# Modularity in Ontologies: Approaches for Light-weight Description Logics

Thomas Schneider[1] *Dirk Walther*[2]

[1]Department of Computer Science, University of Bremen, Germany

[2]Faculty of Informatics, Technical University of Madrid, Spain

ESSLLI, 4 August 2011

# Plan for today

For light-weight DL-ontologies

- modularity and module extraction
- computing the logical difference of large-scale ontologies
- forgetting and uniform interpolation

# Modularity for Light-weight DLs

Logic-based modularity in light-weight DLs

- DL-Lite family
  - [Kontchakov, Wolter, Zakharyaschev, 2010]
- EL family
  - [Lutz, Wolter, 2010]

$\leadsto$ Here we focus on EL.

# Description Logic EL

EL is a fragment of ALC.

EL-syntax:

$$C, D = \top \mid A \mid C \sqcap D \mid \exists r.C$$

TBox $T$ is a finite set of concept inclusions $C \sqsubseteq D$.

Reasoning tasks:

- Satisfiability of EL-concept C wrt. EL-TBox $T$
  - trivial (tractable): always satisfiable in a one-point model
- Subsumption of EL-concepts $C, D$ wrt. EL-TBox $T$
  - tractable (decidable in polynomial time)

# Modularity reasoning for EL

- Deciding whether two EL-TBoxes are $\Sigma$-inseparable wrt. EL is ExpTime-complete.
- For EL-TBoxes, $\Sigma$-inseparability wrt. SO is undecidable.
- For EL-TBoxes, even $T \equiv_{\Sigma}^{SO} \emptyset$, (equivalently, whether

$$\{M_{|\Sigma} \mid M \models T\} = \text{ class of all } \Sigma\text{-models})$$

  is undecidable.
- EL has interpolation but (EL,EL) is not robust under replacement

Today, we consider EL-TBoxes of a particular form.

# EL-terminologies

**Definition**

An EL-TBox $T$ is a EL-terminology if

- every axiom is of the form $A \equiv C$, where $A$ is a concept name;
- no concept name $A$ occurs more than once on the left hand side of an axiom.

A EL-terminology $T$ is acyclic if no concept name refers to itself along definitions:

- let $A \prec_T X$ if there exists an axiom $A \equiv C$ in $T$ such that $X$ occurs in $C$.

Then $T$ is acyclic iff $\prec_T$ is acyclic (equivalently $\prec_T^+$ is irreflexive).

In a TBox $T$, we rewrite $A \sqsubseteq C$ into $A \equiv X \sqcap C$, where $X$ is fresh.

# Plan for EL-terminologies

- deciding '$\mathcal{T} \equiv_{\Sigma}^{SO} \emptyset$' in polynomial time,
  then $\mathcal{T}$ is safe ➺ *Wednesday's lecture*
- extract modules
- logical difference: comparing versions of ontologies
- forgetting and uniform interpolation

# Deciding '$T \equiv_\Sigma^{\mathsf{SO}} \emptyset$'

**Theorem**

The following problem can be solved in polynomial time:
given an acyclic EL-terminology $T$, decide whether

$$T \equiv_\Sigma^{SO} \emptyset.$$

For the proof, we distinguish two types of syntactic dependencies between Σ-symbols in $T$:

(a) direct: 'definition' of a Σ-symbol uses another Σ-symbol

(b) indirect: two Σ-symbols are 'defined' using common non-Σ-symbol

# Direct Σ-dependencies

Let $T$ be an acyclic EL-terminology.

(a) $T$ contains a direct Σ-dependency if there exist $A, X \in \Sigma$ such that $A \prec_T^+ X$.

**Theorem**

If an acyclic EL-terminology $T$ contains a direct Σ-dependency, then $T \not\equiv_\Sigma^{SO} \emptyset$.

Proof. Suppose $T$ contains a syntactic Σ-dependency $A \prec_\Sigma^+ X$. Take a interpretation $\mathcal{I}$ with $A^\mathcal{I} = \Delta^\mathcal{I}$ and $X^\mathcal{I} = \emptyset$. Then $\mathcal{I}$ can't be expanded to a model of $T$.

- Does not work for acyclic ALC-terminologies!
- From now on, we assume $T$ does not contain direct Σ-dependencies.

