

Christian Alrabbaa   Stefan Borgwardt   Patrick Koopmann   Alisa Kovtunova

# Finding Proofs for Description Logic Entailments in Practice

Based on “Finding Small Proofs for Description Logic Entailments—Theory and Practice” (LPAR’20) // Explainable Logic-Based Knowledge Representation (XLoKR 2020), September 14, 2020

# Description Logics and Ontologies

## Syntax of DL $\mathcal{ALC}$

**Concepts:**  $C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$

**Axioms:**  $\alpha ::= C \sqsubseteq C \mid C \equiv C$

## Description Logics

- Well-established formalism for specifying terminological knowledge in **Ontologies**
- Used for many large-scale ontologies
  - SNOMED CT: over 300,000 concepts
  - BioPortal: repository of bio-medical ontologies, currently hosting 889 ontologies defining 12,084,317 terms
  - MOWLCorp: ontologies obtained by web-crawling, containing 21,000 ontologies

# Description Logics and Ontologies

## Syntax of DL $\mathcal{ALC}$

**Concepts:**  $C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$

**Axioms:**  $\alpha ::= C \sqsubseteq C \mid C \equiv C$

## Description Logics

- Well-established formalism for specifying terminological knowledge in **Ontologies**
- Used for many large-scale ontologies
- With increasing complexity of the ontology, understanding entailments becomes both crucial and difficult

# Description Logics and Ontologies

## Syntax of DL $\mathcal{ALC}$

**Concepts:**  $C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$

**Axioms:**  $\alpha ::= C \sqsubseteq C \mid C \equiv C$

## Description Logics

- Well-established formalism for specifying terminological knowledge in **Ontologies**
- Used for many large-scale ontologies
- With increasing complexity of the ontology, understanding entailments becomes both crucial and difficult
- One typical reasoning task is **classification**
  - compute all entailed axioms of form  $A \sqsubseteq B$
  - obtain **concept hierarchy**

# Current Tool of Choice: Justifications

The screenshot displays the OpenGALEN6 ontology editor. The main window shows a class hierarchy on the left and a central workspace. The class hierarchy is organized as follows:

- Class hierarchy (inferred):
  - AnthraxVaccine
  - AntidiureticAgent
  - BCGVaccine
  - BodySystemPhenomenon
    - AnatomicalSystemSign
    - CardiovascularSystemPhenomenon
    - CentralNervousSystemDisorder
    - DigestiveSystemPhenomenon
    - EndocrineSystemPhenomenon
    - ExaminingProcesswhichInvolvesBodySystem
    - GenitoUrinarySystemPhenomenon
    - HaematologicalDisorder
    - HaematologicalSystemPathology
    - ImmuneSystemPhenomenon
    - LymphoreticularSystemPhenomenon
    - MentalDisease
    - MentalDisorder
    - MentalDisturbance
    - MentalIllness
    - MusculoSkeletalSystemPhenomenon
    - NervousSystemPhenomenon
      - ExaminingProcesswhichInvolvesNervousSystem
      - NervousSystemSign
      - PathologicalProcesswhichInvolvesNervousSystem
      - SignwhichInvolvesNervousSystem
    - OrodontalSystemPhenomenon

The central workspace displays the text "Nothing Selected". The bottom panel shows the "Class hierarchy: NervousSystemSign" with the following classes listed:

- NervousSystemSign
- OrodontalSign
- PelvicSign
- ...

# Current Tool of Choice: Justifications

The screenshot displays the OpenGALEN6 ontology editor interface. The main window shows a class hierarchy on the left and details for the selected class, **NervousSystemSign**, on the right.

**Class Hierarchy (Left Panel):**

- OpenGALEN6\_FULL\_WithPropertyChains (file:/home/patrick/Documents/Ontologies/OpenGALEN6/OpenGALEN6\_FULL\_WithPropertyChains.owl)
- TopCategory (DomainCategory) Phenomenon (BodySystemPhenomenon) NervousSystemPhenomenon NervousSystemSign
- Active ontology: Entities | Annotation properties | Individuals by class | DL Query
- Class hierarchy (Inferred): NervousSystemSign
  - AnthraxVaccine
  - AntidiureticAgent
  - BCGVaccine
  - BodySystemPhenomenon
    - AnatomicalSystemSign
    - CardiovascularSystemPhenomenon
    - CentralNervousSystemDisorder
    - DigestiveSystemPhenomenon
    - EndocrineSystemPhenomenon
    - ExaminingProcesswhichinvolvesBodySystem
    - GenitoUrinarySystemPhenomenon
    - HaematologicalDisorder
    - HaematologicalSystemPathology
    - ImmuneSystemPhenomenon
    - LymphoreticularSystemPhenomenon
    - MentalDisease
    - MentalDisorder
    - MentalDisturbance
    - MentalIllness
    - MusculoSkeletalSystemPhenomenon
    - NervousSystemPhenomenon
      - ExaminingProcesswhichinvolvesNervousSystem
      - NervousSystemSign**
      - PathologicalProcesswhichinvolvesNervousSystem
      - SignwhichinvolvesNervousSystem
      - OridentalSystemPhenomenon

