SEMAPRO 2021

Semantic Reasoning with Differentiable Graph Transformations

Alberto Cetoli

alberto.cetoli@uk.qbe.com

QBE Europe

About Me

- Former Condensed Matter Physicist
- 8+ years of industry experience on NLProc
  - Mainly Information Extraction from documents
  - Interested in IE and Knowledge Representation
- Data scientist at QBE Europe

Introduction

Quick Summary

- **Graphs can be represented as a set of predicates**
- **Rules can be represented as linear algebra operations**
- **Linear algebra representation allows for trainable rules**

Sections

- Rules as graph transformations
- Matching and Creating graphs
- Chaining rules
- Differentiable learning over rules
- Examples and results

Rules as graph transformations

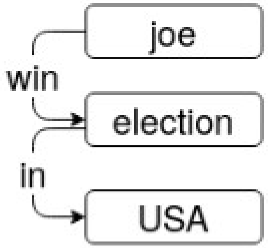
- Rules are precise
- Easy to explain
- Quick to deploy

Rules as graph transformations

Graphs can be represented as set of predicates

Initial facts

joe(a), win(a,b), election(b), in(b,c), USA(c)

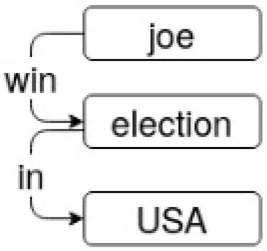


Rules as graph transformations

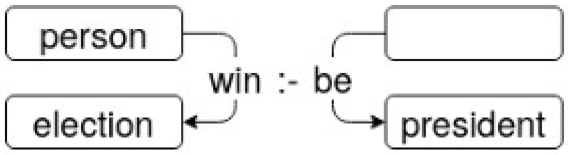
Rules transform match one graph and create another

Initial facts

joe(a), win(a,b), election(b),
in(b,c), USA(c)

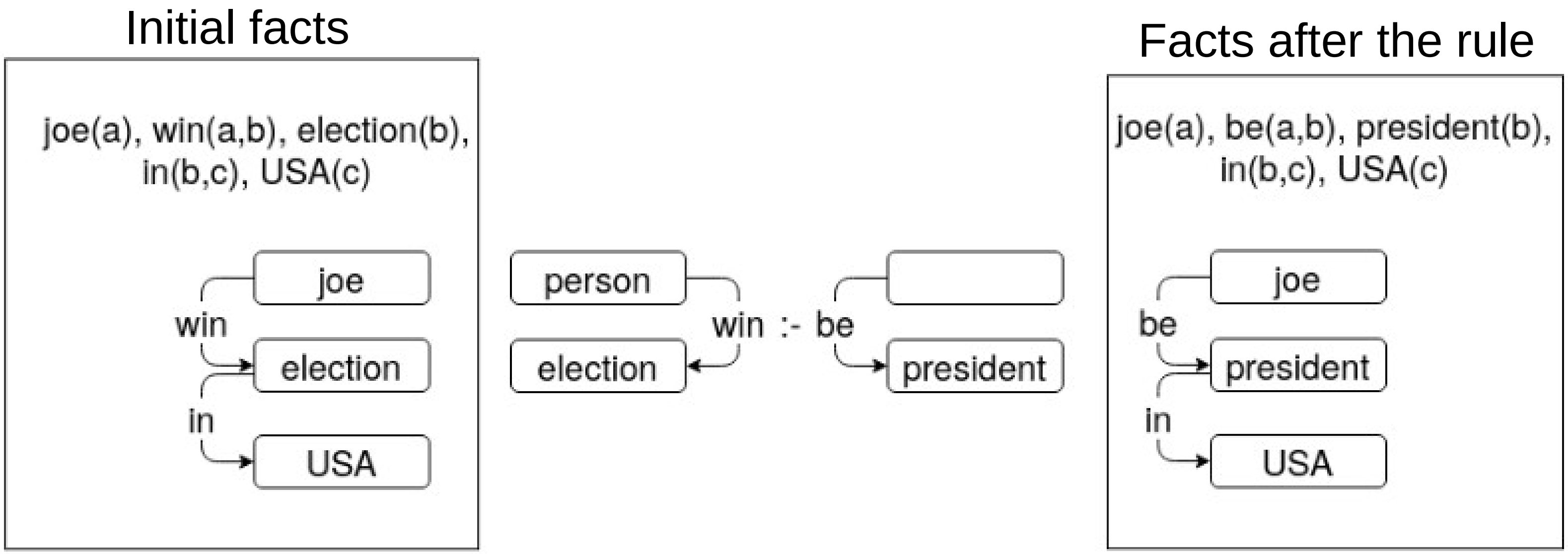



```
MATCH person>0.6(a), win>0.7(a,b), election>0.6(b)
CREATE (a), be(a,b), president(b)
```



