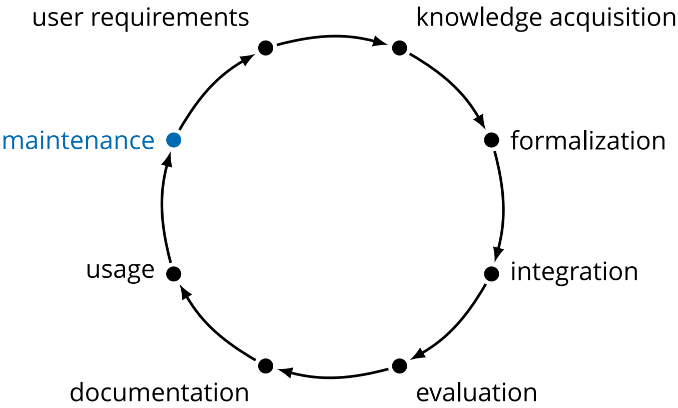
TECHNISCHE UNIVERSITÄT DRESDEN

Stefan Borgwardt
Institute of Theoretical Computer Science, Chair of Automata Theory

Logic-Based Ontology Engineering

Part 4: Ontology Maintenance

The Ontology Life Cycle



Introduction

We discuss automated techniques for supporting ontology engineers with:

Debugging:

- Determine the axioms responsible for an error, e.g., inconsistency
- Suggest ways of fixing the error
- Repair alignments, make them consistent and coherent

Modularization:

- Split the ontology into modules that have smaller vocabularies
- Improve performance of reasoning when restricted to a module
- Reuse modules in other ontologies
- Compute alignments for modules first, combine them later

Outline

4.1 Debugging

Finding Errors

Inconsistency and incoherence of an ontology $\mathcal{O}$ are easy to detect:
Check whether $\mathcal{O}$ entails $\top \sqsubseteq \bot$ or $A \sqsubseteq \bot$ for any concept name $A \in \mathbf{C}$.

Other errors are less obvious:

An old version of SNOMED CT (350,000+ axioms) entailed
AmputationOfFinger $\sqsubseteq$ AmputationOfHand.

Such errors are often found while using the ontology.

What to do once an error is found? Look at all 350,000+ axioms?

Justifications

We want to find out the axioms responsible for an (erroneous) entailment:

Given an ontology $\mathcal{O}$ and an axiom α with $\mathcal{O} \models \alpha$, a justification for α in $\mathcal{O}$ is a subset $\mathfrak{J} \subseteq \mathcal{O}$ such that

- $\mathfrak{J} \models \alpha$ and
- $\mathfrak{J}$ is a minimal set with this property, i.e., for every $\mathfrak{J}' \subset \mathfrak{J}$ it holds that $\mathfrak{J}' \not\models \alpha$.

We denote by $\mathsf{Just}_{\mathcal{O}}(\alpha)$ the set of all justifications for α in $\mathcal{O}$.

$\{A \equiv B \sqcap \exists r.C,\ \ B \sqsubseteq C,\ \ \exists r.\top \sqsubseteq D,\ \ D \sqsubseteq \neg C,\ \ A \sqsubseteq \neg D,\ \ C \sqcap \exists r^-.B \sqsubseteq \bot\}$
has two justifications for $A \sqsubseteq \bot$:
$\{A \equiv B \sqcap \exists r.C,\ \ \exists r.\top \sqsubseteq D,\ \ A \sqsubseteq \neg D\}$
$\{A \equiv B \sqcap \exists r.C,\ \ C \sqcap \exists r^-.B \sqsubseteq \bot\}$

Each justification provides an explanation for the error α.

Justifications for Incoherence

Given an incoherent ontology $\mathcal{O}$, a justification for incoherence of $\mathcal{O}$ is a minimal subset $\mathfrak{J} \subseteq \mathcal{O}$ that is incoherent.

We denote by $\mathsf{Just}_{\mathcal{O}}(\bot)$ the set of all justifications for incoherence of $\mathcal{O}$.

Each $\mathfrak{J} \in \mathsf{Just}_{\mathcal{O}}(\bot)$ explains the unsatisfiability of at least one $A \in \mathbf{C}$:

$$\mathsf{Just}_{\mathcal{O}}(\bot) \subseteq \bigcup_{\mathcal{O} \models A \sqsubseteq \bot} \mathsf{Just}_{\mathcal{O}}(A \sqsubseteq \bot)$$

In general, not every justification for $A \sqsubseteq \bot$ is a justification for incoherence:

$$\begin{aligned} \mathcal{O} &= \{A \sqsubseteq B,\ B \sqsubseteq \bot\} \\ \mathsf{Just}_{\mathcal{O}}(A \sqsubseteq \bot) &= \{\{A \sqsubseteq B,\ B \sqsubseteq \bot\}\} \\ \mathsf{Just}_{\mathcal{O}}(\bot) &= \{\{B \sqsubseteq \bot\}\} \end{aligned}$$

Algorithms to compute $\mathsf{Just}_{\mathcal{O}}(A \sqsubseteq \bot)$ can often be easily adapted to compute $\mathsf{Just}_{\mathcal{O}}(\bot)$.

Diagnoses

We also want to find out which axioms have to be removed to fix the error:

Given an ontology $\mathcal{O}$ and an axiom α with $\mathcal{O} \models \alpha$, a diagnosis for α in $\mathcal{O}$ is a subset $\mathfrak{D} \subseteq \mathcal{O}$ such that

- $\mathcal{O} \setminus \mathfrak{D} \not\models \alpha$ and
- $\mathfrak{D}$ is a minimal set with this property, i.e., for every $\mathfrak{D}' \subset \mathfrak{D}$ it holds that $\mathcal{O} \setminus \mathfrak{D}' \models \alpha$.

We denote by $\mathrm{Diag}_{\mathcal{O}}(\alpha)$ the set of all diagnoses for α in $\mathcal{O}$.

$\{A \equiv B \sqcap \exists r.C,\ \ B \sqsubseteq C,\ \ \exists r.\top \sqsubseteq D,\ \ D \sqsubseteq \neg C,\ \ A \sqsubseteq \neg D,\ \ C \sqcap \exists r^-.B \sqsubseteq \bot\}$
has three diagnoses for $A \sqsubseteq \bot$:

$\{A \equiv B \sqcap \exists r.C\}$

$\{\exists r.\top \sqsubseteq D,\ \ C \sqcap \exists r^-.B \sqsubseteq \bot\}$

$\{A \sqsubseteq \neg D,\ \ C \sqcap \exists r^-.B \sqsubseteq \bot\}$

Justifications vs. Diagnoses

There is a close connection between justifications and diagnoses: To fix the error, we need to remove at least one axiom from every justification.

$$\{A \equiv B \sqcap \exists r.C,\ \ B \sqsubseteq C,\ \ \exists r.\top \sqsubseteq D,\ \ D \sqsubseteq \neg C,\ \ A \sqsubseteq \neg D,\ \ C \sqcap \exists r^{-}.B \sqsubseteq \bot\}$$

Justifications:

$$\{A \equiv B \sqcap \exists r.C,\ \ \exists r.\top \sqsubseteq D,\ \ A \sqsubseteq \neg D\}$$
$$\{A \equiv B \sqcap \exists r.C,\ \ C \sqcap \exists r^{-}.B \sqsubseteq \bot\}$$

Diagnoses:

$$\{A \equiv B \sqcap \exists r.C\}$$
$$\{\exists r.\top \sqsubseteq D,\ \ C \sqcap \exists r^{-}.B \sqsubseteq \bot\}$$
$$\{A \sqsubseteq \neg D,\ \ C \sqcap \exists r^{-}.B \sqsubseteq \bot\}$$

Minimal Hitting Sets

Given a finite universe U and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_n\}$ of U, a hitting set for $\mathcal{S}$ in U is a subset $H \subseteq U$ such that $H \cap S_i \neq \emptyset$ for all $i \in \{1, \ldots, n\}$.

A hitting set is minimal if no proper subset of it is a hitting set.

We denote by $\mathsf{MHS}_U(\mathcal{S})$ the set of all minimal hitting sets for $\mathcal{S}$ in U.

For us, the universe is $\mathcal{O}$ and $\mathcal{S}$ is the set of all justifications.

Lemma (Minimal Hitting Sets of Justifications)

Given an ontology $\mathcal{O}$ and an axiom α with $\mathcal{O} \models \alpha$, we have

$$\mathsf{Diag}_{\mathcal{O}}(\alpha) = \mathsf{MHS}_{\mathcal{O}}(\mathsf{Just}_{\mathcal{O}}(\alpha)).$$

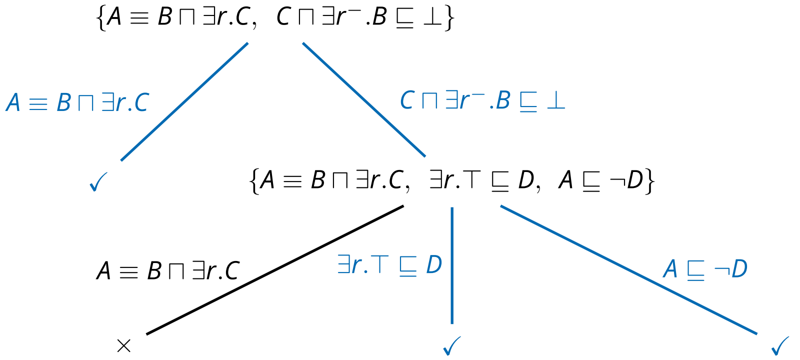
Proof: Blackboard. □

Exercise: Prove that $\mathsf{Just}_{\mathcal{O}}(\alpha) = \mathsf{MHS}_{\mathcal{O}}(\mathsf{Diag}_{\mathcal{O}}(\alpha))$.