**Class Details (Right Panel):**

- Class Annotations: SANCTION-LEVELAP [type: xsd:string], GRAMMATICAL
- Description: Equivalent To: SignwhichinvolvesNervousSystem
- SubClass Of: AnatomicalSystemSign, NervousSystemPhenomenon, PhenomenonwhichinvolvesNervousSystem, SignwhichinvolvesAnatomicalSystem
- General class axioms: Sign and (involves some AnatomicalSystem), Phenomenon and (involves some NervousSystem)
- Instances: (None listed)
- Target for Key: (None listed)

At the bottom, the 'Data properties' tab is active, showing 'Classes' and 'Object properties' for the selected class. The 'Class hierarchy' section shows 'NervousSystemSign' as the selected class, with its subclasses 'OridentalSign', 'PelvicSign', and 'SignwhichinvolvesNervousSystem' listed below it.

# Current Tool of Choice: Justifications

The screenshot displays the OpenALENE ontology editor interface. The main window shows a class hierarchy on the left and a detailed view of the `NervousSystemSign` class on the right. The class hierarchy includes:

- AnthraxVaccine
- AntidiureticAgent
- BCGVaccine
- ▼ BodySystemPhenomenon
  - AnatomicalSystemSign
  - CardiovascularSystemPhenomenon
  - CentralNervousSystemDisorder
  - DigestiveSystemPhenomenon
  - EndocrineSystemPhenomenon
  - ExaminingProcesswhichInvolvesBodySystem
  - GenitoUrinarySystemPhenomenon
  - HaematologicalDisorder
  - HaematologicalSystemPathology
  - ImmuneSystemPhenomenon
  - LymphoreticularSystemPhenomenon
  - MentalDisease
  - MentalDisorder
  - MentalDisturbance
  - MentalIllness
  - MusculoSkeletalSystemPhenomenon
  - NervousSystemPhenomenon
    - ExaminingProcesswhichInvolvesNervousSystem
    - NervousSystemSign
    - PathologicalProcesswhichInvolvesNervousSystem
    - SignwhichInvolvesNervousSystem
  - OrodontalSystemPhenomenon

The detailed view for `NervousSystemSign` shows:

- Annotations:** SANCTION-LEVELAP [type: xsd:string], GRAMMATICAL
- Description:** SignwhichInvolvesNervousSystem
- Equivalent To:** SignwhichInvolvesNervousSystem
- SubClass Of:** AnatomicalSystemSign, NervousSystemPhenomenon, PhenomenonwhichInvolvesNervousSystem, SignwhichInvolvesAnatomicalSystem
- General class axioms:** Sign and (Involves some AnatomicalSystem), Phenomenon and (Involves some NervousSystem)
- SubClass Of (Anonymous Ancestor):** Sign and (Involves some AnatomicalSystem), Phenomenon and (Involves some NervousSystem)
- Instances:** (None listed)
- Target for Key:** (None listed)

The interface also includes a menu bar (File, Edit, View, Reasoner, Tools, Refactor, Window, Help), a search bar, and a status bar at the bottom indicating the reasoner is active and showing inferences.

# Current Tool of Choice: Justifications

The screenshot displays the OpenGALEN6 software interface. The main window title is "OpenGALEN6\_FULL\_WithPropertyChains (file:/home/patrick/Documents/Ontologies/OpenGALEN6/OpenGALEN6.s.owl)". The active ontology is "OpenGALEN6\_FULL\_WithPropertyChains (file:/home/patrick/Documents/Ontologies/OpenGALEN6/OpenGALEN6\_FULL\_WithPropertyChains.owl)".

The "Active ontology" panel shows a "Justifications" window titled "Explanation for NervousSystemSign SubClassOf AnatomicalSystemSign". The justification is structured as follows:

- Justifications
  - Black-Box Justifications
  - Regular justifications
    - 1 justification computed
    - Justification 1 with 5 axioms
      - NervousSystemSign **EquivalentTo** SignwhichinvolvesNervousSystem
        - SignwhichinvolvesNervousSystem **EquivalentTo** Sign and (involves **some** NervousSystem)
          - NervousSystem **SubClassOf** AnatomicalSystem
      - SignwhichinvolvesAnatomicalSystem **EquivalentTo** Sign and (involves **some** AnatomicalSystem)
        - AnatomicalSystemSign **EquivalentTo** SignwhichinvolvesAnatomicalSystem

The "Data properties" panel shows "Classes" with a "Class hierarchy" view. The "NervousSystemSign" class is highlighted. An "OK" button is visible at the bottom of the justification window. The "Reasoner active" status is shown in the bottom right corner.



