y}).$

$\texttt{or}^{\mathrm{R}}_{\theta}([s, i, j], 2, (\varnothing, 1))$

1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [s, i, j], (\varnothing, 1))$
2. $\cdots$
3. $\texttt{unify}_{\theta}([\texttt{grandfatherOf}, \text{X}, \text{Y}], [s, i, j], (\varnothing, 1))$

$S_1 = (\varnothing, \rho_1)$

$S_2 = (\varnothing, \rho_2)$

Example Knowledge Base:

1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER})$.
2. $\texttt{parentOf}(\text{HOMER}, \text{BART})$.
3. $\texttt{grandfatherOf}(\text{X}, \text{Y}) :-$
   $\texttt{fatherOf}(\text{X}, \text{Z}),$
   $\texttt{parentOf}(\text{Z}, \text{Y})$.

$\texttt{or}^{\mathbb{R}}_{\theta}([s, i, j], 2, (\varnothing, 1))$

1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [s, i, j], (\varnothing, 1))$
2. $\cdots$
3. $\texttt{unify}_{\theta}([\texttt{grandfatherOf}, \text{X}, \text{Y}], [s, i, j], (\varnothing, 1))$

$S_1 = (\varnothing, \rho_1)$

$S_2 = (\varnothing, \rho_2)$

$S_3 = (\{\text{X}/i, \text{Y}/j\}, \rho_3)$

$\texttt{and}^{\mathbb{R}}_{\theta}([[\texttt{fatherOf}, \text{X}, \text{Z}], [\texttt{parentOf}, \text{Z}, \text{Y}]], 2, S_3)$

Example Knowledge Base:

1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER})$.
2. $\texttt{parentOf}(\text{HOMER}, \text{BART})$.
3. $\texttt{grandfatherOf}(\text{X}, \text{Y}) :-$
   $\texttt{fatherOf}(\text{X}, \text{Z}),$
   $\texttt{parentOf}(\text{Z}, \text{Y})$.

$\texttt{or}^{\mathbb{R}}_{\theta}([s, i, j], 2, (\varnothing, 1))$

1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [s, i, j], (\varnothing, 1))$
2. $\cdots$
3. $\texttt{unify}_{\theta}([\texttt{grandfatherOf}, \text{X}, \text{Y}], [s, i, j], (\varnothing, 1))$

$S_1 = (\varnothing, \rho_1)$ $S_2 = (\varnothing, \rho_2)$ $S_3 = (\{\text{X}/i, \text{Y}/j\}, \rho_3)$

$\texttt{and}^{\mathbb{R}}_{\theta}([[\texttt{fatherOf}, \text{X}, \text{Z}], [\texttt{parentOf}, \text{Z}, \text{Y}]], 2, S_3)$

substitute

$\texttt{or}^{\mathbb{R}}_{\theta}([\texttt{fatherOf}, i, \text{Z}], 1, S_3)$

Example Knowledge Base:

1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER})$.
2. $\texttt{parentOf}(\text{HOMER}, \text{BART})$.
3. $\texttt{grandfatherOf}(\text{X}, \text{Y}) :-$
   $\texttt{fatherOf}(\text{X}, \text{Z}),$
   $\texttt{parentOf}(\text{Z}, \text{Y}).$

$\texttt{or}^{\mathbb{R}}_{\theta}([s, i, j], 2, (\varnothing, 1))$

1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [s, i, j], (\varnothing, 1))$ → $S_1 = (\varnothing, \rho_1)$
2. $\cdots$ → $S_2 = (\varnothing, \rho_2)$
3. $\texttt{unify}_{\theta}([\texttt{grandfatherOf}, \text{X}, \text{Y}], [s, i, j], (\varnothing, 1))$ → $S_3 = (\{\text{X}/i, \text{Y}/j\}, \rho_3)$

$\texttt{and}^{\mathbb{R}}_{\theta}([[\texttt{fatherOf}, \text{X}, \text{Z}], [\texttt{parentOf}, \text{Z}, \text{Y}]], 2, S_3)$

substitute

$\texttt{or}^{\mathbb{R}}_{\theta}([\texttt{fatherOf}, i, \text{Z}], 1, S_3)$

1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [\texttt{fatherOf}, i, \text{Z}], S_3)$
2. $\texttt{unify}_{\theta}([\texttt{parentOf}, \text{HOMER}, \text{BART}], [\texttt{fatherOf}, i, \text{Z}], S_3)$
3. $\cdots$ → $S_{33} = \texttt{FAIL}$

Example Knowledge Base:
1. $\mathtt{fatherOf}(\mathrm{ABE}, \mathrm{HOMER})$.
2. $\mathtt{parentOf}(\mathrm{HOMER}, \mathrm{BART})$.
3. $\mathtt{grandfatherOf}(\mathrm{X}, \mathrm{Y}) :-$
   $\mathtt{fatherOf}(\mathrm{X}, \mathrm{Z}),$
   $\mathtt{parentOf}(\mathrm{Z}, \mathrm{Y})$.

$$\mathtt{or}^{\mathrm{R}}_{\theta}([s, i, j], 2, (\varnothing, 1))$$

1. $\mathtt{unify}_{\theta}([\mathtt{fatherOf}, \mathrm{ABE}, \mathrm{HOMER}], [s, i, j], (\varnothing, 1))$ $\rightarrow$ $S_1 = (\varnothing, \rho_1)$
2. $\cdots$ $\rightarrow$ $S_2 = (\varnothing, \rho_2)$
3. $\mathtt{unify}_{\theta}([\mathtt{grandfatherOf}, \mathrm{X}, \mathrm{Y}], [s, i, j], (\varnothing, 1))$ $\rightarrow$ $S_3 = (\{\mathrm{X}/i, \mathrm{Y}/j\}, \rho_3)$

$$\mathtt{and}^{\mathrm{R}}_{\theta}([[\mathtt{fatherOf}, \mathrm{X}, \mathrm{Z}], [\mathtt{parentOf}, \mathrm{Z}, \mathrm{Y}]], 2, S_3)$$

substitute

$$\mathtt{or}^{\mathrm{R}}_{\theta}([\mathtt{fatherOf}, i, \mathrm{Z}], 1, S_3)$$

1. $\mathtt{unify}_{\theta}([\mathtt{fatherOf}, \mathrm{ABE}, \mathrm{HOMER}], [\mathtt{fatherOf}, i, \mathrm{Z}], S_3)$ $\rightarrow$ $S_{31} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{31})$
2. $\mathtt{unify}_{\theta}([\mathtt{parentOf}, \mathrm{HOMER}, \mathrm{BART}], [\mathtt{fatherOf}, i, \mathrm{Z}], S_3)$
3. $\cdots$ $\rightarrow$ $S_{33} = \mathtt{FAIL}$

$$\mathtt{and}^{\mathrm{R}}_{\theta}([\mathtt{parentOf}, \mathrm{Z}, \mathrm{Y}], 2, S_{31})$$

Example Knowledge Base:
1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER})$.
2. $\texttt{parentOf}(\text{HOMER}, \text{BART})$.
3. $\texttt{grandfatherOf}(\text{X}, \text{Y}) :-$
   $\texttt{fatherOf}(\text{X}, \text{Z}),$
   $\texttt{parentOf}(\text{Z}, \text{Y}).$

$\texttt{or}^{\mathbb{R}}_{\theta}([s, i, j], 2, (\varnothing, 1))$

1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [s, i, j], (\varnothing, 1))$ → $S_1 = (\varnothing, \rho_1)$
2. $\cdots$ → $S_2 = (\varnothing, \rho_2)$
3. $\texttt{unify}_{\theta}([\texttt{grandfatherOf}, \text{X}, \text{Y}], [s, i, j], (\varnothing, 1))$ → $S_3 = (\{\text{X}/i, \text{Y}/j\}, \rho_3)$