Hitting Set Trees

To efficiently compute $\mathrm{MHS}_{\mathcal{O}}(\mathrm{Just}_{\mathcal{O}}(\alpha))$, we construct a hitting set tree (HST):

$$\mathrm{Just}_{\mathcal{O}}(\alpha) = \\ \{\{A \equiv B \sqcap \exists r.C,\ \ \exists r.\top \sqsubseteq D,\ \ A \sqsubseteq \neg D\},\ \ \{A \equiv B \sqcap \exists r.C,\ \ C \sqcap \exists r^{-}.B \sqsubseteq \bot\}\}$$



Hitting Set Trees

To efficiently compute $\mathrm{MHS}_U(\mathcal{S})$, we construct a hitting set tree (HST):

- Nodes of the tree are labeled with $S \in \mathcal{S}$, edges are labeled with $e \in U$.
- Given a node v, the set of edge labels on the path from the root node to v is denoted by $H(v) \subseteq U$.
- The root node is labeled with an arbitrary $S \in \mathcal{S}$.
- Every node labeled with some $S \in \mathcal{S}$ has an outgoing edge labeled with e, for every $e \in S$.
- Every new node v has a label $S \in \mathcal{S}$ such that $S \cap H(v) = \emptyset$.
  If there is no such S, then $H(v)$ is a hitting set for $\mathcal{S}$ in U.

Optimizations:

- If the tree already contains a node label S that is disjoint with the current $H(v)$, then reuse S as the label for v. This avoids unnecessary access to $\mathcal{S}$.
- Explore the tree breadth-first to find smaller hitting sets first.

The Hitting Set Tree Algorithm

Algorithm (HSTAlgorithm (Reiter, 1987))

Input: Universe U, collection of sets $\mathcal{S}$

Output: The set $\mathsf{MHS}_U(\mathcal{S})$

- Initialize a tree T with a single, unlabeled root node
- While there is an unlabeled node v in T:
  - Choose such a node v of minimal depth in T
  - If there is a node w in T labeled with $\checkmark$ such that $H(w) \subseteq H(v)$, then label v with $\times$
  - Otherwise, if there is a set $S \in \mathcal{S}$ such that $S \cap H(v) = \emptyset$, then
    - Label v with S
    - For each $e \in S$, create a successor w of v in T and label the edge from v to w with e
  - Otherwise, label v with $\checkmark$
- Return the set of all sets $H(v)$ for which v is labeled with $\checkmark$

Correctness of the HST Algorithm

The algorithm is nondeterministic: For each node x, there may be several possible labels $S \in \mathcal{S}$ with $S \cap H(x) = \emptyset$.

This is "don't care" nondeterminism: We can choose any such S.

Lemma (Correctness of HSTAlgorithm)

Given a set U, and a collection of its subsets $\mathcal{S}$, we have

$$\textbf{HSTAlgorithm}(U, \mathcal{S}) = \mathrm{MHS}_U(\mathcal{S}).$$

Proof: Blackboard. $\square$

We can use this algorithm to compute $\mathrm{Diag}_{\mathcal{O}}(\alpha)$ from $\mathrm{Just}_{\mathcal{O}}(\alpha)$ (or $\mathrm{Just}_{\mathcal{O}}(\alpha)$ from $\mathrm{Diag}_{\mathcal{O}}(\alpha)$).

Computing Justifications

How can we compute $\mathsf{Just}_{\mathcal{O}}(\alpha)$ in the first place?

> Black-box algorithms: Use a reasoner for deciding $\mathcal{O} \models \alpha$ as a “black box”, and construct justifications by a series of calls to the reasoner.

Such a black-box approach is built into Protégé, and can be used with any reasoner.

> Glass-box algorithms: Extend an existing reasoning algorithm for checking $\mathcal{O} \models \alpha$ to “trace” the axioms from $\mathcal{O}$ that are used to derive α.

This is generally faster, but requires deep knowledge of the reasoning algorithm, and has to be implemented for each reasoner separately.

Black-Box Algorithms

A naive black-box algorithm for computing $\mathsf{Just}_{\mathcal{O}}(\alpha)$:
Check for all subsets $\mathfrak{J} \subseteq \mathcal{O}$ whether they entail the error α (using the black-box reasoner), and then remove the non-minimal ones.

This algorithm needs exponentially many calls to the reasoner.
Can we do better?

No. In general, there are exponentially many justifications, so verifying all of them already takes exponential time:

$$\begin{aligned}\{A \sqsubseteq B_1 \sqcap C_1,\ B_1 \sqsubseteq B_2 \sqcap C_2, \dots, B_{n-1} \sqsubseteq B_n \sqcap C_n,\ B_n \sqsubseteq D\\ C_1 \sqsubseteq B_2 \sqcap C_2, \dots, C_{n-1} \sqsubseteq B_n \sqcap C_n,\ C_n \sqsubseteq D\}\end{aligned}$$

has 2^n justifications for $A \sqsubseteq D$.

Note: Justifications are sensitive to the syntactical shape of the axioms! The following equivalent ontology has only one justification for $A \sqsubseteq D$:

$$\{A \sqsubseteq B_1 \sqcap C_1,\ B_1 \sqcup C_1 \sqsubseteq B_2 \sqcap C_2,\ \dots,\ B_{n-1} \sqcup C_{n-1} \sqsubseteq B_n \sqcap C_n,\ B_n \sqcup C_n \sqsubseteq D\}$$

A Black-Box Algorithm for Single Justifications (I)

"Binary search" for a justification for α in $\mathcal{O}$: **SingleJustification**$(\emptyset, \mathcal{O}, \alpha)$

Algorithm (SingleJustification)

Input: Ontologies $\mathcal{O}_1, \mathcal{O}_2$, axiom α such that $\mathcal{O}_1 \not\models \alpha$ and $\mathcal{O}_1 \cup \mathcal{O}_2 \models \alpha$

Output: A minimal subset $\mathcal{O}_2' \subseteq \mathcal{O}_2$ such that in $\mathcal{O}_1 \cup \mathcal{O}_2' \models \alpha$

- If $|\mathcal{O}_2| = 1$, then return $\mathcal{O}_2$
- Split $\mathcal{O}_2$ into $\mathcal{O}_l$ and $\mathcal{O}_r$
- If $\mathcal{O}_1 \cup \mathcal{O}_x \models \alpha$ for $x \in \{l, r\}$, return **SingleJustification**$(\mathcal{O}_1, \mathcal{O}_x, \alpha)$
- $\mathcal{O}_l' :=$ **SingleJustification**$(\mathcal{O}_1 \cup \mathcal{O}_r, \mathcal{O}_l, \alpha)$
- $\mathcal{O}_r' :=$ **SingleJustification**$(\mathcal{O}_1 \cup \mathcal{O}_l', \mathcal{O}_r, \alpha)$
- Return $\mathcal{O}_l' \cup \mathcal{O}_r'$

(Horridge, Parsia, Sattler, 2009)

A Black-Box Algorithm for Single Justifications (II)

- The algorithm recursively splits $\mathcal{O}$ into two halves $\mathcal{O}_l, \mathcal{O}_r$, and tries to find a justification in each half separately.
- If none of the halves entails α, it first finds a minimal subset of $\mathcal{O}_l$ that, together with $\mathcal{O}_r$, still entails α, and afterwards minimizes $\mathcal{O}_r$.
- The intuition is that justifications are usually much smaller than the whole ontology. So usually one half of the ontology stills contain a whole justification, and we can discard the other half.

Lemma (Correctness of SingleJustification)

Given an ontology $\mathcal{O}$ and an axiom α with $\mathcal{O} \models \alpha$, we have

$$\textbf{SingleJustification}(\emptyset, \mathcal{O}, \alpha) \in \mathsf{Just}_{\mathcal{O}}(\alpha).$$

Proof: Blackboard. □

A Black-Box Algorithm for All Justifications (I)

Computing a single justification for α is not enough for repairing the error, because there may be other causes for the entailment of α.