# Justifications

**Justifications:** Minimal subsets entailing given subsumption

In practice often insufficient :

- can be large
- inferences often not obvious

Showing *how* to obtain the inference would be better

- simple reasoning steps leading to conclusion
- generally known as proof

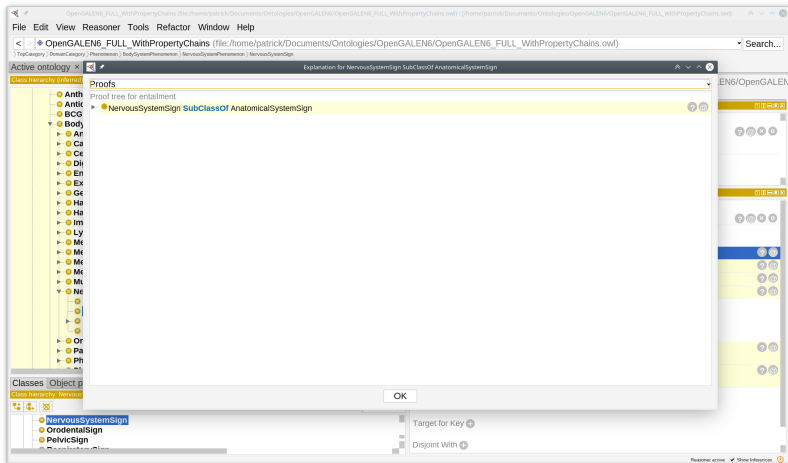
# Proofs for ELK in Protege

The screenshot displays the Protege interface with an active ontology window titled "Explanation for NervousSystemSign SubClassOf AnatomicalSystemSign". The main window shows a class hierarchy on the left and a justification window in the center. The justification window lists the following logical steps:

- Justifications
  - Black-Box Justifications
  - Regular justifications
    - 1 justification computed
      - Justification 1 with 5 axioms
        - NervousSystemSign **EquivalentTo** SignwhichinvolvesNervousSystem
          - SignwhichinvolvesNervousSystem **EquivalentTo** Sign and (involves **some** NervousSystem)
          - NervousSystem **SubClassOf** AnatomicalSystem
          - SignwhichinvolvesAnatomicalSystem **EquivalentTo** Sign and (involves **some** AnatomicalSystem)
          - AnatomicalSystemSign **EquivalentTo** SignwhichinvolvesAnatomicalSystem

The justification window also features an "OK" button and a "Target for Key" field. The bottom status bar indicates "Reasoner active" and "Show Inferences" is checked.

# Proofs for ELK in Protege



# Proofs for ELK in Protege

The screenshot shows the Protege software interface with the 'Reasoner' window open. The main window displays the ontology 'OpenGALEN6\_FULL\_WithPropertyChains'. The Reasoner window shows a 'Proofs' section with a 'Proof tree for entailment'. The tree structure is as follows:

- Proof tree for entailment
  - NervousSystemSign SubClassOf AnatomicalSystemSign
    - Class Hierarchy
      - NervousSystemSign SubClassOf Sign and (involves some AnatomicalSystem)
      - Sign and (involves some AnatomicalSystem) SubClassOf SignwhichinvolvesAnatomicalSystem
      - SignwhichinvolvesAnatomicalSystem SubClassOf AnatomicalSystemSign

# Proofs for ELK in Protege

The screenshot displays the Protege interface with a 'Proofs' dialog box open. The dialog title is 'Explanation for NervousSystemSign SubClassOf AnatomicalSystemSign'. The main content area shows a 'Proof tree for entailment' with the following structure:

- Proof tree for entailment
  - NervousSystemSign SubClassOf AnatomicalSystemSign
    - Class Hierarchy
      - NervousSystemSign SubClassOf Sign and (involves some AnatomicalSystem) (7/2)
      - Intersection Composition
        - NervousSystemSign SubClassOf Sign (7/2)
        - NervousSystemSign SubClassOf Involves some AnatomicalSystem (7/2)
      - Sign and (Involves some AnatomicalSystem) SubClassOf SignwhichinvolvesAnatomicalSystem (7/2)
      - SignwhichinvolvesAnatomicalSystem SubClassOf AnatomicalSystemSign (7/2)

The background shows the 'Active ontology' browser with a class hierarchy for 'NervousSystemSign' and a 'Classes' panel at the bottom.

# Proofs for ELK in Protege

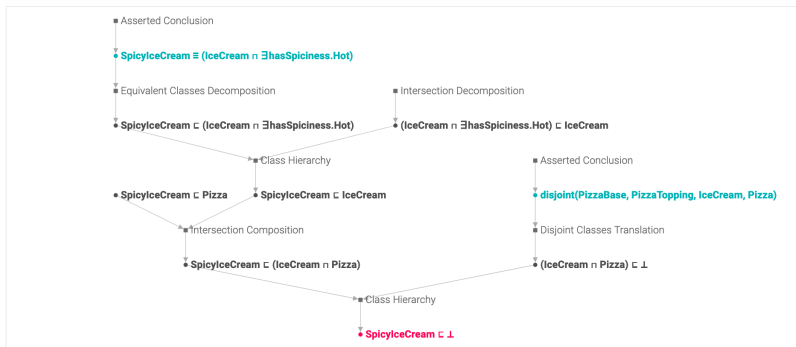
The screenshot shows the Protege interface with a 'Proofs' dialog box open. The dialog title is 'Explanation for NervousSystemSign SubClassOf AnatomicalSystemSign'. The main content is a tree structure under the heading 'Proofs'. The tree starts with 'Proof tree for entailment' leading to 'NervousSystemSign SubClassOf AnatomicalSystemSign'. This is followed by 'Class Hierarchy' leading to 'NervousSystemSign SubClassOf Sign and (involves some AnatomicalSystem)'. This node further branches into 'Intersection Composition' leading to 'NervousSystemSign SubClassOf Sign', which then leads to another 'Class Hierarchy' node. This second 'Class Hierarchy' node branches into 'NervousSystemSign SubClassOf SignwhichinvolvesNervousSystem', 'SignwhichinvolvesNervousSystem SubClassOf Sign and (involves some NervousSystem)', 'Sign and (involves some NervousSystem) SubClassOf Sign', and 'NervousSystemSign SubClassOf involves some AnatomicalSystem'. The 'Sign and (involves some NervousSystem) SubClassOf Sign' node further branches into 'Sign and (involves some AnatomicalSystem) SubClassOf SignwhichinvolvesAnatomicalSystem' and 'SignwhichinvolvesAnatomicalSystem SubClassOf AnatomicalSystemSign'. The dialog also has an 'OK' button and a 'Reasoner active' indicator at the bottom right.