Rules as graph transformations

Rules transform match one graph and create another



Matching and Creating graphs

The MATCH clause is a *precondition* for matching facts

```
MATCH person>0.6(a), win>0.7(a,b), election>0.6(b)
```

- Predicate names are embeddings
  - Pre-trained Glove embeddings are used for nodes
  - Matching of a predicate happens if dot product is > threshold
  - Individuality assertion:

$$joe \approx person \iff \mathrm{embedding}(joe) \cdot \mathrm{embedding}(person) > t$$

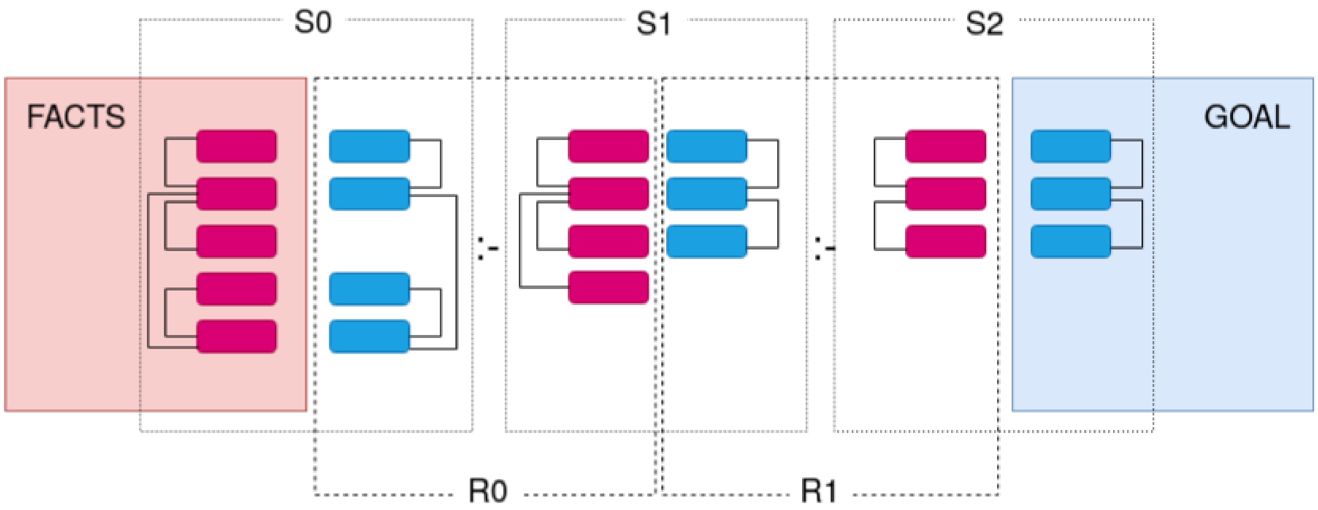
Matching and Creating graphs

The CREATE clause is a *postcondition*

```
CREATE (a), be(a,b), president(b)
```