# Indirect Σ-dependencies

Decomposing an acyclic EL-terminology

- Let $T$ be an acyclic EL-terminology and $\Sigma$ a signature.
- Take partition

$$T = T_\Sigma \cup T',$$

where

$$T_\Sigma = \{A \equiv C \mid A \in \Sigma \text{ or } \exists B \in \Sigma,\ B \prec_T^+ A\}$$

- $T_\Sigma$ does not contain Σ-role names (there are no direct Σ-dependencies in $T$)

**Theorem**

If $\mathcal{I} \models T_\Sigma$, then there exists $\mathcal{J} \models T$ such that $\mathcal{J}_{|\Sigma} = \mathcal{I}_{|\Sigma}$.

Proof. Expand $\mathcal{I}$ inductively by setting $A^\mathcal{J} := C^\mathcal{J}$ for $A \equiv C \in T'$.

**Theorem**

Let $T$ be an acyclic EL-terminology without direct Σ-dependencies. Then the following conditions are equivalent:

1. $T \equiv_{\Sigma}^{SO} \emptyset$;
2. Every one-point Σ-interpr. can be expanded to a model of $T_{\Sigma}$.

Point 2 implies Point 1. Let $\mathcal{I}$ be an interpretation. As $T_{\Sigma}$ contains no Σ-roles, we may assume that Σ contains no roles. For each $d$ in $\mathcal{I}$, let $\mathcal{J}_{\{d\}} \models T_{\Sigma}$ be an expansion of $\mathcal{I}_{\{d\}}$. Then

$$\mathcal{J} = \bigcup_{d \in \mathcal{I}} \mathcal{J}_{\{d\}} \models T_{\Sigma}$$

and $\mathcal{J}$ is an expansion of $\mathcal{I}$.

# Polytime algorithm for $T \equiv_\Sigma^{SO} \emptyset$

To decide whether $T \equiv_\Sigma^{SO} \emptyset$, check

1. $T$ contains no direct $\Sigma$-dependencies;
2. every one point $\Sigma$-model can be expanded to a model of $T_\Sigma$.

Point 2 holds iff

For all $A \in \Sigma$,

$$\{X \mid A \prec_T^+ X\} \not\subseteq \{X \mid \exists B \in \Sigma \setminus \{A\},\ B \prec_T^+ X\}.$$

Observation: For acyclic ALC-terminologies without $\Sigma$-dependencies, one can decide $T \equiv_\Sigma^{SO} \emptyset$ by considering one point-models (then $\Pi_2^p$-complete).

# Module extraction

From deciding inseparability to module extraction.

- Given acyclic EL-terminology $T$ and signature $\Sigma$, the decision procedure extracts from $T$ the smallest $M \subseteq T$ such that

$$T \setminus M \equiv^{SO}_{\Sigma \cup \mathrm{sig}(M)} \emptyset.$$

➾ *then $T \setminus M$ is safe for $\Sigma \cup \mathrm{sig}(M)$ wrt. EL (Wednesday's lecture)*

- Equivalently, by robustness under replacement of (EL,SO),

$$M \equiv^{SO}_{\Sigma \cup \mathrm{sig}(M)} T.$$

➾ *then $M$ is a $\Sigma$-module in $T$ wrt. EL*

# Module extraction algorithm

Input: acyclic EL-terminology $T$ and signature $\Sigma$.

Output: smallest $M \subseteq T$ such that $T \setminus M \equiv^{SO}_{\Sigma \cup \mathrm{sig}(M)} \emptyset$.

Initialise: $M = \emptyset$, $\Sigma' = \Sigma$. Apply rules 1 and 2 exhaustively, preferring Rule 1.

1. collect direct dependencies
   if $A \in \Sigma'$, $A \equiv C \in T \setminus M$, and exists $X \in \Sigma'$ with $A \prec^{+}_{T \setminus M} X$,
   $$M := M \cup \{A \equiv C\}, \quad \Sigma' := \Sigma' \cup \mathrm{sig}(C).$$
2. collect indirect dependencies
   if $A \in \Sigma'$, $A \equiv C \in T \setminus M$, and
   $$\{X \mid A \prec^{+}_{T \setminus M} X\} \subseteq \{X \mid \exists B \in \Sigma' \setminus \{A\} \;\; B \prec^{+}_{T \setminus M} X\},$$
   then set
   $$M := M \cup \{A \equiv C\}, \quad \Sigma' := \Sigma' \cup \mathrm{sig}(C).$$