# Proofs using Evonne (work in progress)

Proof

SELECT FILE

Stepwise Mode



C. Alrabbaa, F. Baader, R. Dachsetl, T. Flemisch, P. Koopmann: *Visualizing Proofs and the Modular Structure for Ontology Repair*, DL 2020.

# Proofs with ELK

- ELK using a *consequence-based* reasoning procedure
- ⇒ inferences performed using a calculus

$$\mathbf{R}_0 \frac{}{C \sqsubseteq C} \quad \mathbf{R}_\top \frac{}{C \sqsubseteq \top} \quad \mathbf{R}_\cap \frac{C \sqsubseteq D \cap E}{C \sqsubseteq D \quad C \sqsubseteq E}$$

$$\mathbf{R}_\cap^+ \frac{C \sqsubseteq D \quad C \sqsubseteq E}{C \sqsubseteq D \cap E} \quad \mathbf{R}_\exists \frac{C \sqsubseteq \exists r.D \quad D \sqsubseteq E}{C \sqsubseteq \exists r.E}$$

$$\mathbf{R}_\perp \frac{C \sqsubseteq \exists r.D \quad D \sqsubseteq \perp}{C \sqsubseteq \perp} \quad \mathbf{R}_\sqsubseteq \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O}$$

$$\mathbf{R}_\circ \frac{C_0 \sqsubseteq \exists r_1.C_1, C_1 \sqsubseteq \exists r_2.C_2 \dots C_{n-1} \sqsubseteq \exists r_n.C_n}{C_0 \sqsubseteq \exists r.C_n} : r_1 \circ \dots \circ r_n \sqsubseteq r \in \mathcal{O}$$

⇒ proofs generated as part of reasoning process



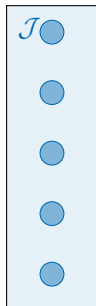
# Proofs For More Expressive DLs

- Currently, ELK is the only DL reasoner supporting proof generation
- ELK supports only a limited fragment of OWL, OWL EL
- More expressive reasoner often use other reasoning procedures
  - for a long time prominent: tableau reasoning
  - less convenient for understanding entailments
- Existing consequence-based reasoning for expressive DLs
  - often involved in complex systems
  - often combined with other reasoning paradigms
  - may use normal forms requiring different syntax
  - ⇒ generation of proofs not obvious
- Can we generate proofs without a calculus?

# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

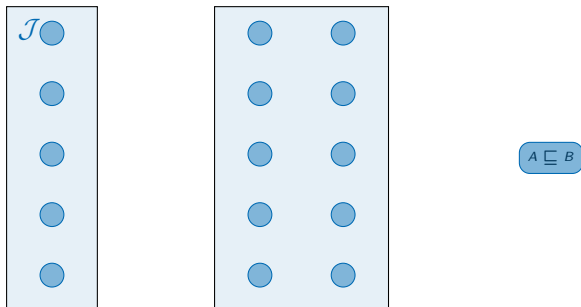
- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

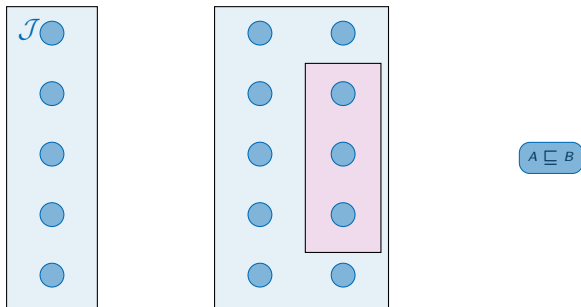
- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

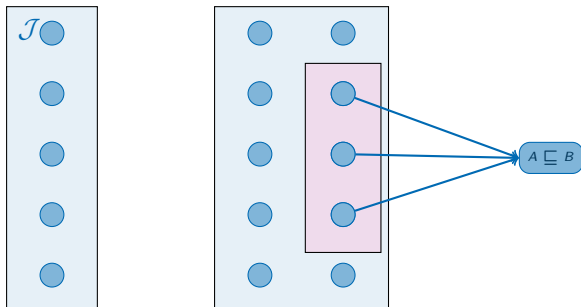
- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