An efficient algorithm to compute all justifications in $\mathsf{Just}_{\mathcal{O}}(\alpha)$ based on HSTAlgorithm (Horridge, Parsia, Sattler, 2009):

- To find all hitting sets (diagnoses), it has to enumerate all justifications
- Needs method to compute a single justification for a subontology of $\mathcal{O}$
  - Either black-box or glass-box
  - Optimizations reduce the number of calls to this subprocedure

We instantiate the general **HSTAlgorithm** for justifications and diagnoses:

- Nodes v are now labeled with justifications $\mathfrak{J}$
- Edges are labeled with axioms α'
- $\mathfrak{D}(v)$ denotes the set of all edge labels on the path from the root to v
- We are not interested in diagnoses, but in justifications

A Black-Box Algorithm for All Justifications (II)

Algorithm (AllJustifications)

Input: Ontology $\mathcal{O}$, axiom α such that $\mathcal{O} \models \alpha$

Output: The set $\mathsf{Just}_{\mathcal{O}}(\alpha)$

- Initialize a tree T with a single, unlabeled root node
- While there is an unlabeled node v in T:
  - Choose such a node v of minimal depth in T
  - If there is a node w in T labeled with ✓ such that $\mathfrak{D}(w) \subseteq \mathfrak{D}(v)$, then label v with $\times$
  - Otherwise, if $\mathcal{O} \setminus \mathfrak{D}(v) \models \alpha$, then
    - Label v with **SingleJustification**$(\emptyset, \mathcal{O} \setminus \mathfrak{D}(v), \alpha)$
    - For each axiom $\alpha' \in \mathfrak{J}$, create a successor w of v in T and label the edge from v to w with α'
  - Otherwise, label v with ✓
- Return the set of all node labels in T (except ✓ and $\times$)

A Black-Box Algorithm for All Justifications (III)

Lemma (Correctness of AllJustifications)

Given an ontology $\mathcal{O}$ and an axiom α with $\mathcal{O} \models \alpha$, we have

$$\textbf{AllJustifications}(\mathcal{O}, \alpha) = \mathsf{Just}_{\mathcal{O}}(\alpha).$$

Proof: Blackboard. □

- If the tree already contains a justification $\mathfrak{J}$ with $\mathfrak{D}(v) \cap \mathfrak{J} = \emptyset$, it can be reused as the label for v
- **SingleJustification** is then only called once for each justification $\mathfrak{J} \in \mathsf{Just}_{\mathcal{O}}(\alpha)$
- A plugin that implements **AllJustifications** is included in Protégé (“Explanation Workbench”)

A Glass-Box Approach: Pinpointing

Glass-box approaches extend existing reasoning algorithms to “trace” the axioms from $\mathcal{O}$ that are used to derive α.

One class of techniques produces so-called pinpointing formulas: formulas in propositional logic that use propositional variables to represent the axioms in $\mathcal{O}$, and encode which combinations of axioms entail α.

A labeling function *lab* for $\mathcal{O}$ assigns each axiom $\beta \in \mathcal{O}$ a unique label $lab(\beta)$. The set of all labels of axioms in $\mathcal{O}$ is $lab(\mathcal{O})$.

$$\begin{array}{ccccccccc}
\{ & A \equiv B \sqcap \exists r.C, & B \sqsubseteq C, & \exists r.\top \sqsubseteq D, & D \sqsubseteq \neg C, & A \sqsubseteq \neg D, & C \sqcap \exists r^-.B \sqsubseteq \bot & \} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\{ & p_1, & p_2, & p_3, & p_4, & p_5, & p_6 & \}
\end{array}$$

Monotone Boolean Formulas

A monotone Boolean formula φ over $\mathit{lab}(\mathcal{O})$ is a propositional formula that uses the labels $\mathit{lab}(\mathcal{O})$ as propositional variables, and uses only the connectives $\land$, $\lor$, and `true` (no negation).

A monotone Boolean formula over $\{p_1, \dots, p_6\}$: $p_1 \land \big((p_3 \land p_5) \lor p_6\big)$

A valuation over $\mathit{lab}(\mathcal{O})$ is a subset $V \subseteq \mathit{lab}(\mathcal{O})$.

It satisfies φ if φ evaluates to `true` after replacing all variables in V by `true`, and replacing all variables in $\mathit{lab}(\mathcal{O}) \setminus V$ by `false`.

A minimal satisfying valuation of φ is a valuation V that satisfies φ, and for which there exists no valuation $V' \subset V$ that also satisfies φ.

The valuation $\{p_1, p_3, p_5, p_6\}$ satisfies $p_1 \land \big((p_3 \land p_5) \lor p_6\big)$.

$\{p_1, p_6\}$ is a minimal satisfying valuation of $p_1 \land \big((p_3 \land p_5) \lor p_6\big)$.

Pinpointing Formulas

Given two monotone Boolean formulas φ, ψ over $lab(\mathcal{O})$, we say that φ implies ψ if all valuations over $lab(\mathcal{O})$ that satisfy φ also satisfy ψ.

Valuations correspond to subontologies of $\mathcal{O}$:

Given a valuation V over $lab(\mathcal{O})$, we define $\mathcal{O}_V := \{\beta \in \mathcal{O} \mid lab(\beta) \in V\}$. We say that $\mathcal{O}_V$ is induced by V.

We have $\mathcal{O}_{lab(\mathcal{O}')} = \mathcal{O}'$ for all $\mathcal{O}' \subseteq \mathcal{O}$.

Given an axiom α with $\mathcal{O} \models \alpha$, the monotone Boolean formula φ over $lab(\mathcal{O})$ is a pinpointing formula for α in $\mathcal{O}$ if, for all valuations $V \subseteq lab(\mathcal{O})$, it holds that V satisfies φ iff $\mathcal{O}_V \models \alpha$.

Example: Pinpointing Formula

Recall $\mathcal{O}$ and $lab(\mathcal{O})$:

$$\begin{array}{ccccccccccccc} \{A \equiv B \sqcap \exists r.C, & & B \sqsubseteq C, & & \exists r.\top \sqsubseteq D, & & D \sqsubseteq \neg C, & & A \sqsubseteq \neg D, & & C \sqcap \exists r^-.B \sqsubseteq \bot\} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \{\ p_1, & & p_2, & & p_3, & & p_4, & & p_5, & & p_6\ \} \end{array}$$

$$p_1 \wedge \big((p_3 \wedge p_5) \vee p_6\big)$$

is a pinpointing formula for $A \sqsubseteq \bot$ in $\mathcal{O}$ with minimal satisfying valuations

$V_1 = \{p_1, p_3, p_5\}$ and $V_2 = \{p_1, p_6\}$.

These valuations induce the two justifications

$\mathcal{O}_{V_1} = \{A \equiv B \sqcap \exists r.C,\ \exists r.\top \sqsubseteq D,\ A \sqsubseteq \neg D\}$ and
$\mathcal{O}_{V_2} = \{A \equiv B \sqcap \exists r.C,\ C \sqcap \exists r^-.B \sqsubseteq \bot\}$.

From Pinpointing Formulas to Justifications

Recall:

> Given an axiom α with $\mathcal{O} \models \alpha$, the monotone Boolean formula φ over $\mathit{lab}(\mathcal{O})$ is a pinpointing formula for α in $\mathcal{O}$ if, for all valuations $V \subseteq \mathit{lab}(\mathcal{O})$, it holds that V satisfies φ iff $\mathcal{O}_V \models \alpha$.

Lemma (Pinpointing Formulas)

If φ is a pinpointing formula for α in $\mathcal{O}$, then the justifications for α in $\mathcal{O}$ are exactly the subontologies of $\mathcal{O}$ that are induced by the minimal satisfying valuations of φ.

Proof: Exercise. $\square$

Pinpointing Formulas in $\mathcal{ELH}$

We now extend a reasoning algorithm for $\mathcal{ELH}$ to compute all justifications via the pinpointing formula.

(Baader, Peñaloza, Suntisrivaraporn, 2007)

To simplify the description, without loss of generality we

- consider as consequences only subsumptions $A \sqsubseteq B$ between concept names $A, B \in \mathbf{C}$
- assume that the ABox is empty
- assume that the TBox is in normal form:

An $\mathcal{ELH}$ TBox is in normal form if all its GCIs have one of the forms

$$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \qquad A \sqsubseteq \exists r.B \qquad \exists r.A \sqsubseteq B$$

where $n \geq 1$ and $A, A_1, \ldots, A_n, B \in \mathbf{C}$.

In the following, we consider an $\mathcal{ELH}$ ontology $\mathcal{O} = (\emptyset, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is in normal form.

A Classification Algorithm for $\mathcal{ELH}$

We discuss the reasoning algorithm for $\mathcal{ELH}$, in preparation to extend it to compute pinpointing formulas.

The classification algorithm for $\mathcal{ELH}$ exhaustively applies the following rules to complete the TBox, where $A, B, C, D, A_1, \dots, A_n \in \mathbf{C}(\mathcal{O})$ and $r, s \in \mathbf{R}(\mathcal{O})$:

$$\frac{}{A \sqsubseteq A}\ \text{(CR1)} \qquad \frac{}{A \sqsubseteq \top}\ \text{(CR2)}$$

$$\frac{A \sqsubseteq A_1 \quad \dots \quad A \sqsubseteq A_n \quad A_1 \sqcap \dots \sqcap A_n \sqsubseteq B}{A \sqsubseteq B}\ \text{(CR3)}$$

$$\frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq C \quad \exists r.C \sqsubseteq D}{A \sqsubseteq D}\ \text{(CR4)} \qquad \frac{A \sqsubseteq \exists r.B \quad r \sqsubseteq s}{A \sqsubseteq \exists s.B}\ \text{(CR5)}$$

If the premises are in $\mathcal{O}$ and the conclusion is not already in $\mathcal{O}$, then add the conclusion to $\mathcal{O}$.