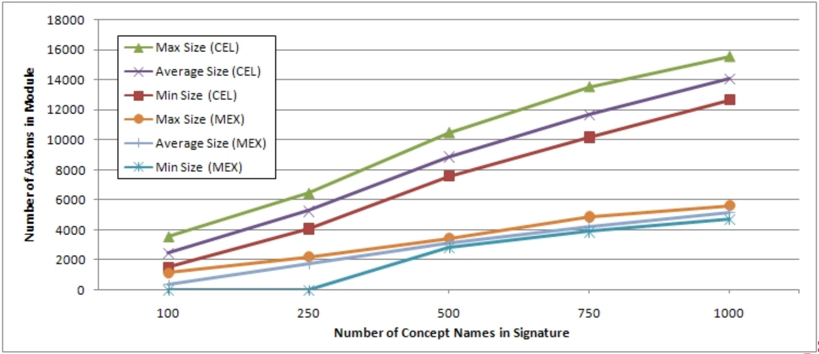
# Experiment with SNOMED CT

SNOMED CT:

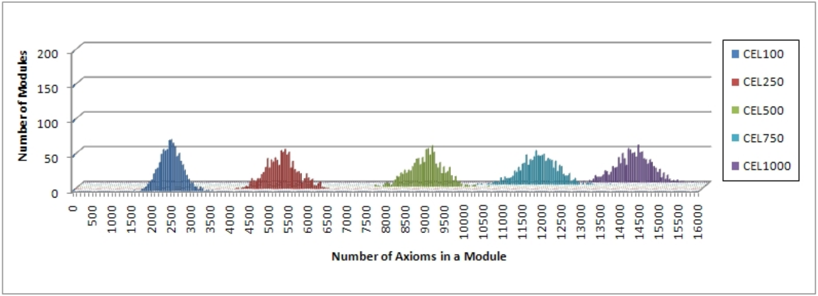
- Systematised Nomenclature of Medicine (Clinical Terms).
- $\sim 400,000$ terms
- used in health care etc. in the US, UK, Australia etc.
- an acyclic EL-terminology (+ role box):

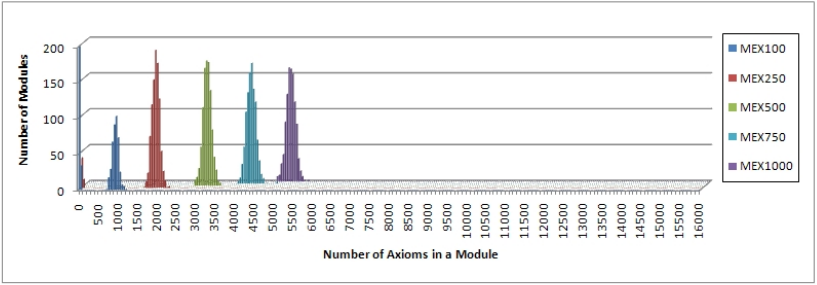
# Experiment: Extraction of modules from SNOMED CT

- MEX: prototype implementation of the algorithm above
- `http://www.csc.liv.ac.uk/~konev/software/`
- $\Sigma$ — randomly selected from SNOMED CT.
- 1000 samples for each signature size



# ⊥-Locality based vs. MEX Modules: Frequency



# Logical Difference: motivation

## Task

- given two versions $T_1$ and $T_2$ of an ontology and a signature $\Sigma$, compute “the difference” between $T_1$ and $T_2$ observable in $\Sigma$ in a query language $\mathcal{QL}$.

## Syntactical difference

- Many tools compute the syntactical difference between versions of texts and program code.
- But many syntactic differences do not affect the semantics of ontologies!
- Example:
  - $T_1 = \{A \sqsubseteq B_1 \sqcap B_2\}, \quad T_2 = \{A \sqsubseteq B_1, A \sqsubseteq B_2\}$
    $\Sigma = \{A, B_1, B_2\}$
  - Then $T_1 \neq T_2$, but $T_1 \equiv_{\Sigma}^{SO} T_2$.