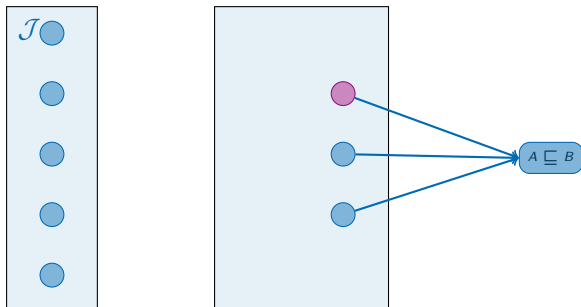
- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

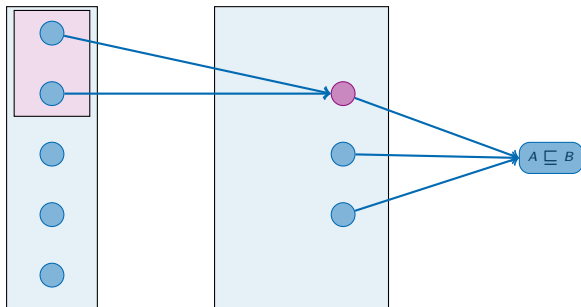
- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

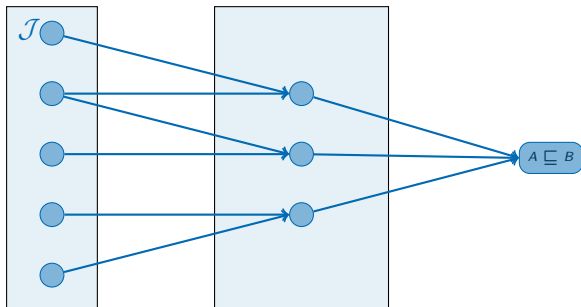
- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof

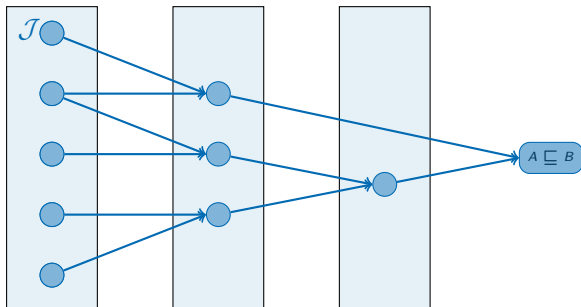




# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof



# Justification Based Proofs

## Justification-Based Proofs (Matthew Horridge 2011)

- derive intermediate steps between conclusion and justification
- consider all axioms of some predefined shapes
- justification-relation between allows to construct a proof
- involved ranking function allows to select axioms to be used in proof

## Advantage of approach:

- generates best proof according to ranking

## Disadvantage of approach:

- no clear inference principle
- hard to implement
- strongly depends on ranking function

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

$$b \quad a \vee b \quad \neg b \vee c \quad \neg b \vee \neg c \quad \neg a \vee c$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

$$b \quad a \vee b \quad \neg b \vee c \quad \neg b \vee \neg c \quad \neg a \vee c$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

$$b \quad \neg b \vee c \quad \neg b \vee \neg c \quad b \vee c$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

$$b \quad \neg b \vee c \quad \neg b \vee \neg c \quad b \vee c$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

$c \quad \neg c$



# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

$c \quad \neg c$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq C \quad C \sqsubseteq \exists r.D \quad \exists r.T \sqsubseteq B$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq C \quad C \sqsubseteq \exists r.D \quad \exists r.T \sqsubseteq B$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq C \quad C \sqsubseteq \exists r.T \quad \exists r.T \sqsubseteq B$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq C \quad C \sqsubseteq \exists r.T \quad \exists r.T \sqsubseteq B$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq C \quad C \sqsubseteq B$$



# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq C \quad C \sqsubseteq B$$

# Forgetting-Based Proofs

## Forgetting Based Proofs

- Idea from propositional resolution:

$$\frac{q_1 \vee p \quad q_2 \vee \neg p}{p_1 \vee p_2}$$

⇒ inference through elimination of  $p$

- Decide satisfiability by eliminating names one after the other:

⊥

- Idea: Use similar approach to prove axioms of form  $A \sqsubseteq B$

$$A \sqsubseteq B$$

# Forgetting

## Definition

Let  $\mathcal{O}$  be an ontology and  $X$  a predicate name. Then,  $\mathcal{O}^{-X}$  is a *result of forgetting  $X$*  iff

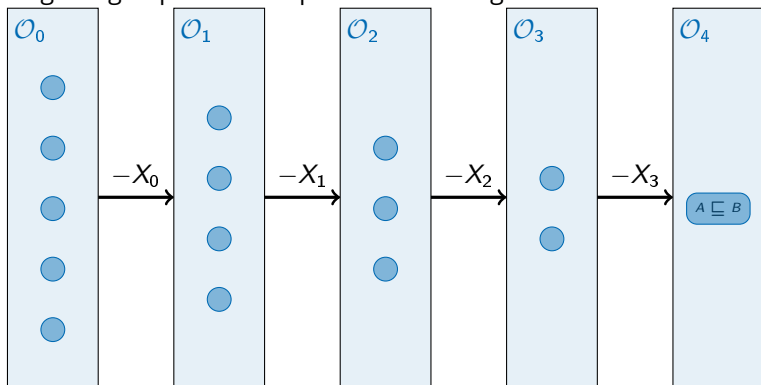
- $X$  does not occur in  $\mathcal{O}^{-X}$
- for every axiom  $\alpha$  in which  $X$  does not occur,  $\mathcal{O} \models \alpha$  iff  $\mathcal{O}^{-X} \models \alpha$