A Classification Algorithm for $\mathcal{ELH}$

Lemma ($\mathcal{ELH}$ classification algorithm)

The classification algorithm for $\mathcal{ELH}$ terminates in time polynomial in the size of $\mathcal{O}$. For all $A, B \in \mathbf{C}(\mathcal{O})$, the resulting ontology $\mathcal{O}'$ contains the GCI $A \sqsubseteq B$ iff $\mathcal{O} \models A \sqsubseteq B$.

Proof: (Baader, Peñaloza, Suntisrivaraporn, 2007) □

We extend this algorithm by labeling all axioms with monotone Boolean formulas over $lab(\mathcal{O})$, with the goal of computing pinpointing formulas for all GCIs $A \sqsubseteq B$ between concept names $A, B \in \mathbf{C}(\mathcal{O})$.

We denote a labeled axiom by α^{φ}, where α is an axiom and φ is a monotone Boolean formula over $lab(\mathcal{O})$.

Initially, the labeled ontology $\mathcal{O}_\ell$ contains all labeled axioms $\beta^{lab(\beta)}$, where $\beta \in \mathcal{O}$.

The Pinpointing Algorithm for $\mathcal{ELH}$

$$\frac{}{(A \sqsubseteq A)^{\texttt{true}}}\ \text{(CR1)} \qquad \frac{}{(A \sqsubseteq \top)^{\texttt{true}}}\ \text{(CR2)}$$

$$\frac{(A \sqsubseteq A_1)^{\varphi_1} \quad \dots \quad (A \sqsubseteq A_n)^{\varphi_n} \quad (A_1 \sqcap \dots \sqcap A_n \sqsubseteq B)^{\varphi}}{(A \sqsubseteq B)^{\varphi_1 \wedge \dots \wedge \varphi_n \wedge \varphi}}\ \text{(CR3)}$$

$$\frac{(A \sqsubseteq \exists r.B)^{\varphi_1} \quad (B \sqsubseteq C)^{\varphi_2} \quad (\exists r.C \sqsubseteq D)^{\varphi_3}}{(A \sqsubseteq D)^{\varphi_1 \wedge \varphi_2 \wedge \varphi_3}}\ \text{(CR4)}$$

$$\frac{(A \sqsubseteq \exists r.B)^{\varphi_1} \quad (r \sqsubseteq s)^{\varphi_2}}{(A \sqsubseteq \exists s.B)^{\varphi_1 \wedge \varphi_2}}\ \text{(CR5)}$$

For a new conclusion α^{φ}:

- if $\mathcal{O}_\ell$ does not already contain a labeled axiom α^{ψ}, then add α^{φ} to $\mathcal{O}_\ell$
- if $\mathcal{O}_\ell$ already contains α^{ψ} and φ does not imply ψ, then replace α^{ψ} in $\mathcal{O}_\ell$ with $\alpha^{\psi \vee \varphi}$.

Example of Pinpointing for $\mathcal{ELH}$

Ontology $\mathcal{O}$:

$$A \sqsubseteq \exists r.A \qquad A \sqsubseteq Y \qquad \exists r.Y \sqsubseteq B \qquad Y \sqsubseteq B$$

Labeled ontology $\mathcal{O}_\ell$:

$$(A \sqsubseteq \exists r.A)^{p_1} \qquad (A \sqsubseteq Y)^{p_2} \qquad (\exists r.Y \sqsubseteq B)^{p_3} \qquad (Y \sqsubseteq B)^{p_4}$$

Rule applications:

$$\frac{(A \sqsubseteq Y)^{p_2} \qquad (Y \sqsubseteq B)^{p_4}}{(A \sqsubseteq B)^{p_2 \wedge p_4}} \text{ (CR3)}$$

$$\frac{(A \sqsubseteq \exists r.A)^{p_1} \qquad (A \sqsubseteq Y)^{p_2} \qquad (\exists r.Y \sqsubseteq B)^{p_3}}{(A \sqsubseteq B)^{(p_2 \wedge p_4) \vee (p_1 \wedge p_2 \wedge p_3)}} \text{ (CR4)}$$

$(p_2 \wedge p_4) \vee (p_1 \wedge p_2 \wedge p_3)$ is a pinpointing formula for $A \sqsubseteq B$ in $\mathcal{O}$.

Correctness of Pinpointing for $\mathcal{ELH}$

Lemma ($\mathcal{ELH}$ Pinpointing Algorithm)

The pinpointing algorithm for $\mathcal{ELH}$ terminates in time exponential in the size of $\mathcal{O}$.

For all $A, B \in \mathbf{C}(\mathcal{O})$, the resulting labeled ontology $\mathcal{O}'_\ell$ contains a labeled GCI $(A \sqsubseteq B)^\varphi$ iff $\mathcal{O} \models A \sqsubseteq B$.

Moreover, if $\mathcal{O} \models A \sqsubseteq B$, then the label of $A \sqsubseteq B$ in $\mathcal{O}'_\ell$ is a pinpointing formula for $A \sqsubseteq B$ in $\mathcal{O}$.

Proof: Blackboard. $\square$

Dealing with Normalization (I)

The $\mathcal{ELH}$ pinpointing algorithm can only deal with TBoxes in normal form.

If the TBox is not in normal form, it can be normalized (modulo AC):

$$
\begin{array}{rcl}
\hat{C} \sqsubseteq \hat{D} & \longrightarrow & \hat{C} \sqsubseteq A,\ A \sqsubseteq \hat{D} \\[1em]
B \sqsubseteq C_1 \sqcap \cdots \sqcap C_n & \longrightarrow & B \sqsubseteq C_1,\ \ldots,\ B \sqsubseteq C_n \\
B_1 \sqcap \cdots \sqcap B_n \sqcap \hat{C} \sqsubseteq D & \longrightarrow & \hat{C} \sqsubseteq A,\ B_1 \sqcap \cdots \sqcap B_n \sqcap A \sqsubseteq D \\[1em]
B \sqsubseteq \exists r.\hat{C} & \longrightarrow & B \sqsubseteq \exists r.A,\ A \sqsubseteq \hat{C} \\
\exists r.\hat{C} \sqsubseteq D & \longrightarrow & \hat{C} \sqsubseteq A,\ \exists r.A \sqsubseteq D
\end{array}
$$

where $\hat{C}, \hat{D}$ are not concept names, and A is a fresh concept name.

In the following, let $\mathcal{O}$ be an ontology, and $\mathcal{O}'$ be obtained from $\mathcal{O}$ by exhaustive application of the normalization rules.

Dealing with Normalization (II)

Lemma (Correctness of Normalization)

For all $A, B \in \mathbf{C}(\mathcal{O})$, we have $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{O}' \models A \sqsubseteq B$.

Proof: (Baader, Lutz, Horrocks, Sattler, 2017) $\square$

A pinpointing formula φ' over $\mathcal{O}'$ refers to axioms that are not in $\mathcal{O}$, so φ' cannot directly be used to find justifications in $\mathcal{O}$!

Remedy: Find out which original axioms produced each normalized axiom:

The sources of $\beta \in \mathcal{O}'$ are all axioms of $\mathcal{O}$ from which β was obtained.

$\mathcal{O} = \{A \sqsubseteq B_1 \sqcap B_2,\ A \sqsubseteq B_2 \sqcap B_3,\ A \sqsubseteq C\}$

$\mathcal{O}' = \{A \sqsubseteq B_1,\ A \sqsubseteq B_2,\ A \sqsubseteq B_3,\ A \sqsubseteq C\}$

The sources of $A \sqsubseteq B_2$ are $A \sqsubseteq B_1 \sqcap B_2$ and $A \sqsubseteq B_2 \sqcap B_3$.

The only source of $A \sqsubseteq C$ is $A \sqsubseteq C$ itself.

Dealing with Normalization (III)

Lemma (Pinpointing and Normalization)

Let $A, B \in \mathbf{C}(\mathcal{O})$, and φ' be a pinpointing formula for $A \sqsubseteq B$ in $\mathcal{O}'$.

Let φ be obtained from φ' by replacing each $lab(\beta)$, $\beta \in \mathcal{O}'$, by $lab(\alpha_1) \vee \dots \vee lab(\alpha_n)$, where $\alpha_1, \dots, \alpha_n$ are all sources of β in $\mathcal{O}$.

Then φ is a pinpointing formula for $A \sqsubseteq B$ in $\mathcal{O}$.

Proof: Blackboard. □

Complexity of Computing all Justifications

Computing all justifications still requires exponential time:

$$\{A \sqsubseteq B_1 \sqcap C_1,\ B_1 \sqsubseteq B_2 \sqcap C_2,\ \dots,\ B_{n-1} \sqsubseteq B_n \sqcap C_n,\ B_n \sqsubseteq D$$
$$C_1 \sqsubseteq B_2 \sqcap C_2,\ \dots,\ C_{n-1} \sqsubseteq B_n \sqcap C_n,\ C_n \sqsubseteq D\}$$

Exponentially many justifications
= exponentially many minimal valuations of a pinpointing formula

An algorithm runs in output polynomial time if its runtime is bounded by a polynomial function in the size of the input and the output.