### Structural difference

- extends syntactic diff by taking into account structural meta-information of distinct versions of ontologies
- regards ontologies as structured objects (e.g., taxonomy, set of RDF triplets, set of axioms)
- changes are structural operations (e.g., adding/deleting/extending/renaming classes)
- but:
  - syntax dependent and no formal semantics
  - tailored to applications of ontologies based on taxonomy
  - ontology based data access not captured

# Logical Difference

$T_1$ and $T_2$ ontologies, $\mathcal{QL}$ a query language, $\Sigma$ a signature.
The logical difference between $T_1$ and $T_2$ wrt. $(\mathcal{QL},\Sigma)$ is defined as

$$\mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_1, T_2) \cup \mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_2, T_1),$$

where

- $\mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_1, T_2) = \{\varphi \in \mathcal{QL} \mid T_1 \models \varphi, T_2 \not\models \varphi, \mathsf{sig}(\varphi) \in \Sigma\}$.
- $\mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_2, T_1) = \{\varphi \in \mathcal{QL} \mid T_2 \models \varphi, T_1 \not\models \varphi, \mathsf{sig}(\varphi) \in \Sigma\}$.

Observation: $\mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_1, T_2) \cup \mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_2, T_1) = \emptyset$ iff $T_1 \equiv^{\mathcal{QL}}_{\Sigma} T_2$.

Problem: How to present $\mathsf{Diff}^{\mathcal{QL}}_{\Sigma}(T_1, T_2)$ if it is non-empty?

# $\Sigma$-difference for EL-terminologies

Take query language $\mathcal{QL}_{EL}$ consisting of $C \sqsubseteq D$, where $C, D$ are EL-concepts. We also denote $\mathcal{QL}_{EL}$ simply as EL.
Set

$$\mathsf{Diff}_\Sigma(T_1, T_2) = \mathsf{Diff}^{\mathsf{EL}}_\Sigma(T_1, T_2).$$

Example of 'large' smallest elements in $\mathsf{Diff}_\Sigma(T_1, T_2)$:

- $T_2 = \emptyset$;
- $T_1 = \{A' \sqsubseteq B_0, A \equiv B_n\} \cup \{B_{i+1} \equiv \exists r.B_i \sqcap \exists s.B_i \mid i < n\}$;
- $\Sigma = \{A', A, r, s\}$.

For the minimal $C \sqsubseteq A \in \mathsf{Diff}_\Sigma(T_1, T_2)$ we have $|C| = 2^n$.

# $\Sigma$-difference for EL-terminologies

**Theorem**

If $(C \sqsubseteq D) \in \text{Diff}_\Sigma(T_1, T_2)$ then either

- $(A \sqsubseteq D_0) \in \text{Diff}_\Sigma(T_1, T_2)$ or
- $(C_0 \sqsubseteq A) \in \text{Diff}_\Sigma(T_1, T_2)$,

where $A$ is a concept name and

$A$, $C_0$ — subconcepts of $C$;

$D_0$, $A$ — subconcepts of $D$, resp.

In propositional EL: if $C \sqsubseteq A_1 \sqcap A_2 \in \text{Diff}_\Sigma(T_1, T_2)$, then

- $C \sqsubseteq A_1 \in \text{Diff}_\Sigma(T_1, T_2)$ or
- $C \sqsubseteq A_2 \in \text{Diff}_\Sigma(T_1, T_2)$.

# Compact representation of $\mathrm{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$

Let

- $\mathrm{diffL}_\Sigma(\mathcal{T}_1, \mathcal{T}_2) =$
  $$\left\{ A \in \Sigma \;\middle|\; \begin{array}{l} \text{there is a } \Sigma\text{-concept } C \text{ in EL s.t.} \\ \mathcal{T}_1 \models A \sqsubseteq C \text{ and } \mathcal{T}_2 \not\models A \sqsubseteq C \end{array} \right\}$$
- $\mathrm{diffR}_\Sigma(\mathcal{T}_1, \mathcal{T}_2) =$
  $$\left\{ A \in \Sigma \;\middle|\; \begin{array}{l} \text{there is a } \Sigma\text{-concept } C \text{ in EL s.t.} \\ \mathcal{T}_1 \models C \sqsubseteq A \text{ and } \mathcal{T}_2 \not\models C \sqsubseteq A \end{array} \right\}$$

$\mathrm{diffL}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ and $\mathrm{diffR}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ provide a list of concept names in $\Sigma$ about which $\mathcal{T}_1$ "says more" than $\mathcal{T}_2$.