⇒ strongest ontology without  $X$  entailed by  $\mathcal{O}$

- which  $\alpha$  to preserve also depends on underlying DL

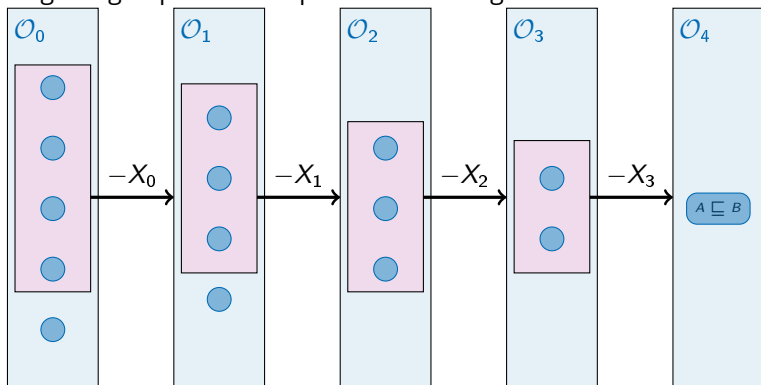
# Forgetting based proofs

- Use forgetting to produce sequence of ontologies



# Forgetting based proofs

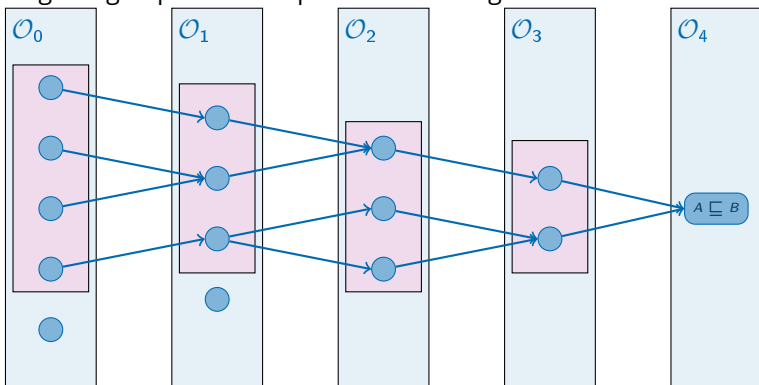
- Use forgetting to produce sequence of ontologies



- In each step, recompute justification for  $A \sqsubseteq B$

# Forgetting based proofs

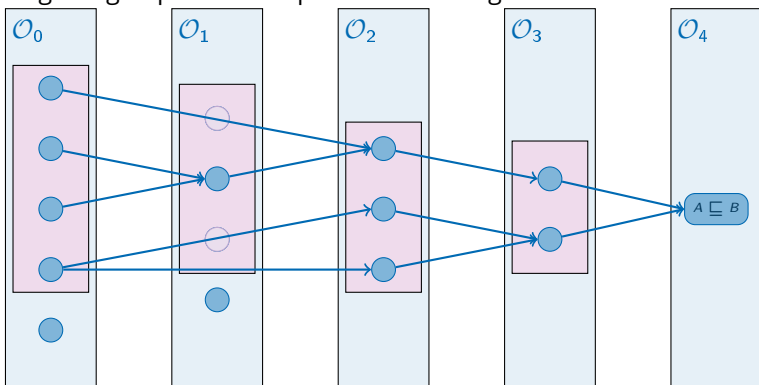
- Use forgetting to produce sequence of ontologies



- In each step, recompute justification for  $A \sqsubseteq B$
- Finally, reconstruct proof using justifications

# Forgetting based proofs

- Use forgetting to produce sequence of ontologies



- In each step, recompute justification for  $A \sqsubseteq B$
- Finally, reconstruct proof using justifications, skipping steps if it makes sense

# Forgetting Based Proof

$$\begin{array}{c} (C) \frac{A \sqsubseteq C}{(r) \frac{A \sqsubseteq \exists r.T}{A \sqsubseteq B}} \quad (D) \frac{C \sqsubseteq \exists r.D}{C \sqsubseteq \exists r.T} \quad \exists r.T \sqsubseteq B \end{array}$$



# Forgetting Order

Forgetting order, as well as selection of justification, affects proof

- Forgetting  $D$  first:

$$(C) \frac{A \sqsubseteq C \quad (D) \frac{C \sqsubseteq \exists r.D}{C \sqsubseteq \exists r.T}}{(r) \frac{A \sqsubseteq \exists r.T \quad \exists r.T \sqsubseteq B}{A \sqsubseteq B}}$$

- Forgetting  $r$  first:

$$(C) \frac{A \sqsubseteq C \quad (r) \frac{C \sqsubseteq \exists r.D \quad \exists r.T \sqsubseteq B}{C \sqsubseteq B}}{A \sqsubseteq B}$$