Problem (EnumerateAllJustifications)

Input: Ontology $\mathcal{O}$, axiom α with $\mathcal{O} \models \alpha$

Output: The set $\mathrm{Just}_{\mathcal{O}}(\alpha)$

Is there an output polynomial algorithm for **EnumerateAllJustifications**?

Enumerating vs. Checking Justifications

Problem (CheckJustification)

Input: Ontology $\mathcal{O}$, axiom α with $\mathcal{O} \models \alpha$, set $\mathcal{J} \subseteq 2^{\mathcal{O}}$

Output: "yes" if $\mathcal{J} = \mathsf{Just}_{\mathcal{O}}(\alpha)$, otherwise "no"

Lemma

If **EnumerateAllJustifications** can be solved in output polynomial time, then **CheckJustification** can be decided in polynomial time.

Proof: Blackboard. □

CheckJustification cannot be decided in polynomial time, unless $\mathrm{P} = \mathrm{NP}$:

Lemma

CheckJustification is co-NP-hard in $\mathcal{ELH}$.

Proof: Blackboard. (Peñaloza, Sertkaya, 2017) □

Final Remarks on Justifications

- EL2MUS: Implementation for $\mathcal{ELH}$ via encoding into SAT
- Justifications are used in non-monotonic reasoning and for measuring the "degree of inconsistency" of ontologies.
- Justifications may not be enough to explain the error, even to DL experts:

> $\{\mathsf{Cow} \sqsubseteq \mathsf{Mammal},\ \mathsf{Mammal} \sqsubseteq \mathsf{Animal},$
> $\mathsf{Cow} \equiv \forall \mathsf{eats}.\mathsf{Grass},\ \mathrm{Dom}(\mathsf{eats}) \sqsubseteq \mathsf{Animal}\} \models \mathsf{Grass} \sqsubseteq \mathsf{Animal}$

because it entails $\neg\exists \mathsf{eats}.\top \sqsubseteq \mathsf{Cow}$, $\exists \mathsf{eats}.\top \sqsubseteq \mathsf{Animal}$, and $\top \equiv \mathsf{Animal}$.

> To explain the error, it needs to be further explained why the axiom follows from the justification.

- Removing $\mathsf{Cow} \equiv \forall \mathsf{eats}.\mathsf{Grass}$ to repair the error may be too much:

> We could instead weaken the first axiom to $\mathsf{Cow} \sqsubseteq \forall \mathsf{eats}.\mathsf{Grass}$, or replace it with $\mathsf{Cow} \equiv \mathsf{Mammal} \sqcap \forall \mathsf{eats}.\mathsf{Grass}$.

Outline

4.2 Modularization

Reuse of Ontologies

We want to develop a new ontology

$$\mathcal{P} = \{\mathsf{LogicCourse} \equiv \mathsf{Course} \sqcap \exists \mathsf{focus}.\mathsf{Logic}, \quad \exists \mathsf{focus}.\top \sqsubseteq \mathsf{Course}, \\ \mathsf{DLSeminar} \equiv \mathsf{Seminar} \sqcap \exists \mathsf{focus}.\mathsf{DL}, \quad \mathsf{Seminar} \sqsubseteq \mathsf{Course}\}$$

and reuse the knowledge from an existing ontology

$$\mathcal{O} = \{\mathsf{DL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall}, \\ \mathsf{FOL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure}, \quad \mathsf{FOL} \sqsubseteq \mathsf{Logic}, \\ \mathsf{Language} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall} \sqsubseteq \mathsf{FOL}, \\ \mathsf{OWL} \sqsubseteq \mathsf{W3CStandard} \sqcap \exists \mathsf{basedOn}.\mathsf{DL}\}$$

but are interested only in the characterization of Logic and DL from $\mathcal{O}$, e.g.,

$$\mathcal{O} \models \mathsf{DL} \sqsubseteq \mathsf{Logic}.$$

For convenience, we only consider $\mathcal{ALC}$ in this section.

Modular Reuse of Ontologies

$$\mathcal{P} = \{\text{LogicCourse} \equiv \text{Course} \sqcap \exists\text{focus}.\text{Logic}, \quad \exists\text{focus}.\top \sqsubseteq \text{Course},$$
$$\text{DLSeminar} \equiv \text{Seminar} \sqcap \exists\text{focus}.\text{DL}, \quad \text{Seminar} \sqsubseteq \text{Course}\}$$

$$\mathcal{O} = \{\text{DL} \equiv \text{Language} \sqcap \exists\text{semantics}.\text{FOStructure} \sqcap \exists\text{quantifier}.\text{Forall},$$
$$\text{FOL} \equiv \text{Language} \sqcap \exists\text{semantics}.\text{FOStructure}, \quad \text{FOL} \sqsubseteq \text{Logic},$$
$$\text{Language} \sqcap \exists\text{quantifier}.\text{Forall} \sqsubseteq \text{FOL},$$
$$\text{OWL} \sqsubseteq \text{W3CStandard} \sqcap \exists\text{basedOn}.\text{DL}\}$$

We consider only a subvocabulary $\Sigma \subseteq \mathbf{C}(\mathcal{O}) \cup \mathbf{R}(\mathcal{O}) \cup \mathbf{I}(\mathcal{O})$, e.g., $\{\text{DL}, \text{Logic}\}$.

We want to use the relevant part $\mathcal{M}$ of $\mathcal{O}$ together with new axioms $\mathcal{P}$:

- $\mathcal{M}$ completely describes the names in Σ (it is a Σ-module of $\mathcal{O}$).
- $\mathcal{P}$ does not affect the semantics of the names in Σ (it is Σ-safe).

(Konev, Lutz, Walther, Wolter, 2009)
(Cuenca Grau, Horrocks, Kazakov, Sattler, 2009)

Conservative Extensions

Modules and safety are defined based on conservative extensions.

A signature is a subset of $\mathbf{C} \cup \mathbf{R} \cup \mathbf{I}$ that contains $\top$ and $\bot$.

For an ontology $\mathcal{O}$ or axiom α, $\mathsf{sig}(\mathcal{O})$/$\mathsf{sig}(\alpha)$ is the signature of $\mathcal{O}/\alpha$, i.e., the set of concept, role, and individual names occurring in $\mathcal{O}/\alpha$.

For two interpretations $\mathcal{I}, \mathcal{J}$, we write $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$ if $\Delta^\mathcal{I} = \Delta^\mathcal{J}$ and $X^\mathcal{I} = X^\mathcal{J}$ for all $X \in \Sigma$, i.e., $\mathcal{I}$ and $\mathcal{J}$ agree on the interpretation of the names in Σ.

Let $\mathcal{O}_1 \subseteq \mathcal{O}_2$ be two ontologies and Σ be a signature.

Then $\mathcal{O}_2$ is a Σ-conservative extension (Σ-CE) of $\mathcal{O}_1$ if,
for every model $\mathcal{I}$ of $\mathcal{O}_1$, there is a model $\mathcal{J}$ of $\mathcal{O}_2$ such that $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$.

The axioms in $\mathcal{O}_2 \setminus \mathcal{O}_1$ do not affect the semantics of the names from Σ.

Note that $\mathcal{O}_1$ can contain more names than those in Σ, and their semantics is allowed to change.

Safety and Modules

Let $\mathcal{O}_1 \subseteq \mathcal{O}_2$ be two ontologies and Σ be a signature.

Then $\mathcal{O}_2$ is a Σ-conservative extension (Σ-CE) of $\mathcal{O}_1$ if,
for every model $\mathcal{I}$ of $\mathcal{O}_1$, there is a model $\mathcal{J}$ of $\mathcal{O}_2$ such that $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$.

Suppose we want to import the knowledge about Σ from $\mathcal{O}$ into $\mathcal{P}$.

$\mathcal{P}$ is Σ-safe if,
for all ontologies $\mathcal{O}$ with $\mathrm{sig}(\mathcal{P}) \cap \mathrm{sig}(\mathcal{O}) \subseteq \Sigma$,
$\mathcal{P} \cup \mathcal{O}$ is a Σ-CE of $\mathcal{O}$.

$\mathcal{P}$ does not affect the semantics of the names in Σ given by $\mathcal{O}$.

A subset $\mathcal{M} \subseteq \mathcal{O}$ is a Σ-module of $\mathcal{O}$ if,
for all ontologies $\mathcal{P}$ with $\mathrm{sig}(\mathcal{P}) \cap \mathrm{sig}(\mathcal{O}) \subseteq \Sigma$,
$\mathcal{P} \cup \mathcal{O}$ is a Σ-CE of $\mathcal{P} \cup \mathcal{M}$.

When we are only interested in Σ, we can import $\mathcal{M}$ instead of $\mathcal{O}$.