# Σ-difference between EL-terminologies

**Theorem**

Let $T_1$ and $T_2$ be EL-terminologies and Σ a signature. Then

- $\mathrm{diffL}_\Sigma(T_1, T_2)$ and
- $\mathrm{diffR}_\Sigma(T_1, T_2)$

can be computed in polynomial time. In particular, Σ-inseparability wrt. EL is tractable.

# Tools

## CEX

- implementation of tractable algorithm computing $\text{DiffL}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ and $\text{DiffR}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ for acyclic $\mathcal{EL}$-terminologies [Konev, Walther, Wolter, 2008]
- `http://www.csc.liv.ac.uk/~konev/software/`

## OWLDiff

- CEX-diff for $\mathcal{EL}$-terminologies [Kremen, Smid, Kouba, 2011, to appear]
- plugins for Protégé and NeON toolkit
- `http://krizik.felk.cvut.cz/km/owldiff`

# Tools

## CEX2

- extends CEX to $\mathcal{ELH}^r$ (i.e. $\mathcal{EL}$ with role inclusion axioms and domain and range restrictions) without loosing tractability [Konev, Ludwig, Walther, Wolter, to appear]
- `http://www.csc.liv.ac.uk/~michel/software/cex2/`

## LogDiffViz

- Protégé plugin that calls CEX2 and visualises ontology versions and the differences as a hierarchical structure
- `http://www.csc.liv.ac.uk/~cs8wg/LogDiffViz/`

# CEX applied to SNOMED CT

Task: Compute the logical difference of two versions of SNOMED CT

- two versions:
  - SNOMED CT 2005 (SM-05):
    - 379 691 axioms
    - 09 February 2005
  - SNOMED CT 2006 (SM-06):
    - 389 472 axioms
    - 30 December 2006
- $\Sigma \subseteq \mathrm{sig}(\text{SM-05}) \cap \mathrm{sig}(\text{SM-06})$ randomly selected
- compute average (of time/memory/diff-size) over 20 samples for every signature size
- hardware: Intel Core 2 CPU at 2.13 GHz and 3 GB of RAM

# SM-05 vs SM-06

| Size of Σ | CEX: diff(SM-05,SM-06) Time (Sec.) | Memory (MByte) | $\lvert diffL_\Sigma \rvert$ | $\lvert diffR_\Sigma \rvert$ |
|---|---|---|---|---|
| 100 | 513.1 | 1 393.7 | 0.10 | 0.10 |
| 1 000 | 512.4 | 1 394.6 | 2.35 | 2.15 |
| 10 000 | 517.7 | 1 424.3 | 155.35 | 125.35 |
| 100 000 | 559.8 | 1 473.2 | 11 795.90 | 4 108.6 |

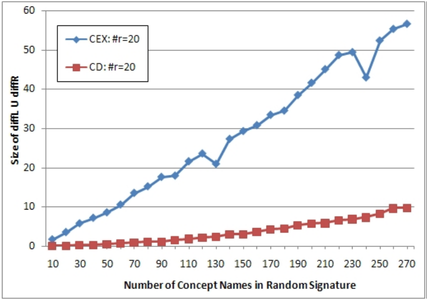
- Note: role box ignored

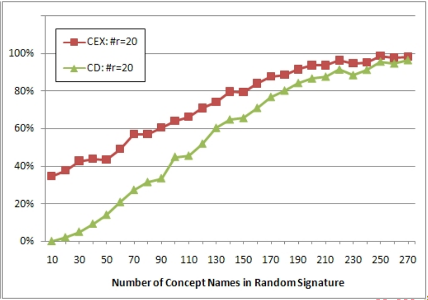
# Comparison on the Joint Signature

- diff(SM-05,SM-06) on
  $\Sigma = \text{sig(SM-05)} \cap \text{sig(SM-06)}$
  - 689 seconds
  - $|\text{diffL}_\Sigma| + |\text{diffR}_\Sigma| = 162010$
  - Class hierarchy comparison misses 32475 of them