# Forgetting Order

To obtain **nicer** proofs *practically*, we process names using the following heuristics:

- role with non-trivial fillers last:
  - otherwise may hide most of inference:

$$(r) \frac{A \sqsubseteq \exists r.B \quad A \sqsubseteq \forall r.C \sqcup D \quad B \sqsubseteq \exists s.D \quad C \sqsubseteq \forall s.\neg D}{A \sqsubseteq D}$$

- unnested names first
  - delay complex inferences
- less frequent names first
  - delay expensive forgetting operations
  - (also used by existing forgetting procedures)

# Evaluation

- Implemented approach in modular fashion
  - easy exchange of different forgetting procedures, provided they produce OWL ontologies
  - easy comparison with proofs generated by ELK
    - Dijkstra-based search to extract shortest proof
  - use 2 forgetting tools in the evaluation
    - *ALCH* variant of LETHE 0.6
    - *ALCOI* variant of FAME 1.0<sup>1</sup>

---

<sup>1</sup>there is a much improved version FAME 2.0, but it often creates ontologies outside of the OWL-standard

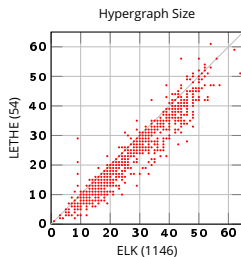
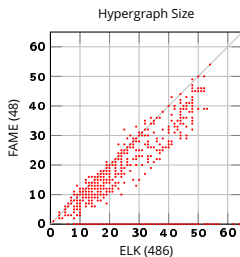
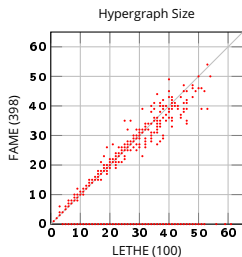
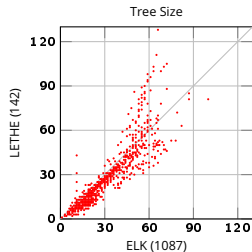
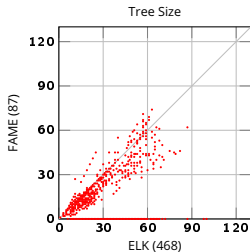
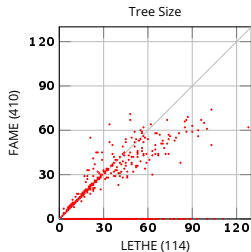
# Evaluation: Corpus

- Focus on proofs in  $\mathcal{ELH}$ 
  - to be able to compare with ELK
  - easier extraction of *justification patterns* (see below)
- Use ontologies from the OWL Reasoner Evaluation 2015, OWL EL Classification Track
  - well-balanced mix of ontologies from different repositories
- Extracted 1,573 *justification patterns*
  - all entailments of form  $A \sqsubseteq B$  or  $A \equiv B$
  - all justifications for these entailments
  - abstract away concept and role names
  - remove resulting duplicates

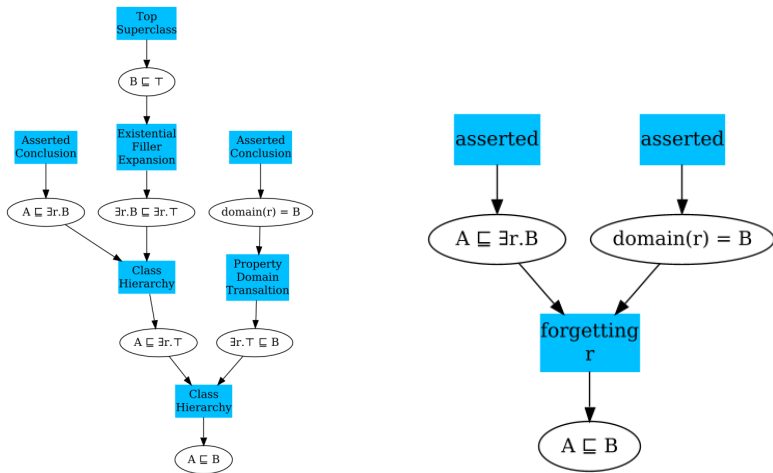
# Evaluation: Metrics

- Hypergraph-size
  - number of distinct axioms used in the proof
- Tree-size
  - sub-proofs count as often as they are used
- Justification Complexity
  - Matthew et al. 2013: "Toward cognitive support for OWL justifications"
  - attempt to measure cognitive complexity of justification
  - provides value for each proof step
  - we measured maximum and sum for each proof