Robustness under Replacement

Before we look at some examples, we first simplify the definitions of safety and modules, by using the following property of Σ-CEs:

Lemma (Replacement)

Let $\mathcal{O}, \mathcal{O}_1, \mathcal{O}_2$ be three ontologies and Σ a signature with $\mathrm{sig}(\mathcal{O}) \cap \mathrm{sig}(\mathcal{O}_1 \cup \mathcal{O}_2) \subseteq \Sigma$.

If $\mathcal{O}_2$ is a Σ-CE of $\mathcal{O}_1$, then $\mathcal{O} \cup \mathcal{O}_2$ is a Σ-CE of $\mathcal{O} \cup \mathcal{O}_1$.

Proof: Blackboard. □

Lemma (Characterization of Safety)

$\mathcal{P}$ is Σ-safe iff $\mathcal{P}$ is a Σ-CE of $\emptyset$.

Lemma (Characterization of Modules)

$\mathcal{M}$ is a Σ-module of $\mathcal{O}$ iff $\mathcal{O}$ is a Σ-CE of $\mathcal{M}$.

Proof: Exercise. □

Example: Safety

$$\begin{aligned}\mathcal{P} = \{&\mathsf{LogicCourse} \equiv \mathsf{Course} \sqcap \exists\mathsf{focus}.\mathsf{Logic}, && \exists\mathsf{focus}.\top \sqsubseteq \mathsf{Course},\\ &\mathsf{DLSeminar} \equiv \mathsf{Seminar} \sqcap \exists\mathsf{focus}.\mathsf{DL}, && \mathsf{Seminar} \sqsubseteq \mathsf{Course}\}\end{aligned}$$

Is $\mathcal{P}$ Σ-safe for $\Sigma = \{\top, \bot, \mathsf{DL}, \mathsf{Logic}\}$? Yes! (see blackboard)

$$\begin{aligned}\mathcal{P}_2 = \{&\mathsf{LogicCourse} \equiv \mathsf{Course} \sqcap \exists\mathsf{focus}.\mathsf{Logic}, && \exists\mathsf{focus}.\top \sqsubseteq \mathsf{Course},\\ &\mathsf{DLSeminar} \equiv \mathsf{Seminar} \sqcap \exists\mathsf{focus}.\mathsf{DL}, && \mathsf{Seminar} \sqsubseteq \mathsf{Course},\\ &\mathsf{Course} \sqcap (\mathsf{Logic} \sqcap \mathsf{DL}) \sqsubseteq \bot,\\ &\forall\mathsf{focus}.\mathsf{DL} \sqsubseteq \exists\mathsf{focus}.\mathsf{Logic}\}\end{aligned}$$

is not Σ-safe (see blackboard).

Example: Modules

$$\begin{aligned}\mathcal{O} = \{&\mathsf{DL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall},\\ &\mathsf{FOL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure}, \quad \mathsf{FOL} \sqsubseteq \mathsf{Logic},\\ &\mathsf{Language} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall} \sqsubseteq \mathsf{FOL},\\ &\mathsf{OWL} \sqsubseteq \mathsf{W3CStandard} \sqcap \exists \mathsf{basedOn}.\mathsf{DL}\}\end{aligned}$$

What are the (minimal) Σ-modules of $\mathcal{O}$ for $\Sigma = \{\mathsf{DL}, \mathsf{Logic}\}$?

All justifications of $\mathsf{DL} \sqsubseteq \mathsf{Logic}$! (see blackboard)

$$\begin{aligned}\mathcal{M}_1 = \{&\mathsf{DL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall},\\ &\mathsf{FOL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure}, \quad \mathsf{FOL} \sqsubseteq \mathsf{Logic}\}\\ \mathcal{M}_2 = \{&\mathsf{DL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall},\\ &\mathsf{Language} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall} \sqsubseteq \mathsf{FOL}, \quad \mathsf{FOL} \sqsubseteq \mathsf{Logic}\}\end{aligned}$$

Σ-modules for $\mathcal{O}$ are related to justifications for all possible axioms over Σ.

Example: "Depleting" Modules

The remaining axioms in

$$\mathcal{O} \setminus \mathcal{M}_1 = \{\mathsf{Language} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall} \sqsubseteq \mathsf{FOL},\ \mathsf{OWL} \sqsubseteq \mathsf{W3CStandard} \sqcap \exists \mathsf{basedOn}.\mathsf{DL}\}$$

can still influence the interpretation of $\mathrm{sig}(\mathcal{M}_1) =$ $\{\mathsf{DL}, \mathsf{Logic}, \mathsf{Language}, \mathsf{FOStructure}, \mathsf{Forall}, \mathsf{FOL}, \mathsf{semantics}, \mathsf{quantifier}\}$!

To avoid this, we consider so-called depleting modules of $\mathcal{O}$:

$$\begin{aligned}\mathcal{M}_3 = \{&\mathsf{DL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall},\\ &\mathsf{FOL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure},\quad \mathsf{FOL} \sqsubseteq \mathsf{Logic},\\ &\mathsf{Language} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall} \sqsubseteq \mathsf{FOL}\}\end{aligned}$$

$\mathcal{M}_3$ contains all axioms relevant for its signature.

$\mathcal{O} \setminus \mathcal{M}_3 = \{\mathsf{OWL} \sqsubseteq \mathsf{W3CStandard} \sqcap \exists \mathsf{basedOn}.\mathsf{DL}\}$ is $[\Sigma \cup \mathrm{sig}(\mathcal{M}_3)]$-safe.

Properties of Σ-Conservative Extensions

Before we formally define depleting modules, we introduce some important properties of Σ-CEs:

Lemma (Monotonicity)

Let $\mathcal{O}_2$ be a Σ-CE of $\mathcal{O}_1$.

a) If $\Sigma' \subseteq \Sigma$, then $\mathcal{O}_2$ is a Σ'-CE of $\mathcal{O}_1$.

b) If $\mathcal{O}_1 \subseteq \mathcal{O}_2' \subseteq \mathcal{O}_2$, then $\mathcal{O}_2'$ is a Σ-CE of $\mathcal{O}_1$.

Lemma (Transitivity)

If $\mathcal{O}_2$ is a Σ-CE of $\mathcal{O}_1$ and $\mathcal{O}_3$ is a Σ-CE of $\mathcal{O}_2$, then $\mathcal{O}_3$ is a Σ-CE of $\mathcal{O}_1$.

Lemma (Reflexivity)

For every ontology $\mathcal{O}$ and signature Σ, $\mathcal{O}$ is a Σ-CE of itself.

Proof: Exercise. $\square$

Depleting Modules

$\mathcal{M}$ is a depleting Σ-module of $\mathcal{O}$ if $\mathcal{O} \setminus \mathcal{M}$ is $[\Sigma \cup \mathrm{sig}(\mathcal{M})]$-safe.

Lemma (Depleting Modules are Modules)

If $\mathcal{M}$ is a depleting Σ-module of $\mathcal{O}$, then $\mathcal{M}$ is a Σ-module of $\mathcal{O}$.

Proof: Blackboard. $\square$

Lemma (Depleting Modules Contain All Minimal Modules)

Every depleting Σ-module of $\mathcal{O}$ contains all minimal Σ-modules of $\mathcal{O}$.

Proof: Blackboard. $\square$

- A (minimal) depleting module $\mathcal{M}$ can be very large, because it includes all axioms relevant for the semantics of $\Sigma \cup \mathrm{sig}(\mathcal{M})$.
- A depleting module $\mathcal{M}$ can be maintained separately from $\mathcal{O} \setminus \mathcal{M}$, since one does not have to worry about changes in $\mathcal{M}$ interacting with the remaining axioms in $\mathcal{O} \setminus \mathcal{M}$ (as long as the signature remains the same).

Complexity of Σ-Conservative Extensions

In $\mathcal{EL}$, deciding whether $\mathcal{O}_2$ is a Σ-CE of $\mathcal{O}_1$ is

- undecidable, even if $\mathcal{O}_1 = \emptyset$
- co-NExpTime$^{\text{NP}}$-complete if $\Sigma \subseteq \mathbf{C}$
- in P if $\mathcal{O}_1 = \emptyset$ and $\Sigma \subseteq \mathbf{C}$

> Due to this high complexity, we consider approximations.

Goal: Find another definition of Σ-conservative extensions that has better complexity, but still retains the nice properties (monotonicity, reflexivity, transitivity, robustness under replacement).

First idea: Instead of looking at all models, we consider only the relevant entailments (e.g., $\mathsf{DL} \sqsubseteq \mathsf{Logic}$).

First Approximation: Deductive CE

Let $\mathcal{O}_1 \subseteq \mathcal{O}_2$ be two ontologies and Σ be a signature.

Then $\mathcal{O}_2$ is a deductive Σ-conservative extension (d-Σ-CE) of $\mathcal{O}_1$ if, for every GCI α with $\mathrm{sig}(\alpha) \subseteq \Sigma$, $\mathcal{O}_2 \models \alpha$ implies $\mathcal{O}_1 \models \alpha$.

- The previous definition of Σ-CE is also called model-based Σ-CE (m-Σ-CE).
- In contrast to m-Σ-CEs, d-Σ-CEs only consider consequences that can be formulated in a given description logic, e.g., $\mathcal{ALC}$.

d-Σ-safety and d-Σ-modules are defined as before, by replacing m-Σ-CE with d-Σ-CE.