$\texttt{and}^{\mathbb{R}}_{\theta}([[\texttt{fatherOf}, \text{X}, \text{Z}], [\texttt{parentOf}, \text{Z}, \text{Y}]], 2, S_3)$

▼substitute

$\texttt{or}^{\mathbb{R}}_{\theta}([\texttt{fatherOf}, i, \text{Z}], 1, S_3)$

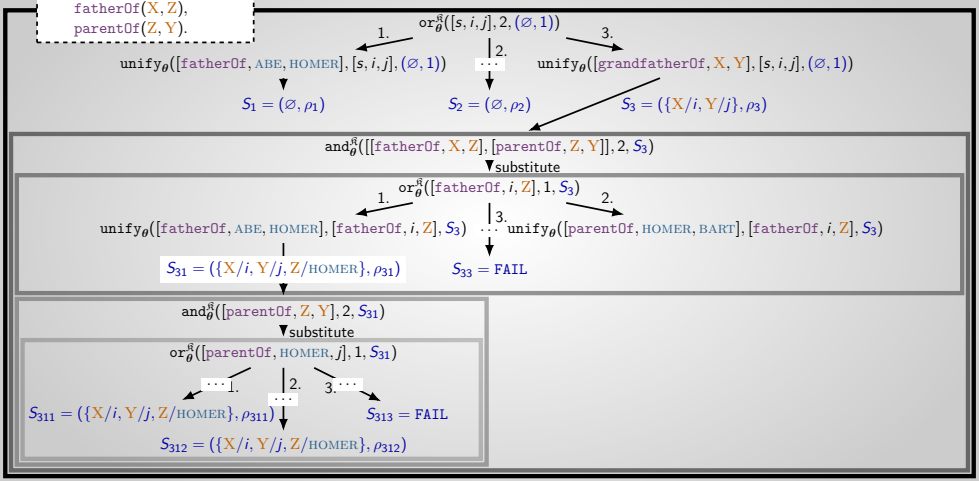
1. $\texttt{unify}_{\theta}([\texttt{fatherOf}, \text{ABE}, \text{HOMER}], [\texttt{fatherOf}, i, \text{Z}], S_3)$ → $S_{31} = (\{\text{X}/i, \text{Y}/j, \text{Z}/\text{HOMER}\}, \rho_{31})$
2. $\texttt{unify}_{\theta}([\texttt{parentOf}, \text{HOMER}, \text{BART}], [\texttt{fatherOf}, i, \text{Z}], S_3)$
3. $\cdots$ → $S_{33} = \texttt{FAIL}$

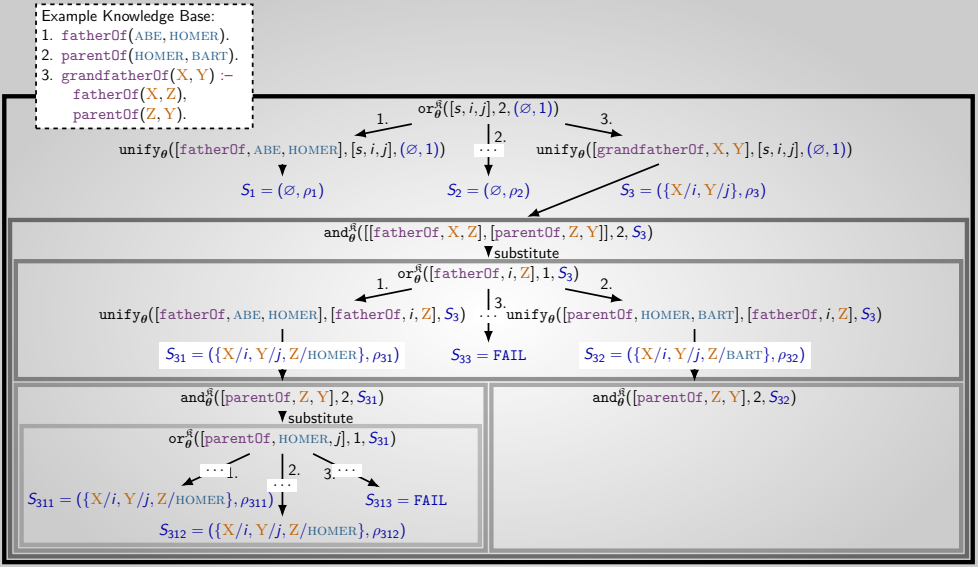
$\texttt{and}^{\mathbb{R}}_{\theta}([\texttt{parentOf}, \text{Z}, \text{Y}], 2, S_{31})$

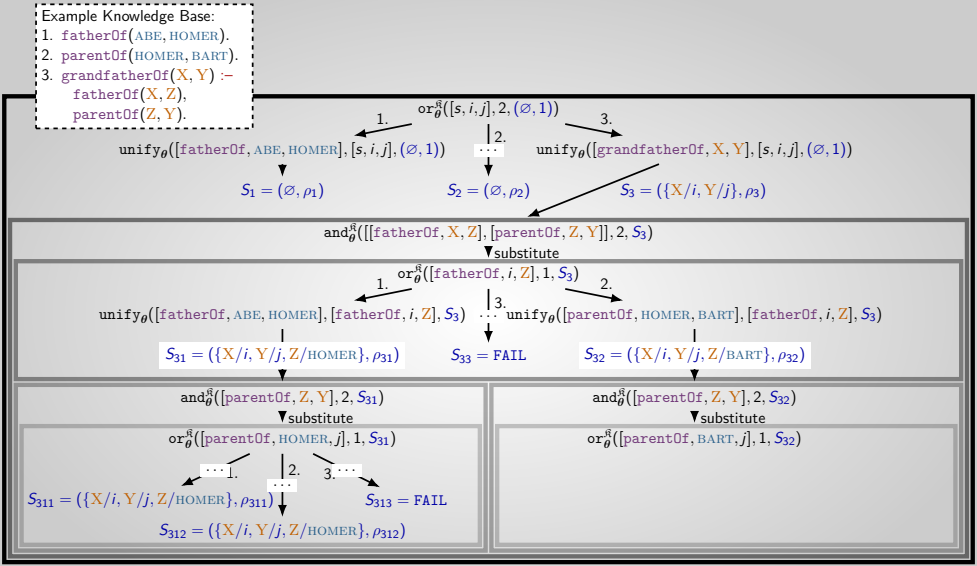
▼substitute

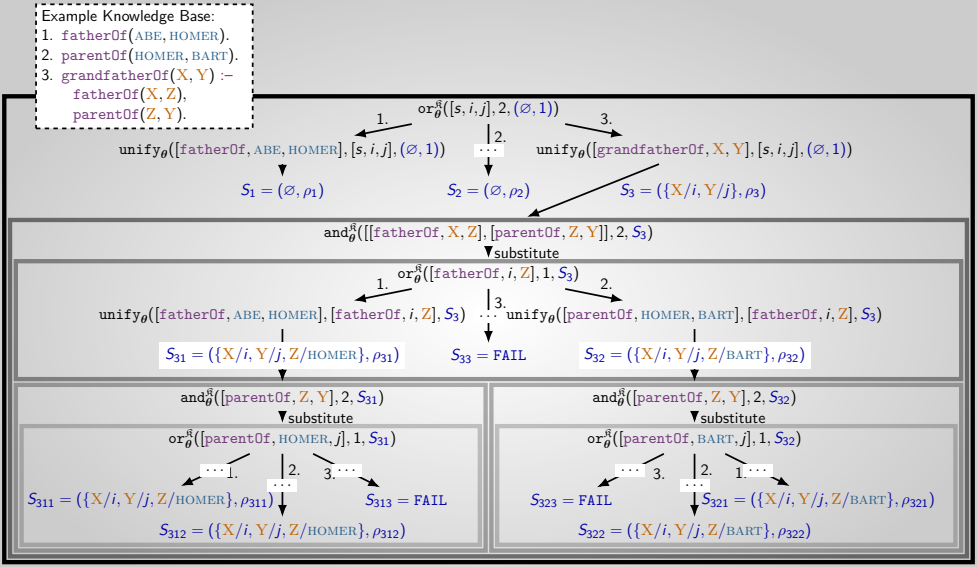
$\texttt{or}^{\mathbb{R}}_{\theta}([\texttt{parentOf}, \text{HOMER}, j], 1, S_{31})$