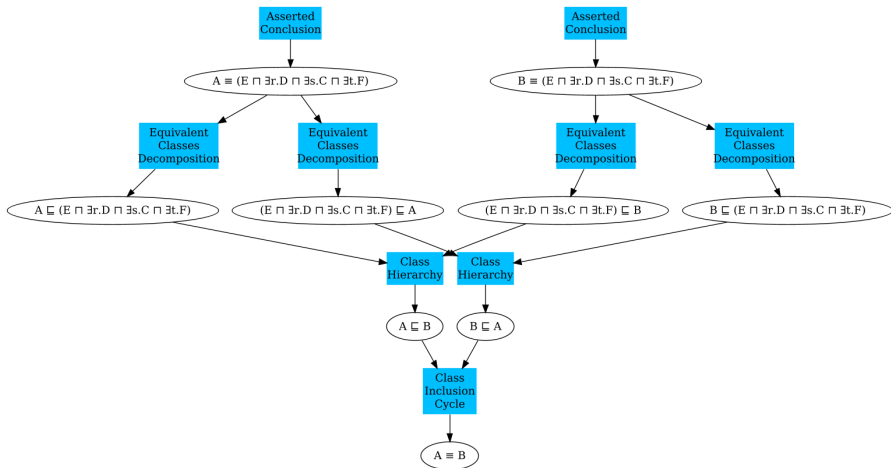
# Evaluation: Proof size



# Evaluation: ELK vs. Forgetting-Based Proofs

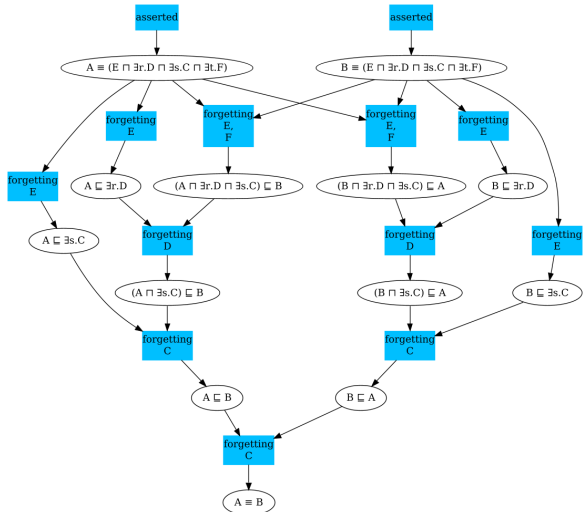


# Evaluation: ELK Proof

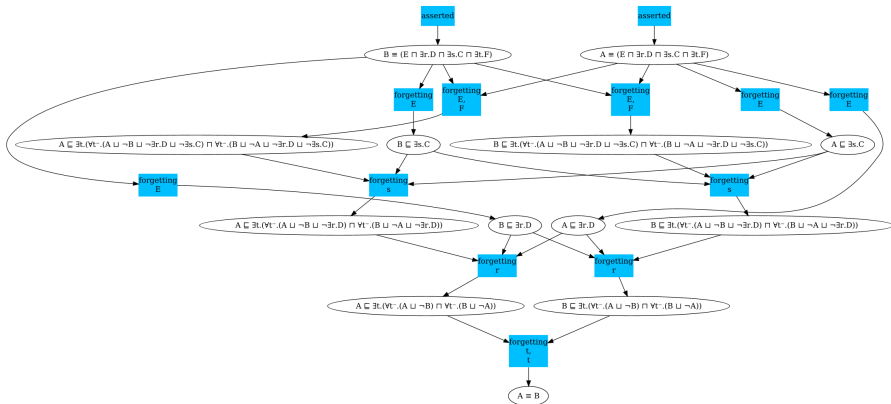




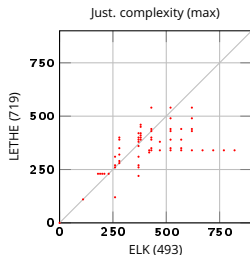
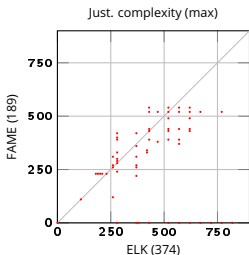
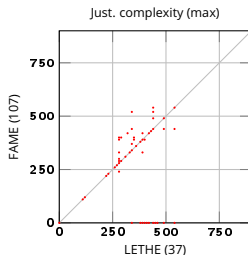
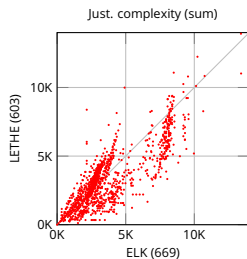
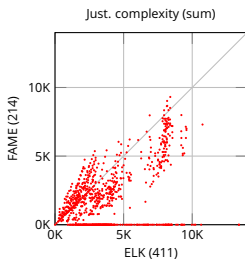
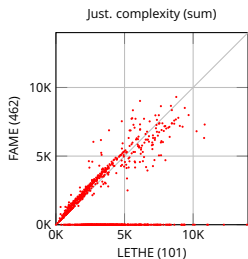
# Evaluation: LETHE Proof



# Evaluation: FAME Proof



# Evaluation: Justification Complexity



# Evaluation: Complexity of Inferences

- Both LETHE and FAME may use logical operators outside of  $\mathcal{EL}$
- Large number of distinct “inference rules” was used:
  - LETHE: 362 different rules
  - FAME: 281 different rules

# Conclusion

- New proof generation procedure based on forgetting
- Generate proof by repeated use of forgetting and justification
- Proofs for expressive DLs without calculus
- Can sometimes even compete with ELK
- Several possibilities to improve:
  - better heuristics on forgetting order or when to skip steps
  - dynamic selection of forgetting order
  - use learned “rules” to shorten proof computation times
  - integrate newer version of FAME