# Comparing with classification

- Combined $\mathsf{diffL}_\Sigma(\emptyset, M)$ and $\mathsf{diffR}_\Sigma(\emptyset, M)$
  - $M$ is a subset of SM-05 containing $\sim 140,000$ axioms
  - $\Sigma$ — randomly selected from $M$ (incl. 20 role names)
  - avg. over 500 samples for each signature size
- Difference in class hierarchy



# CEX on MEX

Instead of computing $\text{diffL}_\Sigma(\mathcal{T}_1, \mathcal{T}_2) \cup \text{diffR}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ directly,

- first extract minimal $\Sigma$-modules $\mathcal{T}_1'$ and $\mathcal{T}_2'$ from $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively,
- then compute $\text{diffL}_\Sigma(\mathcal{T}_1', \mathcal{T}_2') \cup \text{diffR}_\Sigma(\mathcal{T}_1', \mathcal{T}_2')$.

| Size of $\Sigma$ | CEX: diff(SM-05,SM-06) | | | | CEX: diff(Mod'05,Mod'06) | |
|---|---|---|---|---|---|---|
| | Time (Sec.) | Memory (MByte) | $\lvert\text{diffL}_\Sigma\rvert$ | $\lvert\text{diffR}_\Sigma\rvert$ | Time (Sec.) | Memory (MByte) |
| 100 | 513.1 | 1 393.7 | 0.0 | 0.0 | 3.66 | 116.5 |
| 1 000 | 512.4 | 1 394.6 | 2.5 | 2.5 | 4.46 | 122.5 |
| 10 000 | 517.7 | 1 424.3 | 183.2 | 122.0 | 22.29 | 126.5 |
| 100 000 | 559.8 | 1 473.2 | 11 322.1 | 4 108.5 | 189.98 | 615.8 |
| 379741 | 790.0 | 1999.3 | 191714 | 684.1 | 1850.7 | 237044 |

# Forgetting Vocabulary

Forgetting vocabulary is eliminating that vocabulary from the ontology (involving a reformulation of the ontology).

Use-cases

- re-use: instead of whole ontology, use a potentially much smaller ontology resulting from forgetting
- predicate hiding: concealing confidential information in ontologies
- ontology summary: succinct presentation of what ontology states about non-forgotten vocabulary

The dual notion of forgetting is uniform interpolation.

Let $\mathcal{T}$ be a $\mathcal{EL}$-TBox and $\Sigma$ a signature. A TBox $\mathcal{T}'$ is called a uniform interpolant of $\mathcal{T}$ wrt. $\Sigma$ if the following holds:

- $\mathrm{sig}(\mathcal{T}') \subseteq \Sigma$;
- $\mathcal{T} \equiv_{\Sigma}^{\mathcal{EL}} \mathcal{T}'$.

**Theorem**

Let $\mathcal{T}_1', \mathcal{T}_2'$ be uniform interpolants of $\mathcal{T}_1$ and $\mathcal{T}_2$ wrt. $\Sigma$. The following conditions are equivalent:

- $\mathcal{T}_1 \equiv_{\Sigma}^{\mathcal{EL}} \mathcal{T}_2$;
- $\mathcal{T}_1'$ and $\mathcal{T}_2'$ are logically equivalent.

# EL-terminologies

**Theorem**

There exist an EL-terminology $T$ and $\Sigma$ such that there does not exist an uniform interpolant of $T$ wrt. $\Sigma$.

Proof. Let

$$T = \{A \sqsubseteq B, B \sqsubseteq \exists r.B\}, \quad \Sigma = \{A, r\}.$$

An infinite axiomatisation of the uniform interpolant is given by

$$\{A \sqsubseteq \underbrace{\exists r. \cdots \exists r}_{n}.\top \mid n \geq 1\}.$$

A finite $T_\Sigma$ does not exist (even in first-order logic).

# Acyclic EL-terminologies

**Theorem**

For acyclic EL-terminologies, uniform interpolants always exist. In the worst case, exponentially many axioms are required.

Proof of second part. Let

$$T = \{A \equiv B_1 \sqcap \cdots \sqcap B_n\} \cup \{A_{ij} \sqsubseteq B_i \mid 1 \leq i, j \leq n\}.$$

and

$$\Sigma = \{A\} \cup \{A_{ij} \mid 1 \leq i, j \leq n\}.$$

Then

$$T_\Sigma = \{A_{1j_1} \sqcap \cdots \sqcap A_{n,j_n} \sqsubseteq A \mid 1 \leq j_1, \ldots, j_n \leq n\}$$

is a minimal uniform interpolant. Note that $|T_\Sigma| = n^n$.