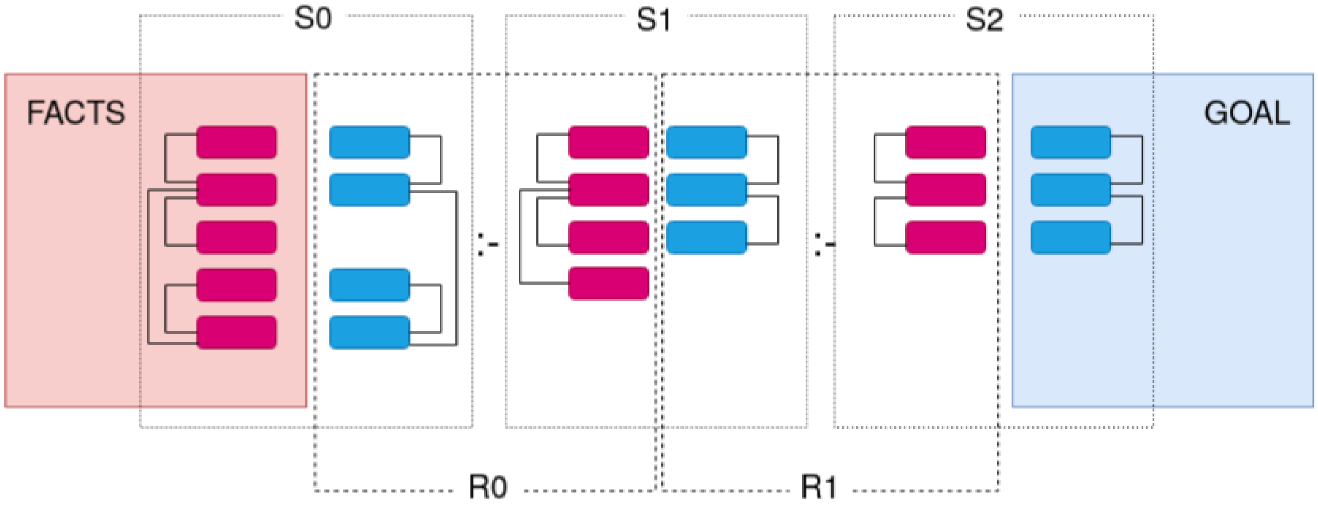
- It adds new predicates to the set of facts matched in the *preconditions*

Chaining rules



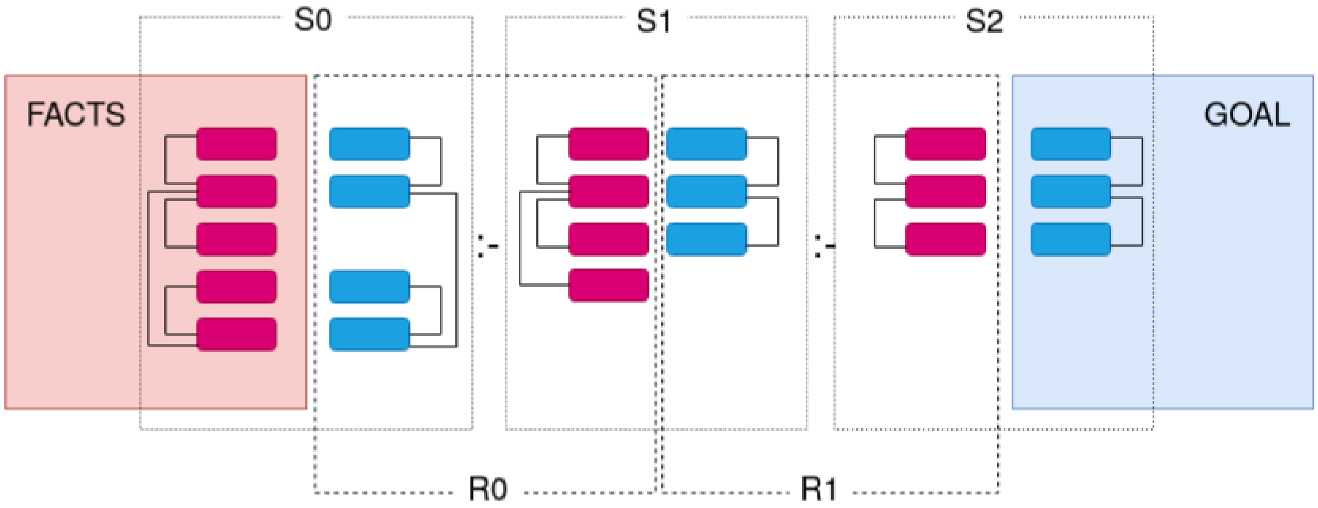
- Fact predicates have truth conditions described in a vector **f**
- Goal predicates have truth conditions described in a vector **g**
- Rules can be applied sequentially
- A chain of rules has a 1-to-1 correspondence with a sequence of matrices