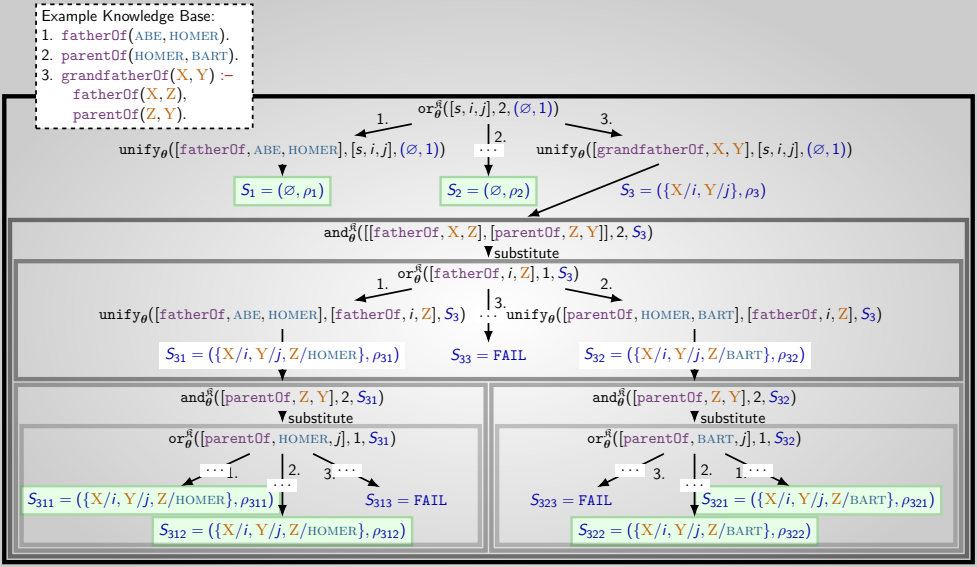
Example Knowledge Base:
1. $\texttt{fatherOf}(\text{ABE}, \text{HOMER})$.
2. $\texttt{parentOf}(\text{HOMER}, \text{BART})$.
3. $\texttt{grandfatherOf}(\text{X}, \text{Y}) :-$
   $\texttt{fatherOf}(\text{X}, \text{Z}),$
   $\texttt{parentOf}(\text{Z}, \text{Y}).$




Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
fatherOf(X, Z),
parentOf(Z, Y).
$\mathrm{or}_\theta^{\mathrm{S}}([s,i,j],2,(\varnothing,1))$
1.
2.
3.
$\mathrm{unify}_\theta([\mathrm{fatherOf},\mathrm{ABE},\mathrm{HOMER}],[s,i,j],(\varnothing,1))$
$\mathrm{unify}_\theta([\mathrm{grandfatherOf},\mathrm{X},\mathrm{Y}],[s,i,j],(\varnothing,1))$
$S_1 = (\varnothing, \rho_1)$
$S_2 = (\varnothing, \rho_2)$
$S_3 = (\{\mathrm{X}/i, \mathrm{Y}/j\}, \rho_3)$
$\mathrm{and}_\theta^{\mathrm{S}}([[\mathrm{fatherOf},\mathrm{X},\mathrm{Z}],[\mathrm{parentOf},\mathrm{Z},\mathrm{Y}]],2,S_3)$
substitute
$\mathrm{or}_\theta^{\mathrm{S}}([\mathrm{fatherOf},i,\mathrm{Z}],1,S_3)$
1.
2.
3.
$\mathrm{unify}_\theta([\mathrm{fatherOf},\mathrm{ABE},\mathrm{HOMER}],[\mathrm{fatherOf},i,\mathrm{Z}],S_3)$
$\mathrm{unify}_\theta([\mathrm{parentOf},\mathrm{HOMER},\mathrm{BART}],[\mathrm{fatherOf},i,\mathrm{Z}],S_3)$
$S_{31} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{31})$
$S_{33} = \mathtt{FAIL}$
$S_{32} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{32})$
$\mathrm{and}_\theta^{\mathrm{S}}([\mathrm{parentOf},\mathrm{Z},\mathrm{Y}],2,S_{31})$
substitute
$\mathrm{or}_\theta^{\mathrm{S}}([\mathrm{parentOf},\mathrm{HOMER},j],1,S_{31})$
1.
2.
3.
$S_{311} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{311})$
$S_{313} = \mathtt{FAIL}$
$S_{312} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{312})$
$\mathrm{and}_\theta^{\mathrm{S}}([\mathrm{parentOf},\mathrm{Z},\mathrm{Y}],2,S_{32})$


Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
fatherOf(X, Z),
parentOf(Z, Y).
$\mathrm{or}_\theta^{\mathrm{R}}([s,i,j],2,(\varnothing,1))$
1.
2.
3.
$\mathrm{unify}_\theta([\mathrm{fatherOf},\mathrm{ABE},\mathrm{HOMER}],[s,i,j],(\varnothing,1))$
$\mathrm{unify}_\theta([\mathrm{grandfatherOf},\mathrm{X},\mathrm{Y}],[s,i,j],(\varnothing,1))$
$S_1 = (\varnothing, \rho_1)$
$S_2 = (\varnothing, \rho_2)$
$S_3 = (\{\mathrm{X}/i, \mathrm{Y}/j\}, \rho_3)$
$\mathrm{and}_\theta^{\mathrm{R}}([[\mathrm{fatherOf},\mathrm{X},\mathrm{Z}],[\mathrm{parentOf},\mathrm{Z},\mathrm{Y}]],2,S_3)$
substitute
$\mathrm{or}_\theta^{\mathrm{R}}([\mathrm{fatherOf},i,\mathrm{Z}],1,S_3)$
1.
2.
3.
$\mathrm{unify}_\theta([\mathrm{fatherOf},\mathrm{ABE},\mathrm{HOMER}],[\mathrm{fatherOf},i,\mathrm{Z}],S_3)$
$\mathrm{unify}_\theta([\mathrm{parentOf},\mathrm{HOMER},\mathrm{BART}],[\mathrm{fatherOf},i,\mathrm{Z}],S_3)$
$S_{31} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{31})$
$S_{33} = \mathtt{FAIL}$
$S_{32} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{32})$
$\mathrm{and}_\theta^{\mathrm{R}}([\mathrm{parentOf},\mathrm{Z},\mathrm{Y}],2,S_{31})$
substitute
$\mathrm{or}_\theta^{\mathrm{R}}([\mathrm{parentOf},\mathrm{HOMER},j],1,S_{31})$
1.
2.
3.
$S_{311} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{311})$
$S_{313} = \mathtt{FAIL}$
$S_{312} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{312})$
$\mathrm{and}_\theta^{\mathrm{R}}([\mathrm{parentOf},\mathrm{Z},\mathrm{Y}],2,S_{32})$
substitute
$\mathrm{or}_\theta^{\mathrm{R}}([\mathrm{parentOf},\mathrm{BART},j],1,S_{32})$


Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
fatherOf(X, Z),
parentOf(Z, Y).
$\mathrm{or}^{\mathrm{S}}_{\theta}([s, i, j], 2, (\varnothing, 1))$
1.
2.
3.
$\mathrm{unify}_{\theta}([\mathrm{fatherOf}, \mathrm{ABE}, \mathrm{HOMER}], [s, i, j], (\varnothing, 1))$
$\mathrm{unify}_{\theta}([\mathrm{grandfatherOf}, \mathrm{X}, \mathrm{Y}], [s, i, j], (\varnothing, 1))$
$S_1 = (\varnothing, \rho_1)$
$S_2 = (\varnothing, \rho_2)$
$S_3 = (\{\mathrm{X}/i, \mathrm{Y}/j\}, \rho_3)$
$\mathrm{and}^{\mathrm{S}}_{\theta}([[\mathrm{fatherOf}, \mathrm{X}, \mathrm{Z}], [\mathrm{parentOf}, \mathrm{Z}, \mathrm{Y}]], 2, S_3)$
substitute
$\mathrm{or}^{\mathrm{S}}_{\theta}([\mathrm{fatherOf}, i, \mathrm{Z}], 1, S_3)$
1.
2.
3.
$\mathrm{unify}_{\theta}([\mathrm{fatherOf}, \mathrm{ABE}, \mathrm{HOMER}], [\mathrm{fatherOf}, i, \mathrm{Z}], S_3)$
$\mathrm{unify}_{\theta}([\mathrm{parentOf}, \mathrm{HOMER}, \mathrm{BART}], [\mathrm{fatherOf}, i, \mathrm{Z}], S_3)$
$S_{31} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{31})$
$S_{33} = \mathrm{FAIL}$
$S_{32} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{32})$
$\mathrm{and}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{Z}, \mathrm{Y}], 2, S_{31})$
substitute
$\mathrm{or}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{HOMER}, j], 1, S_{31})$
1.
2.
3.
$S_{311} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{311})$
$S_{313} = \mathrm{FAIL}$
$S_{312} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{312})$
$\mathrm{and}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{Z}, \mathrm{Y}], 2, S_{32})$
substitute
$\mathrm{or}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{BART}, j], 1, S_{32})$
3.
2.
1.
$S_{323} = \mathrm{FAIL}$
$S_{321} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{321})$
$S_{322} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{322})$


Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
fatherOf(X, Z),
parentOf(Z, Y).
$\mathrm{or}^{\mathrm{S}}_{\theta}([s, i, j], 2, (\varnothing, 1))$
1.
2.
3.
$\mathrm{unify}_{\theta}([\mathrm{fatherOf}, \mathrm{ABE}, \mathrm{HOMER}], [s, i, j], (\varnothing, 1))$
$\mathrm{unify}_{\theta}([\mathrm{grandfatherOf}, \mathrm{X}, \mathrm{Y}], [s, i, j], (\varnothing, 1))$
$S_1 = (\varnothing, \rho_1)$
$S_2 = (\varnothing, \rho_2)$
$S_3 = (\{\mathrm{X}/i, \mathrm{Y}/j\}, \rho_3)$
$\mathrm{and}^{\mathrm{S}}_{\theta}([[\mathrm{fatherOf}, \mathrm{X}, \mathrm{Z}], [\mathrm{parentOf}, \mathrm{Z}, \mathrm{Y}]], 2, S_3)$
substitute
$\mathrm{or}^{\mathrm{S}}_{\theta}([\mathrm{fatherOf}, i, \mathrm{Z}], 1, S_3)$
1.
2.
3.
$\mathrm{unify}_{\theta}([\mathrm{fatherOf}, \mathrm{ABE}, \mathrm{HOMER}], [\mathrm{fatherOf}, i, \mathrm{Z}], S_3)$
$\mathrm{unify}_{\theta}([\mathrm{parentOf}, \mathrm{HOMER}, \mathrm{BART}], [\mathrm{fatherOf}, i, \mathrm{Z}], S_3)$
$S_{31} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{31})$
$S_{33} = \mathrm{FAIL}$
$S_{32} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{32})$
$\mathrm{and}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{Z}, \mathrm{Y}], 2, S_{31})$
substitute
$\mathrm{or}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{HOMER}, j], 1, S_{31})$
1.
2.
3.
$S_{311} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{311})$
$S_{313} = \mathrm{FAIL}$
$S_{312} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{HOMER}\}, \rho_{312})$
$\mathrm{and}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{Z}, \mathrm{Y}], 2, S_{32})$
substitute
$\mathrm{or}^{\mathrm{S}}_{\theta}([\mathrm{parentOf}, \mathrm{BART}, j], 1, S_{32})$
3.
2.
1.
$S_{323} = \mathrm{FAIL}$
$S_{321} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{321})$
$S_{322} = (\{\mathrm{X}/i, \mathrm{Y}/j, \mathrm{Z}/\mathrm{BART}\}, \rho_{322})$

Batch Proving

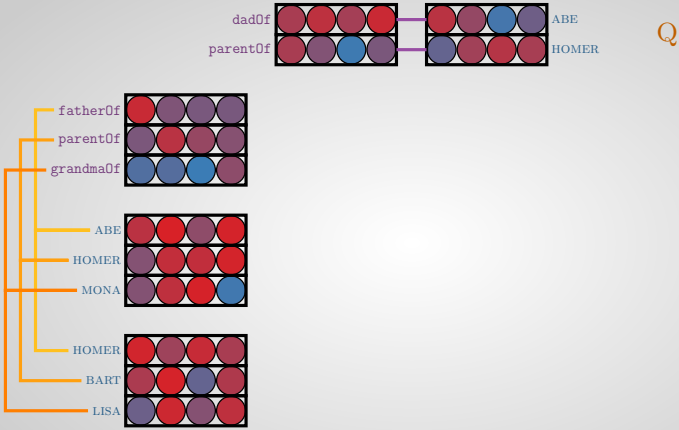
- $\boldsymbol{A} \in \mathbb{R}^{N \times k}$ matrix of N subsymbolic representations
- $\boldsymbol{B} \in \mathbb{R}^{M \times k}$ matrix of M other subsymbolic representations

$$\exp\left(-\sqrt{\left(\begin{bmatrix} \sum_{i=1}^{k} \boldsymbol{A}_{1i}^2 \\ \vdots \\ \sum_{i=1}^{k} \boldsymbol{A}_{Ni}^2 \end{bmatrix} \mathbf{1}_M^\top\right) + \left(\mathbf{1}_N \begin{bmatrix} \sum_{i=1}^{k} \boldsymbol{B}_{1i}^2 \\ \vdots \\ \sum_{i=1}^{k} \boldsymbol{B}_{Mi}^2 \end{bmatrix}^\top\right) - 2\boldsymbol{A}\boldsymbol{B}^\top}\right) \in \mathbb{R}^{N \times M}$$

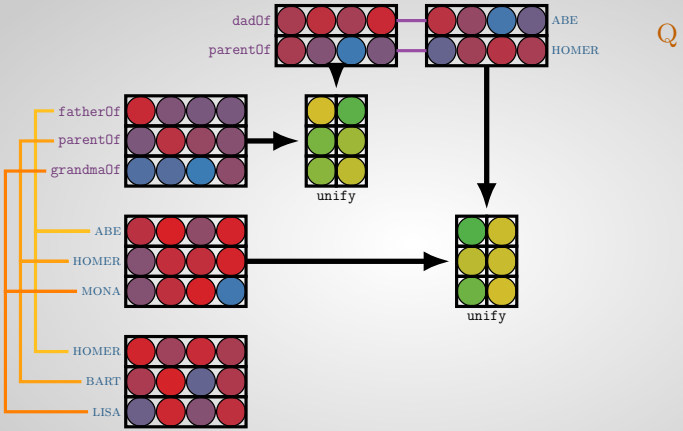
where $\mathbf{1}_N$ and $\mathbf{1}_M$ are vectors of N and M ones respectively, and the square root is taken element-wise.