Deductive vs. Model-Based CE

Lemma

If $\mathcal{O}_2$ is an m-Σ-CE of $\mathcal{O}_1$, then it is also a d-Σ-CE of $\mathcal{O}_1$.

Proof: Exercise. $\square$

In particular, finding an axiom α with $\mathsf{sig}(\alpha) \subseteq \Sigma$, $\mathcal{O}_2 \models \alpha$, and $\mathcal{O}_1 \not\models \alpha$ proves that $\mathcal{O}_2$ is not an m-Σ-CE of $\mathcal{O}_1$.

The other direction does not hold:

$\mathcal{M} = \{\top \sqsubseteq \exists r.\top \sqcap \exists s.\top\}$ is a d-Σ-module of
$\mathcal{O} = \mathcal{M} \cup \{\top \sqsubseteq \exists r.A \sqcap \exists s.\neg A\}$ for $\Sigma = \{r, s\}$.

But there are models of $\mathcal{M}$ that cannot be extended to models of $\mathcal{O}$ when the interpretation of r and s is fixed.

This means that the sets of minimal modules for the two (model-based and deductive) definitions are incomparable.

Flashback: Examples

$$\mathcal{P}_2 = \{\text{LogicCourse} \equiv \text{Course} \sqcap \exists\text{focus}.\text{Logic}, \quad \exists\text{focus}.\top \sqsubseteq \text{Course},$$
$$\text{DLSeminar} \equiv \text{Seminar} \sqcap \exists\text{focus}.\text{DL}, \quad \text{Seminar} \sqsubseteq \text{Course},$$
$$\text{Course} \sqcap (\text{Logic} \sqcap \text{DL}) \sqsubseteq \bot,$$
$$\forall\text{focus}.\text{DL} \sqsubseteq \exists\text{focus}.\text{Logic}\}$$

entails $\text{Logic} \sqcap \text{DL} \sqsubseteq \bot$, which is not entailed by $\emptyset$. Thus, it is not d-Σ-safe, and therefore not m-Σ-safe.

All d-Σ-modules (and all m-Σ-modules) of

$$\mathcal{O} = \{\text{DL} \equiv \text{Language} \sqcap \exists\text{semantics}.\text{FOStructure} \sqcap \exists\text{quantifier}.\text{Forall},$$
$$\text{FOL} \equiv \text{Language} \sqcap \exists\text{semantics}.\text{FOStructure}, \quad \text{FOL} \sqsubseteq \text{Logic},$$
$$\text{Language} \sqcap \exists\text{quantifier}.\text{Forall} \sqsubseteq \text{FOL},$$
$$\text{OWL} \sqsubseteq \text{W3CStandard} \sqcap \exists\text{basedOn}.\text{DL}\}$$

must entail $\text{DL} \sqsubseteq \text{Logic}$.

Properties of d-Σ-CE

The following properties also hold for d-Σ-CEs (Exercise):

- Monotonicity
- Transitivity
- Reflexivity
- (Depleting modules are modules)

The following properties do not hold:

- Robustness under replacement
- Characterizations of safety and modules

$\mathcal{O}_2 = \{A \sqsubseteq \exists r.B\}$ is a d-Σ-CE of $\emptyset$, for $\Sigma = \{A, B\}$.

For $\mathcal{O} = \{A \equiv \top, B \equiv \bot\}$, we have $\mathcal{O} \cup \mathcal{O}_2 \models \top \sqsubseteq \bot$, but $\mathcal{O} \not\models \top \sqsubseteq \bot$, i.e., $\mathcal{O} \cup \mathcal{O}_2$ is not a d-Σ-CE of $\mathcal{O}$.

Complexity of d-Σ-CE

Deciding d-Σ-CE:

- undecidable for $\mathcal{SROIQ}$, i.e., OWL 2 DL
- 2-ExpTime-complete for $\mathcal{ALC}$
- ExpTime-complete for $\mathcal{EL}$

Deciding d-Σ-modules:

- undecidable for $\mathcal{ALC}$ with nominals

> These complexities are better, but still quite high.

New idea for approximation: Concentrate on minimal depleting modules, i.e., on (approximately) deciding safety.

Second Approximation: Locality-Based CE

Can we find a new definition of Σ-conservative extensions for which we can efficiently check whether $\mathcal{M}$ is a depleting module of $\mathcal{O}$, i.e., whether $\mathcal{O} \setminus \mathcal{M}$ is $[\Sigma \cup \mathsf{sig}(\mathcal{M})]$-safe?

Locality: Individually check each axiom to see whether it affects the semantics of symbols from Σ (i.e., whether it is Σ-safe or not).

An interpretation $\mathcal{I}$ is $\emptyset$-Σ-local if $X^{\mathcal{I}} = \emptyset$ for all $X \in (\mathbf{C} \cup \mathbf{R}) \setminus \Sigma$.

An axiom α is $\emptyset$-Σ-local if it is satisfied in every $\emptyset$-Σ-local interpretation.

An ontology $\mathcal{O}$ is $\emptyset$-Σ-local if all axioms $\alpha \in \mathcal{O}$ are $\emptyset$-Σ-local.

- We only need to check each axiom $\alpha \in \mathcal{O}$ separately.
- To test locality, we can replace all concept names outside of Σ with $\bot$ (because in $\emptyset$-Σ-local interpretations they are equivalent to $\bot$), and similarly for role names.

Testing Locality

For GCIs and concept assertions α, the axiom $\alpha|_\Sigma$ is obtained from α by replacing subconcepts as follows:

- every $A \in \mathbf{C} \setminus \Sigma$ with $\bot$;
- every $\exists r.C$, where $r \in \mathbf{R} \setminus \Sigma$, with $\bot$;
- every $\forall r.C$, where $r \in \mathbf{R} \setminus \Sigma$, with $\top$.

If α is a role assertion $(a, b) : r$ with $r \in \mathbf{R} \setminus \Sigma$, then $\alpha|_\Sigma$ is defined as $\top \sqsubseteq \bot$ (since it can never be satisfied by a $\emptyset$-Σ-local interpretation).

Lemma (Testing $\emptyset$-Locality)

An axiom α is $\emptyset$-Σ-local iff $\emptyset \models \alpha|_\Sigma$.

Proof: Exercise. □

Example for $\emptyset$-Locality

$\Sigma =$ $\{\mathsf{DL}, \mathsf{Logic}, \mathsf{Language}, \mathsf{FOStructure}, \mathsf{Forall}, \mathsf{FOL}, \mathsf{semantics}, \mathsf{quantifier}\}$

$\mathsf{OWL} \sqsubseteq \mathsf{W3CStandard} \sqcap \exists \mathsf{basedOn}.\mathsf{DL}$ is replaced by $\bot \sqsubseteq \bot \sqcap \bot$, which is entailed by the empty ontology $\emptyset$.

Thus, the axiom is $\emptyset$-Σ-local.

$\emptyset$-Σ-Conservative Extensions

Let $\mathcal{O}_1 \subseteq \mathcal{O}_2$ be two ontologies and Σ be a signature.

Then $\mathcal{O}_2$ is an $\emptyset$-locality-based Σ-conservative extension ($\emptyset$-Σ-CE) of $\mathcal{O}_1$ if $\mathcal{O}_2 \setminus \mathcal{O}_1$ is $\emptyset$-[$\Sigma \cup \mathrm{sig}(\mathcal{O}_1)$]-local.

$\emptyset$-Σ-CEs also satisfy all the nice properties we wanted: monotonicity, transitivity, reflexivity, and robustness under replacement, and hence the results about depleting modules and simplifying characterizations of safety and modules.

If we define safety and modules as before, we obtain:

$\mathcal{P}$ is $\emptyset$-Σ-safe iff it is $\emptyset$-Σ-local.

$\mathcal{M}$ is an $\emptyset$-Σ-module of $\mathcal{O}$ iff $\mathcal{O} \setminus \mathcal{M}$ is $\emptyset$-[$\Sigma \cup \mathrm{sig}(\mathcal{M})$]-local.

By definition, all $\emptyset$-Σ-modules are depleting.

$\emptyset$-Locality-Based vs. Model-Based CE

Lemma

If $\mathcal{O}_2$ is an $\emptyset$-Σ-CE of $\mathcal{O}_1$, then $\mathcal{O}_2$ is an m-Σ-CE of $\mathcal{O}_1$.

Proof: Blackboard. $\square$

Again, the other direction does not hold:

$$\mathcal{M}_1 = \{\mathsf{DL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall}, \quad \mathsf{FOL} \equiv \mathsf{Language} \sqcap \exists \mathsf{semantics}.\mathsf{FOStructure}, \quad \mathsf{FOL} \sqsubseteq \mathsf{Logic}\}$$

is an m-Σ-module of $\mathcal{O}$ for $\Sigma = \{\mathsf{DL}, \mathsf{Logic}\}$, but it is not depleting.