Chaining rules



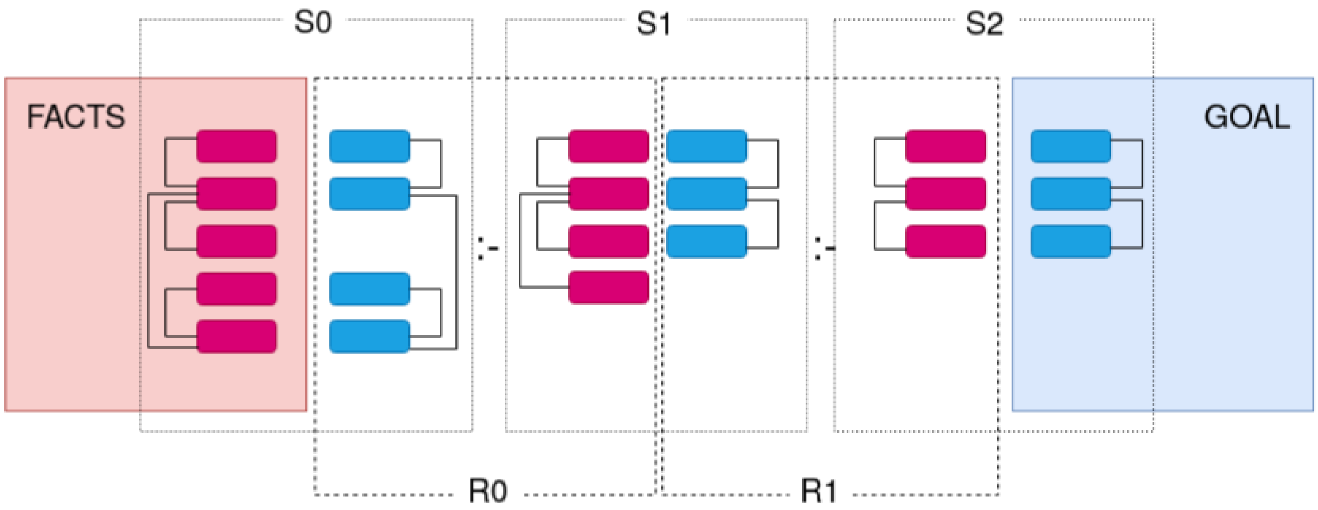
Similarity Matrix S
Describes the matching of facts and preconditions

Chaining rules



Rule Propagation Matrix R
Describes how information is propagated from the precondition to the postconditions