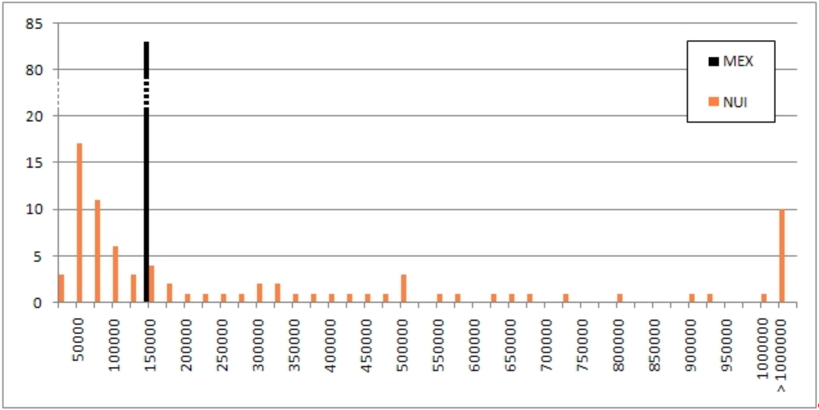
# Computing uniform interpolants for SNOMED CT and NCI

- NUI: prototype implementation computing uniform interpolants for acyclic EL-terminologies.
- $\Sigma$ — randomly selected from sig(SNOMED CT) and sig($NCI$), respectively.
- table shows success rate of NUI

| $\lvert\Sigma\rvert$ | SNOMED CT | $\lvert\Sigma\rvert$ | NCI |
|---|---|---|---|
| 2 000 | 100.0% | 5 000 | 97.0% |
| 3 000 | 92.2% | 10 000 | 81.1% |
| 4 000 | 67.0% | 15 000 | 72.0% |
| 5 000 | 60.0% | 20 000 | 59.2% |

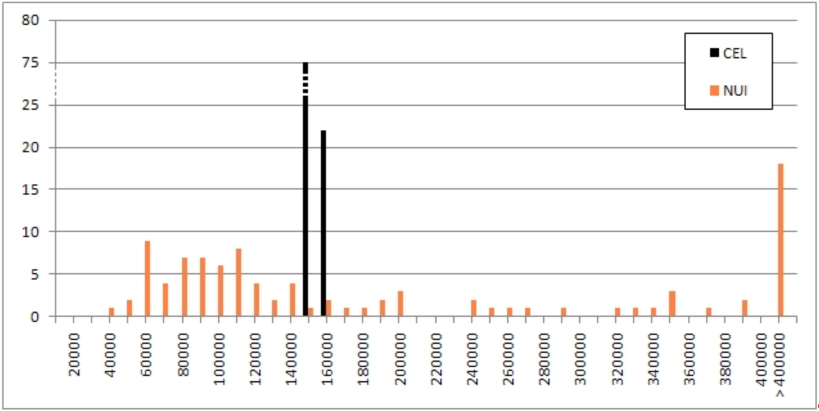
# Comparing the size of MEX-modules and Σ-interpolants

- Size distribution of MEX-modules and instance Σ-interpolants of SNOMED CT wrt. signatures containing 3 000 concept names and 20 role names



# Comparing the size of ⊤-local modules and Σ-interpolants

- Size distribution of CEL-modules and instance Σ-interpolants of NCI wrt. signatures containing 7 000 concept names and 20 role names



# Uniform interpolants beyond EL

**Theorem**

For ALC-TBoxes, uniform interpolants expressed in FOL do not always exist. [Ghilardi, Lutz, Wolter, 2006]

**Theorem**

For ALC-TBoxes, deciding the existence of uniform interpolants in ALC is 2ExpTime-complete. If they exist, uniform interpolants are most triple exponential in the size of the original TBox.
[Lutz, Wolter, 2011]

# Conclusion

We have shown for acyclic EL-terminologies:

- module extraction
- computing the logical difference of large-scale ontologies
- forgetting and uniform interpolation

# Course overview

5. Recent Advances/Current Work
   - Atomic decomposition
   - Signature decomposition, relevance of terms