The axiom $\mathsf{Language} \sqcap \exists \mathsf{quantifier}.\mathsf{Forall} \sqsubseteq \mathsf{FOL}$ is not $\emptyset$-$[\Sigma \cup \mathrm{sig}(\mathcal{M}_1)]$-local.

Thus, $\mathcal{M}_1$ is not an $\emptyset$-Σ-module of $\mathcal{O}$.

Complexity of $\emptyset$-Σ-CE

The complexity of the locality test is the same as for axiom entailment:

- 2-NExpTime-complete for $\mathcal{SROIQ}$, i.e., OWL 2 DL
- ExpTime-complete for $\mathcal{ALC}$
- P-complete for $\mathcal{EL}$

Can we compute “useful” (small, depleting) modules even faster?

Third Approximation: Syntactic-Locality-Based CE

Idea: To check that $\emptyset \models \bot \sqsubseteq \bot \sqcap \bot$, we do not need a $\mathcal{SROIQ}$ reasoner.

We can instead use a simple syntactic check on the axiom
OWL $\sqsubseteq$ **W3CStandard** $\sqcap \exists$**basedOn**.**DL** to determine that it is local.

Intuition: We collect in $\mathbf{C}_\Sigma^\bot$ ($\mathbf{C}_\Sigma^\top$) all concepts for which it is "easy to prove" that they are equivalent to $\bot$ ($\top$). For example, if $C \in \mathbf{C}_\Sigma^\bot$ and D is another concept, then $C \sqcap D$ also belongs to $\mathbf{C}_\Sigma^\bot$.

$$\begin{aligned}
\mathbf{C}_\Sigma^\bot &::= \bot \mid A^\bot \mid \neg C^\top \mid C \sqcap C^\bot \mid C_1^\bot \sqcup C_2^\bot \mid \exists r^\bot.C \mid \exists r.C^\bot \\
\mathbf{C}_\Sigma^\top &::= \top \mid \neg C^\bot \mid C \sqcup C^\top \mid C_1^\top \sqcap C_2^\top \mid \forall r^\bot.C \mid \forall r.C^\top
\end{aligned}$$

where $A^\bot \in \mathbf{C} \setminus \Sigma$, $C_{(i)}^\bot \in \mathbf{C}_\Sigma^\bot$, $C_{(i)}^\top \in \mathbf{C}_\Sigma^\top$, $r^\bot \in \mathbf{R} \setminus \Sigma$, $r \in \Sigma$, and C is a concept.

An axiom α is $\bot$-Σ-local if it is of the form $C \sqsubseteq C^\top$, $C^\bot \sqsubseteq C$, or $a : C^\top$.

An ontology $\mathcal{O}$ is $\bot$-Σ-local if all axioms $\alpha \in \mathcal{O}$ are $\bot$-Σ-local.

$\bot$-Locality-Based vs. $\emptyset$-Locality-Based CE

We can define $\bot$-Σ-CE, $\bot$-Σ-safety, and $\bot$-Σ-modules as we did for $\emptyset$-locality.

$\bot$-Σ-CEs are monotone, transitive, reflexive, and robust under replacement.

Lemma

If $\mathcal{O}_2$ is a $\bot$-Σ-CE of $\mathcal{O}_1$, then $\mathcal{O}_2$ is an $\emptyset$-Σ-CE of $\mathcal{O}_1$.

Proof: Blackboard. $\square$

As usual, the other direction does not hold:

$\exists r.\neg A \sqsubseteq \exists r.\neg B$ is $\emptyset$-$\{r\}$-local since $\emptyset \models \exists r.\neg\bot \sqsubseteq \exists r.\neg\bot$,
but it is not $\bot$-$\{r\}$-local.

$\Sigma =$
$\{$DL, Logic, Language, FOStructure, Forall, FOL, semantics, quantifier$\}$

OWL $\sqsubseteq$ W3CStandard $\sqcap$ $\exists$basedOn.DL is $\bot$-Σ-local since it is of the form $C^\bot \sqsubseteq C$, because OWL $\notin \Sigma$.

Overview of Conservative Extensions

Name	synt. locality $\Rightarrow$	sem. locality $\Rightarrow$	model-based $\Rightarrow$	deductive
Symbol	$\perp$	$\emptyset$	m	d
Monotonicity, Transitivity, Reflexivity	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
Robustness under Replacement	$\checkmark$	$\checkmark$	$\checkmark$	$\times$
All Modules are Depleting	$\checkmark$	$\checkmark$	$\times$	$\times$
Complexity (for $\mathcal{ALC}$)	P	ExpTime	undecidable	2-ExpTime

Extracting Depleting Modules

- Instead of deciding whether $\mathcal{M}$ is an x-Σ-module of $\mathcal{O}$, we want to extract a minimal x-Σ-module from $\mathcal{O}$ (for some x $\in \{\mathsf{m}, \mathsf{d}, \emptyset, \bot\}$).
- For depleting modules, we only need to ensure that $\mathcal{O} \setminus \mathcal{M}$ is x-[$\Sigma \cup \mathsf{sig}(\mathcal{M})$]-safe.
- To do this, we start from the empty set $\mathcal{M} = \emptyset$, and iteratively add all axioms to $\mathcal{M}$ that cause $\mathcal{O} \setminus \mathcal{M}$ to violate the safety property.
- For this, we only need a black-box that can check x-Σ-safety.

An Algorithm for Extracting Depleting Modules

Algorithm (Black-Box Module Extraction)

Input: Ontology $\mathcal{O}$, signature Σ, $\mathsf{x} \in \{\mathsf{m}, \mathsf{d}, \emptyset, \bot\}$

Output: A depleting x-Σ-module of $\mathcal{O}$

- $\mathcal{M} := \emptyset$; $\mathcal{W} := \emptyset$
- While $\mathcal{M} \cup \mathcal{W} \neq \mathcal{O}$:
  - Choose an axiom $\alpha \in \mathcal{O} \setminus (\mathcal{M} \cup \mathcal{W})$
  - $\mathcal{W} := \mathcal{W} \cup \{\alpha\}$
  - If $\mathcal{W}$ is not x-$[\Sigma \cup \mathsf{sig}(\mathcal{M})]$-safe, then
    - $\mathcal{M} := \mathcal{M} \cup \{\alpha\}$; $\mathcal{W} := \emptyset$
- Return $\mathcal{M}$

When extending $\mathcal{M}$, we need to reset $\mathcal{W}$ since the signature of $\mathcal{M}$ has changed, and hence $\mathcal{W}$ (even without α) may not be x-$[\Sigma \cup \mathsf{sig}(\mathcal{M})]$-safe anymore.

Correctness

Lemma (Correctness of Black-Box Module Extraction)

If x-safety is decidable, the black-box module extraction algorithm computes a depleting x-Σ-module of $\mathcal{O}$.

Proof: Blackboard. □

The runtime of the algorithm is only polynomially larger than that of the x-safety test.

Lemma (Uniqueness of Depleting Modules)

If x-CE are monotone and robust under replacement, then there is a unique minimal depleting x-Σ-module of $\mathcal{O}$, which is computed by the black-box module extraction algorithm.

Proof: Blackboard. □

Extracting Locality-Based Modules

For locality-based notions of safety, the safety check for $\mathcal{W}$ can be replaced by a safety check for α.

Algorithm (Locality-Based Module Extraction)

Input: Ontology $\mathcal{O}$, signature Σ, $x \in \{\emptyset, \bot\}$

Output: The unique minimal (depleting) x-Σ-module of $\mathcal{O}$

- $\mathcal{M} := \emptyset$; $\mathcal{W} := \emptyset$
- While $\mathcal{M} \cup \mathcal{W} \neq \mathcal{O}$:
  - Choose an axiom $\alpha \in \mathcal{O} \setminus (\mathcal{M} \cup \mathcal{W})$
  - $\mathcal{W} := \mathcal{W} \cup \{\alpha\}$
  - If α is not x-$[\Sigma \cup \mathsf{sig}(\mathcal{M})]$-local, then
    - $\mathcal{M} := \mathcal{M} \cup \{\alpha\}$; $\mathcal{W} := \emptyset$
- Return $\mathcal{M}$

Correctness

Lemma (Correctness of Locality-Based Module Extraction)

The locality-based module extraction algorithm computes the unique minimal (depleting) x-Σ-module of $\mathcal{O}$.

For $\perp$-locality, the safety check can be done in polynomial time, which implies the following:

Corollary

The unique minimal (depleting) $\perp$-Σ-module of $\mathcal{O}$ can be computed in polynomial time in the size of $\mathcal{O}$.

Extracting Modules in Practice

- m-Σ-modules can only be computed in special cases, e.g., for $\mathcal{EL}$ and $\Sigma \subseteq \mathbf{C}$
- $\bot$-Σ-modules can be computed much faster, even for large and expressive ontologies
- Minimal $\bot$-Σ-modules are also m-Σ-modules, but not necessarily minimal m-Σ-modules
- In an evaluation, minimal $\bot$-Σ-modules differed from the minimal depleting m-Σ-modules in 27% of the cases, with size differences up to 80% (varying with the structure of the ontology)

  (Del Vescovo, Klinov, Parsia, Sattler, Schneider, Tsarkov, 2013)