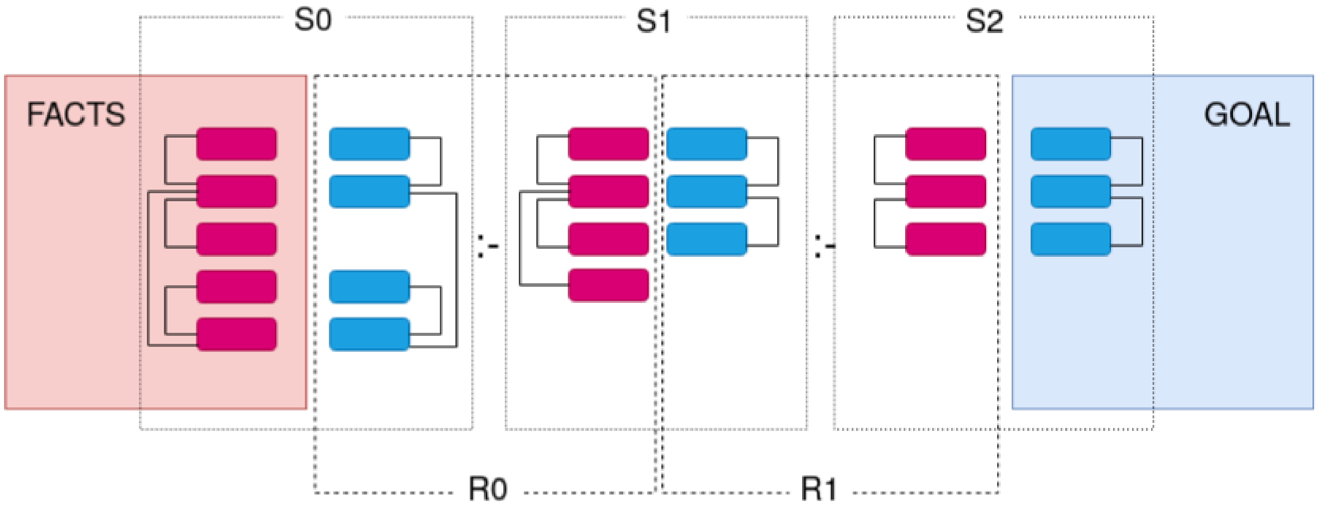
Chaining rules



Set of truth conditions for each predicate after n steps is

$$f_n = S_{n-1} \ldots R_1 S_1 R_0 S_0 f_0$$

Chaining rules



Given a pair of *fact* and *goal* predicates, embeddings can be trained using backpropagation with loss

$$\mathcal{L} = g \log(f_n)$$

Examples and results

One-rule learning

```
person(a), spouse(a,b), person(b), be(a,c), first-lady(c)
```

Facts

```
MATCH *(a), *(a,b), *(b), *(a,c), *(c)
CREATE (b), *(b,d), *(d)
```

Template rule with random embeddings

```
person(a), profession(a,b), president(b)
```

Goal

Examples and results

One-rule learning

```
person(a), spouse(a,b), person(b), be(a,c), first-lady(c)
```

Facts

```
MATCH person>0.6(a), first-lady>0.6(b), person>0.6(c),
      be>0.63631916(a,b), spouse>0.6338593(a,c)
CREATE (b), president(d), profession(b,d)
```

Learned rule
connecting facts and goal

```
person(a), profession(a,b), president(b)
```

Goal

Examples and results

Chained Two-rule learning

```
fruit(a), be(a,b), round(b), be(a,c), delicious(c)
```

Facts

```
MATCH *(a), *(a,b), *(b), *(a,c), *(c)
CREATE (b), and(b,c), (c)

MATCH *(a), and(a,b), *(b)
CREATE *(c), *(c,d), *(d)
```

Template rules with random embeddings

```
fruit(a), be(a,b), apple(b)
```

Goal

Examples and results

Chained Two-rule learning

```
fruit(a), be(a,b), round(b), be(a,c), delicious(c)
```

Facts

```
MATCH fruit>0.6(a), round>0.6(b), delicious>0.6(c),
      be>0.6953449(a,b), be>0.6957883(a,c)
CREATE (b), (c), and(b,c)

MATCH round>0.6(a), delicious>0.6(b), and>0.9(a,b)
CREATE fruit(c), apple(d), be(c,d)
```

Learned rules

```
fruit(a), be(a,b), apple(b)
```

Goal

Conclusions and future work

- Presented a semantic reasoner
- Facts are described as graphs/predicates
- Rules transform graphs using preconditions and postconditions
- A chain of rules is equivalent to a sequence of linear algebra transformations
- Rules can be learned from pairs of facts and goals
- Limitations
  - The learning algorithm is slow because it searches a solution among all the possible paths
  - A Montecarlo/guided approach would be faster
  - A faster algorithm will allow more template rules

Conclusions and future work

Thank you!