Calculation on GPU

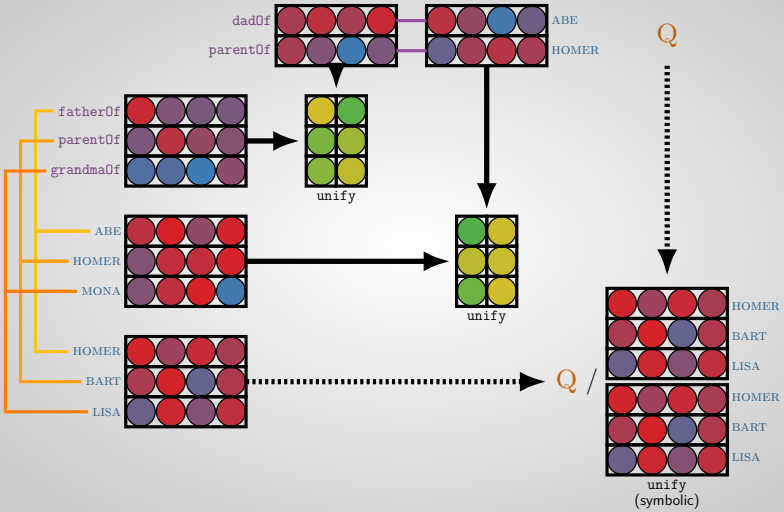
Calculation on GPU



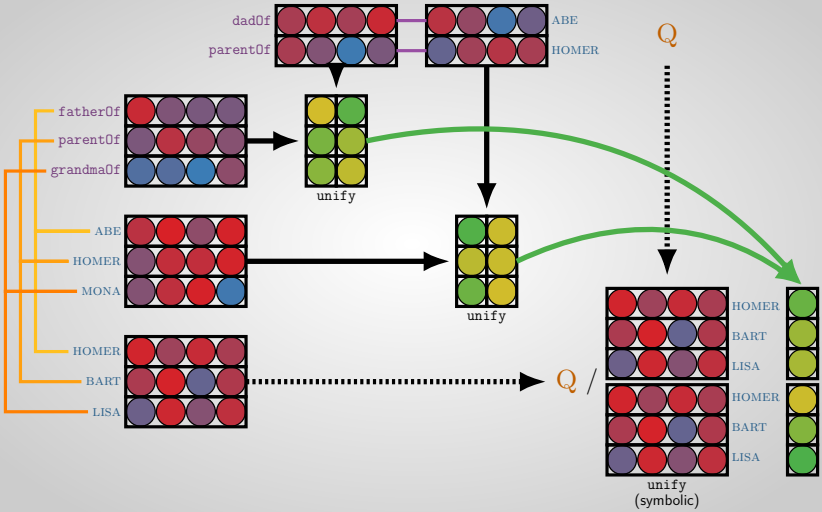
Calculation on GPU



Calculation on GPU



Calculation on GPU



Neural Inductive Logic Programming

1 $\boldsymbol{v}_{\texttt{fatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{HOMER}}).$
2 $\boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{LISA}}).$
3 $\boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{BART}}).$
4 $\boldsymbol{v}_{\texttt{grandpaOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{LISA}}).$
5 $\boldsymbol{v}_{\texttt{grandfatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{MAGGIE}}).$

Neural Inductive Logic Programming

$$\boldsymbol{v}_{\texttt{fatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{HOMER}}).$$
$$\boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{LISA}}).$$
$$\boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{BART}}).$$
$$\boldsymbol{v}_{\texttt{grandpaOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{LISA}}).$$
$$\boldsymbol{v}_{\texttt{grandfatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{MAGGIE}}).$$

$$\boldsymbol{\theta}_1(\mathrm{X}_1, \mathrm{Y}_1) \coloneqq \boldsymbol{\theta}_2(\mathrm{X}_1, \mathrm{Z}_1), \boldsymbol{\theta}_3(\mathrm{Z}_1, \mathrm{Y}_1).$$
$$\boldsymbol{\theta}_4(\mathrm{X}_2, \mathrm{Y}_2) \coloneqq \boldsymbol{\theta}_5(\mathrm{X}_2, \mathrm{Y}_2).$$

Neural Inductive Logic Programming

1 $\boldsymbol{v}_{\texttt{fatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{HOMER}}).$
2 $\boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{LISA}}).$
3 $\boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{BART}}).$
4 $\boldsymbol{v}_{\texttt{grandpaOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{LISA}}).$
5 $\boldsymbol{v}_{\texttt{grandfatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{MAGGIE}}).$
6 $\boldsymbol{\theta}_1(\mathrm{X}_1, \mathrm{Y}_1) \coloneqq \boldsymbol{\theta}_2(\mathrm{X}_1, \mathrm{Z}_1), \boldsymbol{\theta}_3(\mathrm{Z}_1, \mathrm{Y}_1).$
7 $\boldsymbol{\theta}_4(\mathrm{X}_2, \mathrm{Y}_2) \coloneqq \boldsymbol{\theta}_5(\mathrm{X}_2, \mathrm{Y}_2).$

Decoding Induced Rules

- Find closest representations of known predicate

Neural Inductive Logic Programming

$$
\begin{array}{ll}
1 & \boldsymbol{v}_{\texttt{fatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{HOMER}}). \\
2 & \boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{LISA}}). \\
3 & \boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{BART}}). \\
4 & \boldsymbol{v}_{\texttt{grandpaOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{LISA}}). \\
5 & \boldsymbol{v}_{\texttt{grandfatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{MAGGIE}}). \\
6 & \boldsymbol{\theta}_1(\mathrm{X}_1, \mathrm{Y}_1) \coloneqq \boldsymbol{\theta}_2(\mathrm{X}_1, \mathrm{Z}_1), \boldsymbol{\theta}_3(\mathrm{Z}_1, \mathrm{Y}_1). \\
7 & \boldsymbol{\theta}_4(\mathrm{X}_2, \mathrm{Y}_2) \coloneqq \boldsymbol{\theta}_5(\mathrm{X}_2, \mathrm{Y}_2).
\end{array}
$$

Decoding Induced Rules

- Find closest representations of known predicate
- Take minimum RBF similarity as rule confidence

Neural Inductive Logic Programming

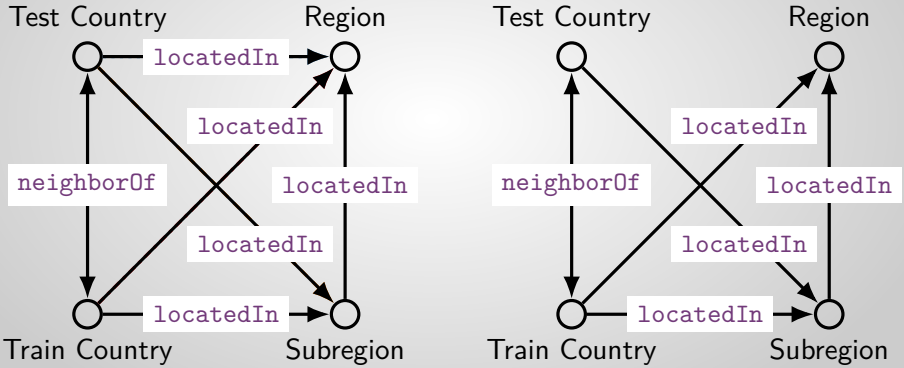
$$
\begin{array}{ll}
1 & \boldsymbol{v}_{\texttt{fatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{HOMER}}). \\
2 & \boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{LISA}}). \\
3 & \boldsymbol{v}_{\texttt{parentOf}}(\boldsymbol{v}_{\text{HOMER}}, \boldsymbol{v}_{\text{BART}}). \\
4 & \boldsymbol{v}_{\texttt{grandpaOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{LISA}}). \\
5 & \boldsymbol{v}_{\texttt{grandfatherOf}}(\boldsymbol{v}_{\text{ABE}}, \boldsymbol{v}_{\text{MAGGIE}}). \\
6 & \boldsymbol{\theta}_1(\mathrm{X}_1, \mathrm{Y}_1) \coloneqq \boldsymbol{\theta}_2(\mathrm{X}_1, \mathrm{Z}_1), \boldsymbol{\theta}_3(\mathrm{Z}_1, \mathrm{Y}_1). \\
7 & \boldsymbol{\theta}_4(\mathrm{X}_2, \mathrm{Y}_2) \coloneqq \boldsymbol{\theta}_5(\mathrm{X}_2, \mathrm{Y}_2).
\end{array}
$$

Decoding Induced Rules

- Find closest representations of known predicate
- Take minimum RBF similarity as rule confidence
- Rule confidence is an upper bound on the proof success that can be achieved when applying the rule

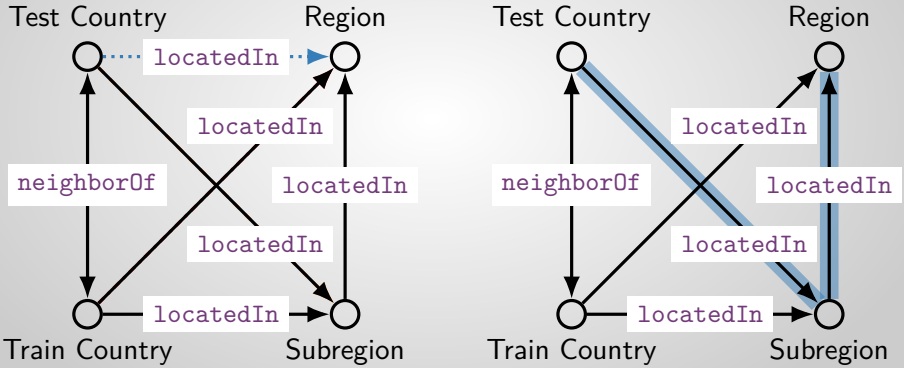
Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)



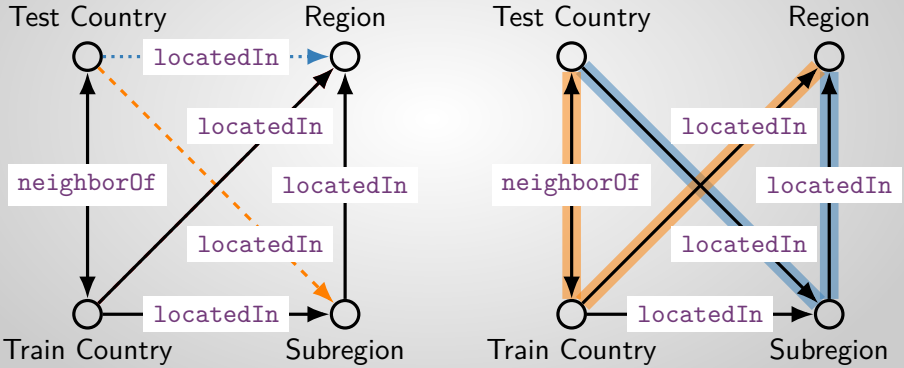
Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)



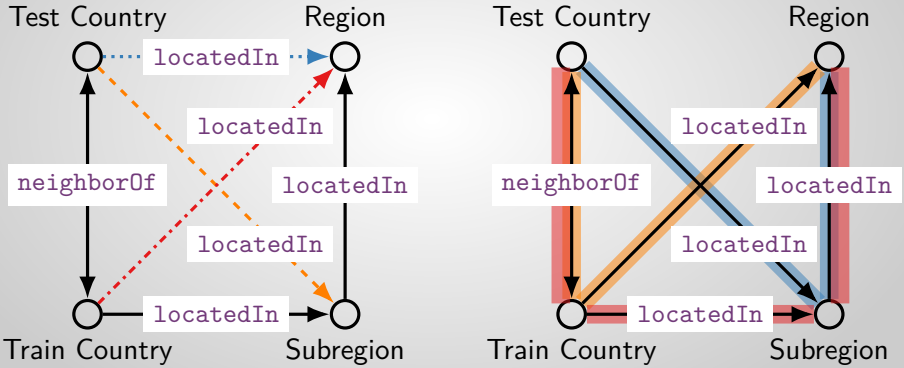
Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)



Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)



Details

- Models implemented in TensorFlow

Details

- Models implemented in TensorFlow

  ComplEx Neural link prediction model by Trouillon et al. (2016)

Details

- Models implemented in TensorFlow

  ComplEx Neural link prediction model by Trouillon et al. (2016)

  NTP End-to-end differentiable prover

Details

- Models implemented in TensorFlow

  ComplEx Neural link prediction model by Trouillon et al. (2016)

  NTP End-to-end differentiable prover

  NTPλ Prover trained with ComplEx as auxiliary loss

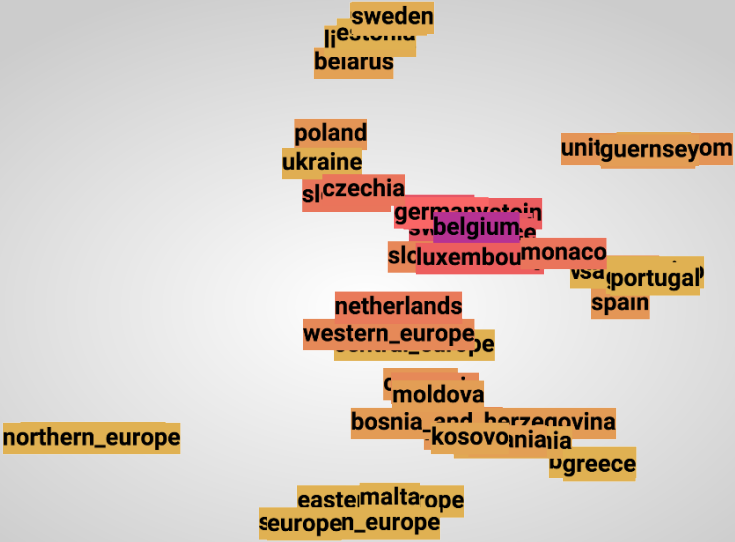
Details

- Models implemented in TensorFlow

  ComplEx Neural link prediction model by Trouillon et al. (2016)
  NTP End-to-end differentiable prover
  NTPλ Prover trained with ComplEx as auxiliary loss

- Rule Templates:

  Kinship, Nations & UMLS
  20 $\#1(X, Y) :- \#2(X, Y).$
  20 $\#1(X, Y) :- \#2(Y, X).$
  20 $\#1(X, Y) :- \#2(X, Z), \#3(Z, Y).$
  Countries S1
  3 $\#1(X, Y) :- \#1(Y, X).$
  3 $\#1(X, Y) :- \#2(X, Z), \#2(Z, Y).$
  Countries S2
  3 $\#1(X, Y) :- \#2(X, Z), \#3(Z, Y).$
  Countries S3
  3 $\#1(X, Y) :- \#2(X, Z), \#3(Z, W), \#4(W, Y).$


sweden
belarus
poland
ukraine
czechia
belgium
monaco
portugal
spain
netherlands
western_europe
moldova
bosnia
kosovo
greece
northern_europe
malta
europe

Results

Corpus		Metric	Model		
			ComplEx	**NTP**	**NTPλ**
Countries	S1	AUC-PR	99.37 ± 0.4	90.83 ± 15.4	**100.00** $\pm$ 0.0
	S2	AUC-PR	87.95 ± 2.8	87.40 ± 11.7	**93.04** $\pm$ 0.4
	S3	AUC-PR	48.44 ± 6.3	56.68 ± 17.6	**77.26** $\pm$ 17.0
Kinship		MRR	0.46	0.36	**0.48**
		HITS@1	0.34	0.24	**0.39**
		HITS@3	**0.49**	0.40	0.47
		HITS@10	**0.74**	0.60	0.71
Nations		MRR	0.60	**0.63**	0.62
		HITS@1	0.46	**0.48**	0.45
		HITS@3	0.67	0.69	**0.72**
		HITS@10	0.97	0.98	**0.99**
UMLS		MRR	0.58	0.57	**0.60**
		HITS@1	0.47	0.47	**0.51**
		HITS@3	0.63	0.60	**0.64**
		HITS@10	0.80	0.79	**0.81**

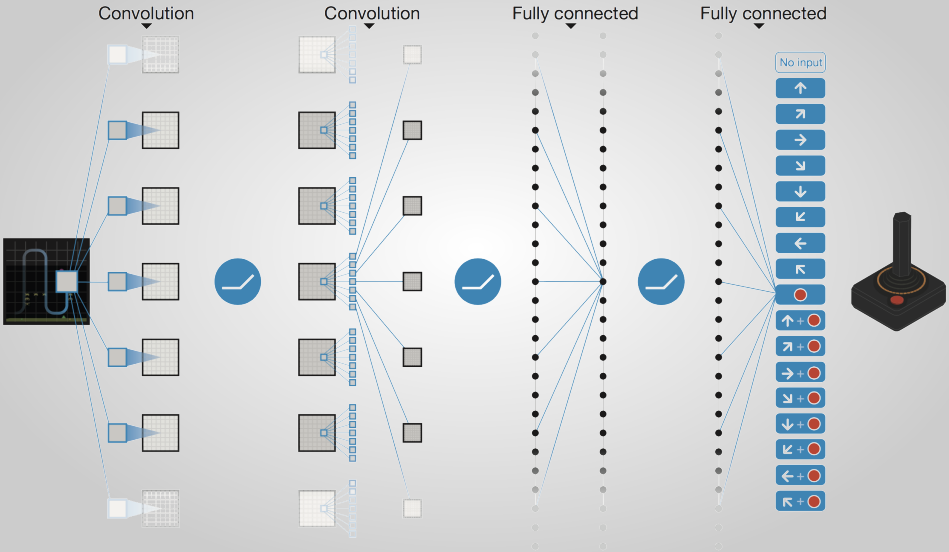
Results

Corpus		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).`

End-to-end Differentiable Planning

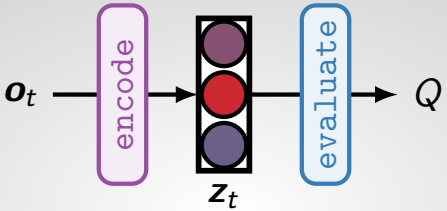
work-in-progress

DQN



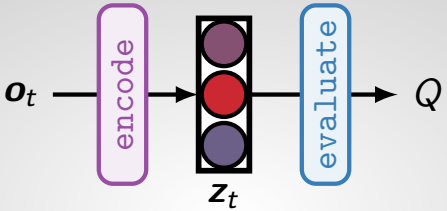
Mnih et al. (2015)

DQN



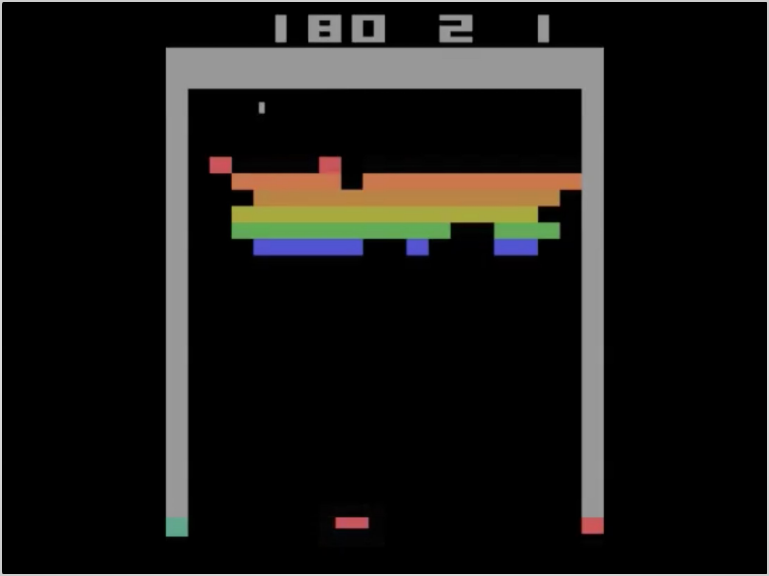
Mnih et al. (2015)

DQN



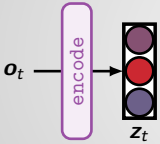
$$\mathcal{L}(\boldsymbol{\theta}) = \left(\overbrace{\underbrace{r}_{\text{reward}} + \underbrace{\gamma}_{\text{discount}} \max_{a'} Q(s', a', \boldsymbol{\theta}^-)}^{target} - Q(s, a, \boldsymbol{\theta}) \right)^2$$

Mnih et al. (2015)

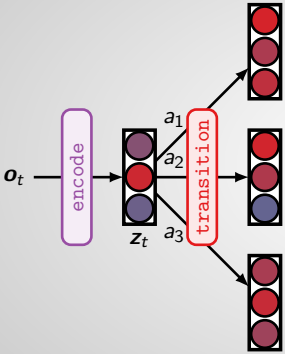

180 2 1


460
OXYGEN
ACTIVISION

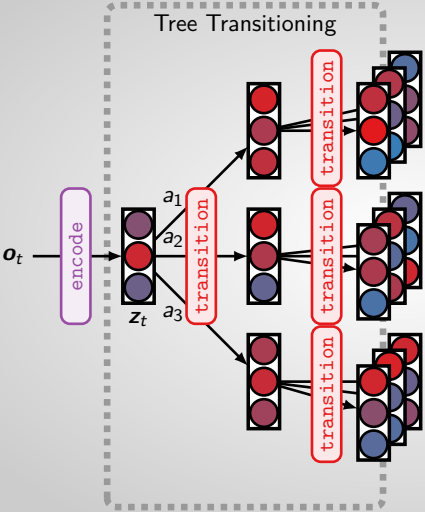
Tree Planning



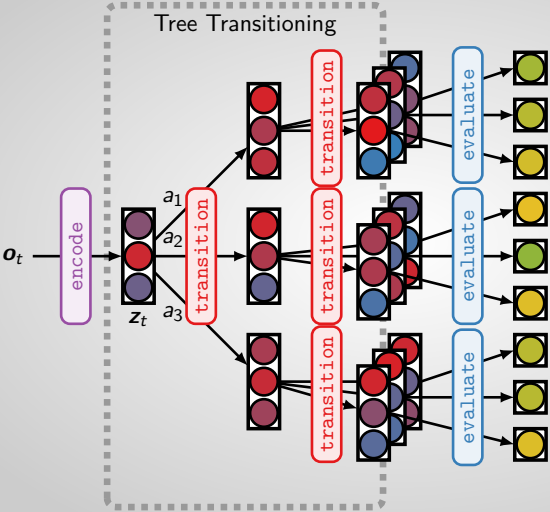
Tree Planning



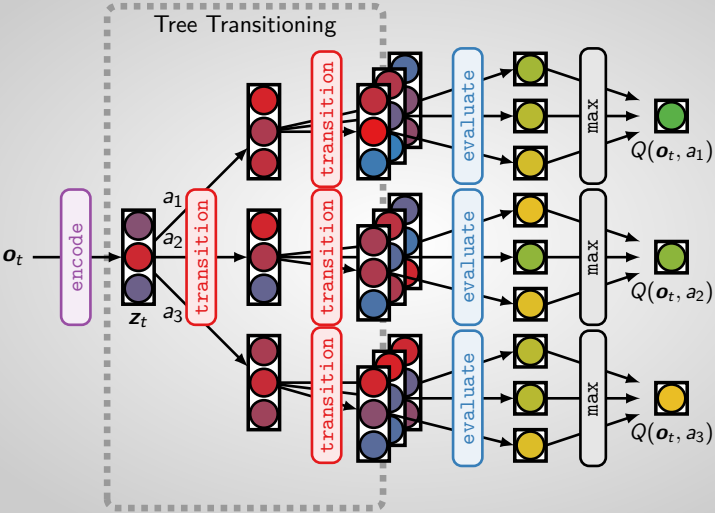
Tree Planning



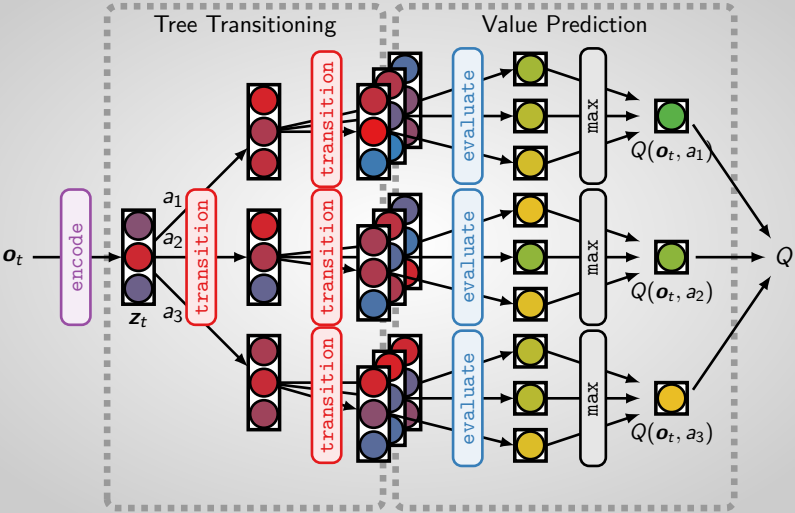
Tree Planning



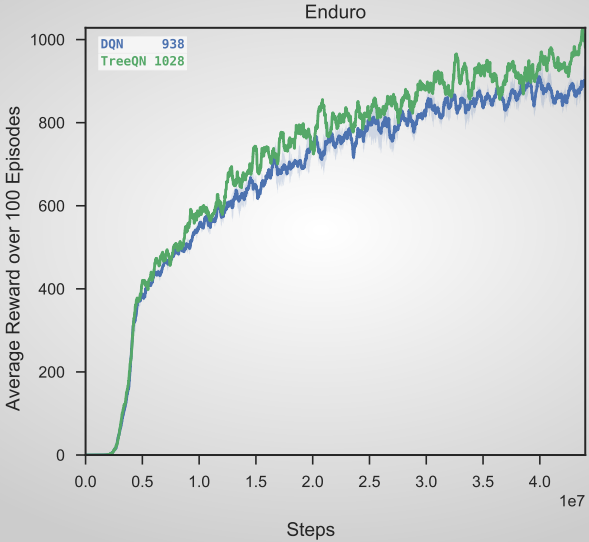
Tree Planning



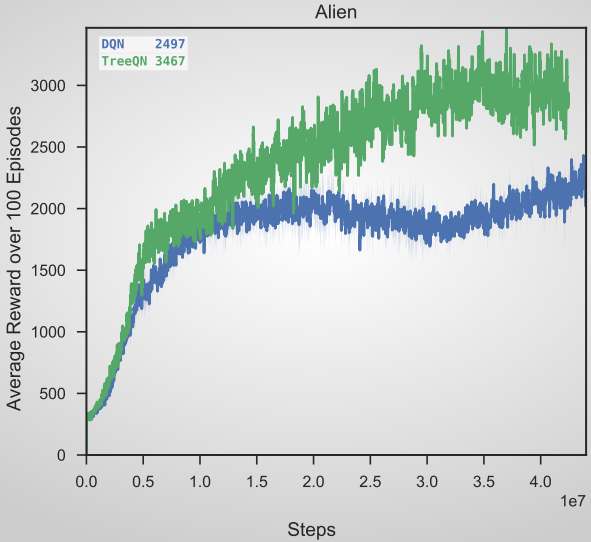
Tree Planning



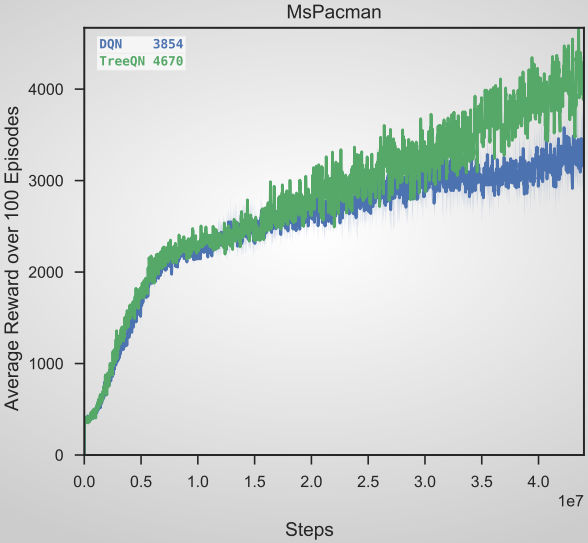
Results


Enduro
DQN 938
TreeQN 1028
Average Reward over 100 Episodes
0
200
400
600
800
1000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e7
Steps

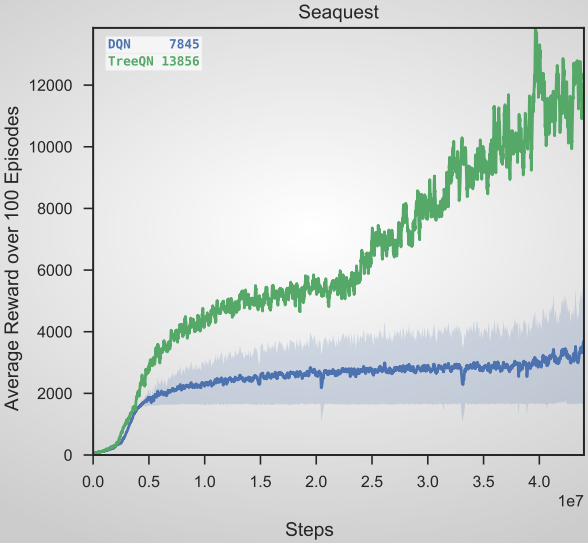
Results


Alien
DQN 2497
TreeQN 3467
Average Reward over 100 Episodes
0
500
1000
1500
2000
2500
3000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e7
Steps

Results


MsPacman
DQN 3854
TreeQN 4670
Average Reward over 100 Episodes
0
1000
2000
3000
4000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e7
Steps

Results


Seaquest
DQN 7845
TreeQN 13856
Average Reward over 100 Episodes
0
2000
4000
6000
8000
10000
12000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e7
Steps

ACTIVISION

ACTIVISION

ACTIVISION

ACTIVISION

ACTIVISION

ACTIVISION

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases
- Future research:

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases
- Future research:
  - **Scale** to larger knowledge bases

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases
- Future research:
  - **Scale** to larger knowledge bases
  - Connect to RNNs for natural language statements

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases
- Future research:
  - **Scale** to larger knowledge bases
  - Connect to RNNs for natural language statements
  - Proving of mathematical theorems

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases
- Future research:
  - **Scale** to larger knowledge bases
  - Connect to RNNs for natural language statements
  - Proving of mathematical theorems
  - Visual reasoning

Summary

- Prolog's backward chaining as recipe for recursively constructing a neural network to prove facts in a knowledge base
- Proof success differentiable w.r.t. subsymbolic representations
- Can learn vector representations of symbols and induce interpretable rules of predefined structure
- Various GPU optimizations: batch proving, tree pruning etc.
- Outperform neural link prediction model on benchmark knowledge bases
- Future research:
  - **Scale** to larger knowledge bases
  - Connect to RNNs for natural language statements
  - Proving of mathematical theorems
  - Visual reasoning
- Encouraging preliminary results using tree planning for Atari

Thank you!

http://rockt.github.com
tim.rocktaschel@cs.ox.ac.uk
Twitter: @_rockt

References

Guillaume Bouchard, Sameer Singh, and Theo Trouillon. 2015. On approximate reasoning capabilities of low-rank vector spaces. In *Proceedings of the 2015 AAAI Spring Symposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Stanley Kok and Pedro M. Domingos. 2007. Statistical predicate invention. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 433–440.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. URL https://doi.org/10.1038/nature14236.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, California, United States*, volume abs/1705.11040.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2071–2080.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*.